Residential iPad User Interface System

Author: Barry Gordon
Website: http://the-gordons.net/ [http://the-gordons.net/]
CommandFusion Products: iViewer 4

All information and GUI's in this case study are by Barry Gordon himself. These are put together to suit his unique requirements and are designed to reflect his personal tastes. All visual and back end system design are Barry's own and should be looked as an example, CommandFusion recommends you design a system that suits your own requirements and tastes.

This case study is divided up into 3 pages:

1. System Overview - a detailed overview of the house, and what makes up the system.
2. Interface Display Pages - a detailed look at the iViewer 4 GUI
3. JavaScript Modules - info on the JavaScript modules that Barry uses to provide additional functionality to his system

The Operating Environment

Purpose

The purpose of this system is to control all of the Home Automation attributes of my home and to display their current state at any time.

The House Itself

The house was designed by my late wife, Jill and myself circa 1997. We moved in on my birthday October 1, 1998. The house is a single story open floor plan design with about 3800 sq. ft. of space under air, with an additional 1500 sq. ft. (the garage) not under air. Ceilings in the house are 10 feet, except in the Great Room where they are 12 and 13 feet. The exterior walls are concrete block. We had the builder do a full 2×4 interior inside the concrete block. This makes the walls very thick providing a pleasing architectural look to the window openings. In addition it makes it easy to mount pictures; Jill was a needlepoint artist. The roof is concrete tile.

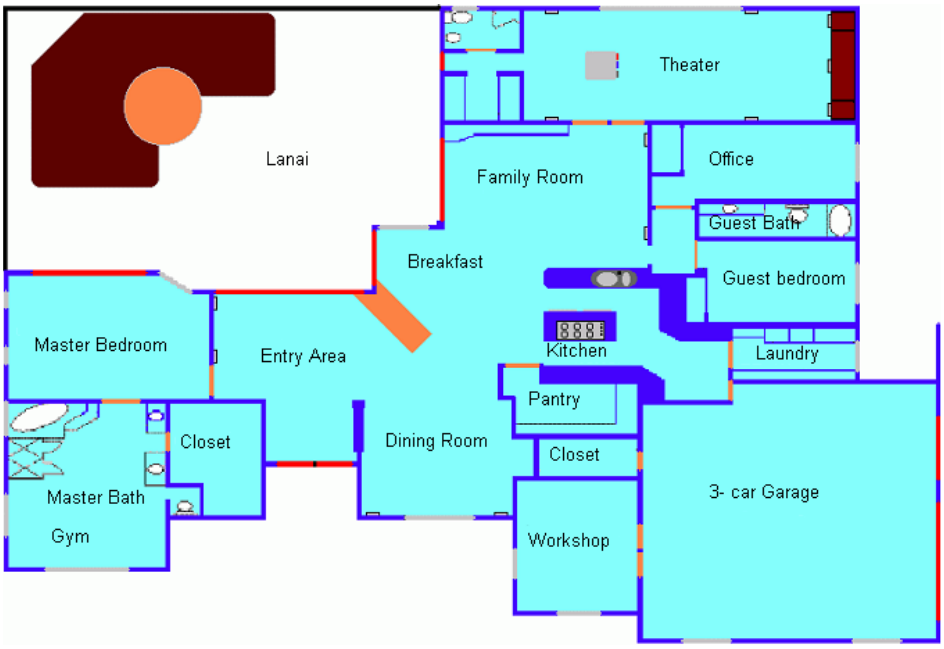
The house is logically divided into zones:

- **The Great Room** - The central area comprised of the Entry way, Kitchen, Dining Room, Breakfast area, and Family area.
- **The Master Suite** - Comprised of the Master Bedroom, Master bath, Walk-in Closet and Gym.
- **The Guest Wing** - Comprised of the Guest Bedroom, Office (an extra bedroom) and the Guest Bath.
- **The Theater** - Comprised of a formal theater, the bar and a small pool bath.
- **The Lanai** - This is the pool area which has a free form pool, a SPA, and covered seating areas.
- **The Utility Zone** - which consists of the Garage, a large air conditioned closet which holds the house electronics and server systems plus seldom used clothing, and a full woodworking shop. The Garage is the only portion of the house not air conditioned.

The electrical system for the house is 300 amp service split to two 150 amp service panels. One panel has a backup generator sized to run that complete panel and propane driven from a 500 gallon underground storage tank which also supplies the gas needs of the house. The system is ready for a second generator, but I have not yet seen the requirement. Special care was taken to ensure a proper system ground (8' x 1" copper pipe driven into the ground at the service entrance). Every box which takes switches, and every ceiling junction box have neutrals to allow for the operation of sophisticated electronic switches (X10, Insteon) without problems.

The view of the house from the street it resides on (a little old, landscaping has changed):

The floor plan (not to scale) for the house roughly aligned with the above picture:



Each zone of the house, except the Lanai and the Utility zone, has a wall mounted iPad. The guest wing has two, one in the guest bedroom and one in the office. There are announcement speakers mounted in the ceilings of all the major rooms.

The main zone, the great room, has three zones of music, each with its own volume control, but all playing the same source. After all, it is just a large open space. The master bedroom has an audiophile quality “stereo only” audio system, while the theater has a full 7.1 audiophile quality sound system to match its 133” HD projection system.

All software (code, data, music, video) is maintained on a 16 TB server which is expandable to 44 TB. Music is served by the “Squeeze Server” system (aka LMS, aka Slim System) while video (movies) are handled by a system I developed specifically for my theater. If I were to do it today I would just use XBMC.

The mobile user interfaces are Philips Pronto Pro remotes for which I developed the code that drives them. The kitchen and master bedroom use 9600's, the guest bedroom a 9400 and the Theater a 9800. They are all coded in Javascript with a significant amount of code reuse. There is an iPad version of these remotes as part of the wall mounted iPads' application. I am in the process of converting the mobile devices to iPod Touch units using most of the code from the iPads.

The wall mounted iPads are actually in the wall using a docking system that is slightly thinner than standard wallboard. This means that iPad placement is independent of stud locations and makes for a very high WAF. The iPads can be reloaded and restarted on command from a central point. They are constantly being charged, and revert to a black screen with the backlight off when not used. They become active on a double tap, or when commanded to display a callerID or similar message by the Home Automation system.

The Home Automation system is “Homeseer” and runs on a dedicated rack mounted 2U server. The rack is in the utility closet which has its own HVAC system so it is never heated in the winter whereas the main living areas might be. The lighting sub-system is being migrated from X10 to Insteon under the control of a Universal Devices ISY994i. The ISY994i is integrated with Homeseer so that all status can be collected at a single point. The house has two zones of Air Conditioning controlled by the ISY994, and Homeseer directly monitors the security system.

The design of the Home Automation system is based upon independently operating subsystems (Security, HVAC, Lighting, etc.) all coordinated by a single overseeing entity - Homeseer. The system will operate completely without Homeseer or the ISY994 or the iPads, just not as conveniently.

All of the electronics (PC based) rides on a single Local Area Network (LAN) with high speed access (60 mb/s) to the Internet. Phone traffic is Internet based (VOIP). The network is 100 mb/sec using only twisted pair (Cat 5e). The systems are very reliable with up times measured in years, not counting planned off times for maintenance and upgrades.

All the PC's in the house run Windows 7 and can be remotely controlled from the PC in my office which is my development system. It is a very fast system that I designed and built from scratch. I do that with most of the PC's in the house. This system has 15,000 rpm SCSI drives, 6 GB of very fast memory and a high speed 4 core processor. The monitor has a screen size of 28" and runs at 1920×1200.

The entire server rack, all of the networking equipment and all of the A/V equipment is powered with a 1.5 KVA backup UPS. The backup generator will start in about 20 seconds after a power outage and powers all of the UPS protected components before the UPS batteries cause them to go offline. Because of the UPS, the switching to and from the generator is seamless to the installed electronics.

The local ISP (cable TV and Internet) feeds into a CATV signal splitter (1:2). One output goes to the TV signal distribution system (1:8) and the other to a Motorola cable modem. The cable modem has its own internal battery backup. All the equipment associated with Ethernet, TV and telephone signal distribution is feed off Panel A, but is not UPS protected. I have plans to change this so the "Head End" signal distribution system is powered off the Utility closet UPS.

The output of the cable modem is fed into a Linksys router which has an internal 4 port Ethernet switch and handles two channels of Voice over IP (VOIP). Only two of the Ethernet ports are used, each of them feeding a 16 port Ethernet switch for a total distribution system of 32 Ethernet lines.

Three lines from one of the Ethernet switches go to POE voltage inserters and then to the three ceiling mounted WAPs. The remaining Ethernet circuits go to wall jacks in the various rooms and the outdoor pool area. Several of the rooms (Utility closet, Master Bedroom, Theater, and the Office) have a second 8 port Ethernet switch for local signal distribution.

The Ethernet wiring is all UTP CAT 5e and runs at 100 mbps. No run is greater than 100 feet, and no path between any two devices consist of more than three hops. There are WAP's located in the Theater, the Master Bedroom and the Great Room ceilings.

The network runs very lightly loaded (2-5%) unless I am playing music or watching a movie. I have not yet had a need to run two movies at the same time as I live alone, however I suspect it will probably work. Download speed from the WWW averages about 60 mbps.

The house also has a cell phone repeater system. The outdoor antenna of the repeater is aimed at the cell tower that services my geographic area on a direct line of sight. This was done because the house construction is of concrete block with a large amount of rebar. This made cell phone reception in the house very poor.

The remainder of this document will concentrate on the system I developed for the wall mounted iPads. The fact they are wall mounted is not germane to the system design.

Home Automation Philosophy

My philosophy regarding home automation has evolved over the years. Things change, new capabilities come about, I change.

First and foremost I want important things e.g. Security, HVAC, Lighting control) to work even when computers fail. Not if, but when. The key characteristic of my HA system is that major subsystems (HVAC, Security, etc) will operate with no computers in the loop. This is also important from a home resale factor. The next owner, and there will be one, may not have the same computer orientation that I do.

The HA system has one major "Brain" and that is the Homeseer HA software product. Information about all subsystems is constantly sent to Homeseer from sub-controllers (ISY994 for Insteon devices, Autelis controller for the Pool/SPA, iPads, etc.).

Dynamic scheduling, i.e. the scheduling of events based upon changing status, or by request from a user via an iPad; is handled by Homeseer, with some simple fixed schedules (e.g. landscape lights on at sunset + 30 minutes) handled by the ISY994.

The iPad is used as the primary user interface to all HA functionality. You can change the state of a light for example; from an iPad, from the ISY994, from Homeseer, and most importantly from the switch that directly controls that light. All lighting switches are computers and they will fail at some time. Such failures are single point in nature, of minor operational consequence, and no different than the light bulb burning out. The systems must be self restarting and self healing to the greatest degree possible. All major computers are setup to automatically restart after a power loss. Power loss to the electronic systems in my home is extremely rare due to the use of UPS and a standby generator. I live in the state of FL in the US. The central region (where I am) of FL is the second most prevalent location for Air-to-Ground lightning strikes in the world. Momentary power outages and voltage fluctuations are the norm.

I have never had to replace a computer or other piece of electronics gear due to a power related event, and there is a lot of electronics in my home, probably way to much.

The System Design - Overview and General Practices

System Design

The system consists of Javascript code modules and display pages. The system uses Javascript extensively and the graphics on the display pages merely make calls to the various Javascript functions where ever possible. The system is highly table driven with most details about the configuration kept in a configuration module (configuration.js) as a Javascript object. When the system starts it reads the configuration object and adjusts all key parameters according to the table entries. If IP addresses for components are not provided and the component is uPnP or DLNA compliant, then it will be searched for and the tables adjusted.

Display Pages

There are several display pages three of which are special. The first page, the splash page, is only seen on a boot of the system while all images are cached into the memory of the iPad, and the configuration tables are processed.

The second special page is the home page and all other pages revert to this page after a period of inactivity. When the home page has been inactive for a configured length of time, it reverts to an all black screen, and after several minutes of that page being displayed the iPad backlight is turned off. This is done since a wall mounted iPad is quite bright even with an all black screen at minimum brightness. Too bright for a bedroom at night IMHO.

The final special page is the Caller ID page. This page is shown for 15 seconds when a phone call is received. It displays the originating phone number and the caller's name. At the same time Homeseer is announcing the call giving an alias name, if available, based upon the originating telephone number.

The remaining pages with the exception of the first page all follow an identical layout. The left and right hand sides contain buttons to move between the various functions. Each function

has a dedicated page whose central area contains the controls and displays for that function.

Javascript Modules

There are many Javascript modules. The modules are all functionally oriented to a specific page and/or functionality. There is a main module that is given control by the CommandFusion (CF) system when all modules have been loaded. Specifically the CF.userMain function is called which resides in the module making it, in effect, the startup module.

Before going into the individual modules and the associated display pages allow me to digress and speak about the coding and Javascript organization style used for the project.

Project Organization and Coding Style

I organize my CF projects into one folder per project all of which are kept on a server in a master folder. The project folder has subfolders called Documentation, Javascript, Resources, and sometimes additional folders for supporting objects. The main project folder contains the .gui module for the project. The CommadFusion architecture wants all resources (Javascript files, images, data files, etc. to be in folders headed by a root folder. By definition the root folder (the root of the directory tree) is the folder that contains the systems GUI file. *[TIP: The GUI file is an XML formatted text file. I have been able to resolve and "fix" several issues by dealing with the gui file directly using a simple text editor. I use TextPad, a free editor with many useful features.]*

I keep my project folders, the root folder and directory tree that make up a project, in a master folder on my web server. This folder is not accessible external to my LAN and is where all my iPads go to load their systems.

Each Javascript module with two exceptions, is coded as a Javascript object. The name of the object can be thought of as the root of the namespace for that module. All things in that module must be referenced (even from within the module itself) as properties of the modules root name. This allows you to reuse names inside different modules without concern for their also existing in another Javascript module. For example each of my modules has an "initialize" function. In the HVAC module it is referred to as HV.initialize since the root name of the object that is the HVAC module is "HV". Similarly the Pool module has a root name of "PL" so its initialize function is referenced as PL.initialize.

Each project has one JS module (I generally name it "Main") that is written in "pure JS" and not as a large object. There is also a second "pure JS" module that holds functions common to all the other modules. This causes things in those modules which are externally available to be named directly without the need of a prefix representing the module itself. The Main module contains the CF.userMain function along with event handlers for page flips, Clock and Timing (called once per second), and preloading completion. The userMain function sets up "watches" for the major events.

The subroutine module contains many common library routines that I have found useful over the years. For example there is a function called uPack which takes a string argument, and returns the same string with all spaces removed and in uppercase. Very useful (and forgiving) when doing text compares. Most library functions are simple one liners, but some are more complex e.g. showHEX which displays any text string as a sequence of hex characters. This is very handy for viewing communications messages; showCOM shows a string with all formatting codes (Carriage return, tab, Line feed, space) explicitly visible using the standard ASCII mnemonics for the control codes. Another key function in the Subroutines module is the Dialog Function which handles popup displays for special notices such as troubles and errors.

All other modules (I call them the Work modules) are written as a single object encapsulating all the functions and data elements used by that module. This means that all functions and data elements defined in that module need to be referenced using the module prefix as the object name. For example, the Autelis Pool Control functions and key variables are contained in an object variable named PL, while the HVAC functionality is contained in an object variable named HV. *[Note: I generally use two or three capital letters to name the object]*. Ergo to access the Autelis module's initialization function one would write PL.initialize(a, b). This technique, which is recommended by the CF team, allows each module to have the same functions with no naming conflicts. Following the modules encompassing object definition a single line of code exists which is used by CF to instantiate the module and call its setup function if required.

The work modules supply supporting code to one or several pages. They have certain functions that I generally provide in every module with similar operational characteristics:

setup:	Called by CF when the module is loaded.
initialize:	Initializes the module. It is normally called when a Page that uses this module is loaded (page flip event).
feedback:	The function that handles feedback from a system that is capable of two way communications. I generally code the feedback regex associated with an external device as (^*) which allows any string of characters and is equivalent I am told to (*) or (.). This presents to the modules feedback function the text received from the device up to that point in time. It should be noted that when dealing with RS232 and interface devices such as Global Cache units or the Xeta server (A single port TCP/IP to RS232 interface) a message may not arrive in a single packet. If the message protocol does not use a terminating sequence then code must be used to ascertain when the full message has been received and to validate its content.
actions:	The function that handles user actions such as button presses.
timeout:	Called when a timer associated with a message transmission times out waiting for a reply (feedback).
reset:	Called when the communications system needs to be reset to start looking for another reply, normally after a timeout or error.
send:	The function to send a command, or possibly resend, a command to an external device. The send function is responsible for enveloping the message in any necessary control characters and starting the transmission timer. In some systems it is possible to use a "sendQueue" to schedule many commands. A command will only be sent after the prior command has returned a reply.

Dealing with External Systems

When dealing with an external system e.g. an iTach, a GC-100xx, A PC handling a specific capability in my home, or the ISY994i; I generally handle the code in a fairly uniform manner.

Class 1:

TCP system I did not write and only handle one connection at a time:

I setup the control system to be always on in the guiDesigner. As soon as I initialize the system I use CF.setSystemProperties to disable the system so it does not attempt to make a connection. I then enable and disable the connection as needed. I enable it when I am ready to send, and disable it after I receive feedback to the transmission.

If the transmission times out waiting for feedback, I reset the system (disable it) and resend the message. I will do this up to 5 times before declaring the system down.

If I am using a queue to drive the system I put items into the queue I want sent then start a send function which transmits the earliest item in the queue (FIFO operation). I pull the next item from the queue when feedback has been received and processed. This provides perfect timing for things such as IR macro sequences if the IR pattern has the proper final off time.

Class 2

TCP systems using a server I wrote that handles multiple connections.

The system is setup as always on in guiDesigner and left enabled when I initialize the system. This causes iViewer to connect to that system and keep trying to do that periodically until successful. When successful we have what is known as an active session since I do not disconnect it. If the external system fails, iViewer will continuously attempt to reconnect.

Transmissions to the system are generally Queue based with an interesting caveat. I optionally include as a prefix in the transmission the name of the feedback function which is to process the reply. The server sends the feedback with this prefix as a prefix in the feedback message. This allows me to decide based on the feedback received, what process will handle the message with almost no effort.

Class 3

TCP systems which I have no Control over.

I just obey the protocol that the system requires.

I know that the CF staff have made recent improvements in the handling of TCP systems and getting feedback. I need to update (simplify) my code to use their efforts.

Debugging

Each module has debugging built in. The debugging aids consist of variables to control the level of debugging and when things will be placed into the CF.log file if debugging has been turned on. The production systems (wall mounted iPads) run with debugging turned off. The highest level debugging variable in each module is named trace. Almost every function has as its first line:

```
if(XX.trace) CF.log("The XX.nameOfFunction function has started with inputs= . . .");
```

This provides a simple flow trace when things do not work correctly after a change.

Activity Log

The system maintains a set of activity log files containing the most recent 50 log entries relating to a specific functionality, e.g. the ISY994. These log entries are not the same as those sent to CF.log for debugging, but document the progress of a specific application functionality as it is running. Significant events are logged as well as warning and error conditions. All asynchronous inputs from either Homeseer or the ISY994 are also logged here. This log can be displayed from a button present on most pages. The log display page allows for the selection of which particular log is to be displayed.

Handy Tools and Aids

Agent Ransack

This a free piece of software that will search all folders of a tree and all files within these folders for a specific string of text. It is very fast. I use it to verify that some Javascript function is not in use any longer so I can remove it and clean up the code.

Textpad

This a free text editor that I use as my standard editor. It understands the difference between PC and Unix line endings. It provides line numbers. It has a list of all the ASCII characters and their decimal indices.

Compare and Merge

A free utility that compares two files flagging all discrepancies with simple commands to correct or remove the discrepancies.

Antechinus Javascript Editor

This is the Javascript editor I use. It is not free but not very expensive either. It has color coding of Javascript types, Pretty printing, Syntax checking and much more. On the advice of others here I am going to look at WebStorm, but old habits die hard.

Next Page

Go to Page 2 - Interface Display Pages