

**AAC Keys for Mac OS X**

# **System Documentation**

## **System Requirements**

Mac OS 10.2. or greater

Usb to serial adaptor or Bluetooth enabled Mac and AAC Device

## **Frameworks and API's**

AAC Keys makes use of the following frameworks:

### **Carbon.framework**

The Carbon framework includes support for all of the standard Aqua user interface elements such as windows, controls, and menus, which you can design using Interface Builder. It also provides an extensive infrastructure for handling events, managing data, and using system resources. The keyboard emulation for AAC Keys is provided by the carbon framework.

### **Cocoa.framework**

The Cocoa application environment consists of two object-oriented frameworks: Foundation (Foundation.framework) and the Application Kit (AppKit.framework).

### **Appkit.framework**

The classes in the Application Kit framework implement the user interface layer of an application, including windows, dialogs, controls, menus, and event handling.

### **Foundation.framework**

The classes in the Foundation framework implement data management, file access, process notification, memory management, network communication, and other low-level features.

### **IOKit.framework**

The framework used for developing device interfaces. This framework was used to detect serial devices for use with AAC Keys.

## **Serial I/O**

To communicate with a serial device from a Mac OS X application, you use the I/O Kit to obtain a path to the device file for that device. The serial port is then accessed using the traditional UNIX serial port access methods provided by the POSIX termios API. The code used to enumerate all the serial devices and access them is derived from the samples provided in the apple developer document "Working with Serial I/O"  
(See /Documents/WorkingWithSerial.pdf on CD)

For further information for programming using the termios API see /Documents/termios serial.pdf on the CD.

## **Keyboard and Mouse Emulation**

The easiest way to implement keyboard and mouse emulation was at the Window Server level. The mechanisms for doing this are provided by **CGRemoteOperation.h** in the **CoreGraphics.framework**, part of the ApplicationServices umbrella. (See /Documents/Keyboard & Mouse Emulation.pdf on CD)

## **Keycode Translation**

The code used for translating ASCII characters in to keycodes was derived from the iGetKeys sample on apples web site. iGetKeys provides a set of keyboard state inquiry routines built on top of GetKeys that provide automatic mapping of ascii codes to their corresponding virtual key codes based on the current keyboard mapping table. On the fly translation of ascii codes to virtual key codes allows applications to use GetKeys in a keyboard layout independent way. (See /SampleCode/iGetKeys on CD)

## **Differences From Windows Version**

- Implemented the *mougo* and *moustop* commands. The mouse direction can be changed at any time by issuing another mougo command (moustop does not have to be issued first). In addition, the full range of commands can be used while the mouse is in movement. The mougo command changes the serial port from interrupt mode to polling mode in order to provide smooth mouse movement.
- Implemented the anchor command (see /Documents/gidei.pdf on CD). Anchors are stored in the users */Library/Preferences* folder as *com.AACKeys.Anchors.plist*.
- Offers error notification and handling via a pop-up error. The PopUp Error box used by AAC Keys is provided by the CustomWindow Class which is a subclass of NSWindow. Any window created using this class will fade away after 3 seconds. To change the duration the error box is displayed change the errorDisplayDuration variable (line 8) in CustomWindow.m.

## **Important Functions used in AAC Keys**

### **ReadText & ReadCommand**

When read text is called it attempts to read ASCII text from the serial port if an escape character is read ReadText calls ReadCommand and the resulting command is then parsed by ParseCommand

### TypeString

This function accepts a c string and types the string using keyboard emulation.

### ParseCommand

Separates the command into tokens the first token is the command being issued and the rest of the tokens are that commands parameters. It then executes the proper code for the command token.

### Keylookup

Accepts a keyname (such as pageup) as a c string and returns the MacOS X keycode for it.

### PressKey

Accepts a keycode and sends a “keyup” and “keydown” event for that key.

## **Linux Port Information**

Because Mac OS X uses standard POSIX termios API, the code for reading from the serial port will also work in Linux. In particular ReadText, ReadCommand, OpenSerialPort, CloseSerialPort, and ParseCommand will all work in linux with little or no modification.

In addition the code to implement the anchor function in linux is located in the LinuxPort Directory on the CD.

## **Changes for future versions of AAC Keys For Max OS X**

- Convert Remaining C functions into Objective-C Methods
- Store the keys in hold array and Lock array as (integer) Keyvalues instead of c strings. That way the keycode is only looked up once.
- Test & implement multilingual capabilities.
- Error Box preferences (display duration, sound only, visual only, customize sound)
- Speed range for mougo adjustable by user via preferences
- Commands called startmacro, endmacro, and runmacro.
  - startmacro marks the beginning of the macro
  - Ex. startmacro dragfolder would put AAC Keys in macro programming mode for the macro called drag folder
  - Commands or text entered inbetween the startmacro and endmacro commands would be recorded to a plist file for use when runmacro is called
  - runmacro – runs the commands stored in a preference (plist) file
  - ex. <esc>, runmacro, dragfolder. Would execute the macro dragfolder.

## **Contents of CD**

AAC Keys Source Code (/AAC\_Keys)

AAC Serial Emulator (windows) (/AAC\_serial)

Documents used for development (/Documents)

Sample code used in the creation of AAC Keys (/Sample Code)

Information and Code for Linux Port (/LinuxPort)