SOFTKISS

TNC-Less Packet for the Macintosh

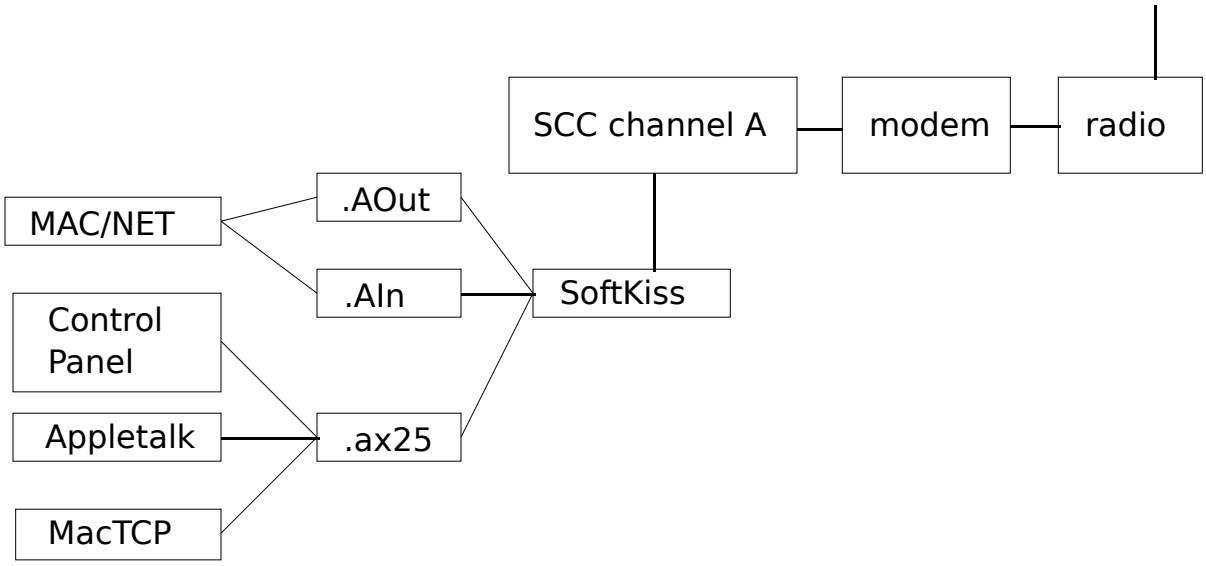
Aaron Wohl, N3LIW
6393 Penn Avenue, #303
Pittsburgh, PA 15206
n3liw+@cmu.edu (Internet)
n3liw (America Online)

ABSTRACT

This paper presents the design and implementation of a software TNC emulator for the Macintosh called SoftKiss. The tools used to build Softkiss are described. These tools will be useful for any Macintosh device driver development. Macintosh TNC less packet is compared to IBM PC TNC less packet.

INTRODUCTION

Softkiss is a family of drivers for the Macintosh. The Mac is connected to a RF modem. Softkiss emulates a TNC in kiss mode. The Mac serial drivers are replaced. Existing Mac programs access the serial port as if a tnc in kiss mode where attached.



COMPARISON WITH IBM PC TNC-LESS PACKET

IBM PC clones use various serial controller chips and line drivers. A typical IBM PC has control signals for at least DTR and carrier detect. However, some implementations such as the Hewlett Packard HP95 have a three wire implementation with no extra signals for transmitter control. The typical output drivers are capable of supplying enough current to power a CMOS modem. Although some systems use charge pumps to develop the RS232 +12 and -12 voltages and can not supply sufficient current to power a modem. The IBM PC serial ports can not directly handle the synchronous HDLC bitstream used in packet radio.

The "Mac" computer may be any one from the original 128K byte Mac to a quadra 950. All of the Macs have a Zilog 85C30 UART connected to high current drivers that can power a CMOS modem. The portable and power book models adjust power to various subsystems such as the serial port and also vary the processor clock speed to conserve power. Currently, when Softkiss is active it leaves the appropriate serial port powered up and disables the slower processor clock and sleep mode.

All of the Macs have high power data out, data in, and HSKo (handshake output, typically called DTR (data terminal ready)). HSKo is used by SoftKiss to key the transmitter. From there the hardware diverges. There are two input pins HSKi (commonly called carrier detect) and GPi (general purpose input). The HSKi is available on all Macs except the Macintosh LC where it is not connected.

The GPi varies considerably from Mac to Mac. It is nominally connected to the DCD input on the SCC chip. However on the Mac+ the mouse horizontal is connected to one SCC channel and the mouse vertical is connected to the other. On the Mac+ this interrupt must be left enabled for the mouse. On some Macs this DCD input on the SCC can be switched to GPi input or to a high speed clock line. The sense of setting this switch varies. The DCD interrupt enable bit in the SCC is write only. Softkiss needs to reset the SCC to set it up to run in HDLC mode. Currently, Softkiss guesses whether to enable the DCD interrupt based on checking the DCD interrupt vector to examine if it points to a return instruction.

On the newer high-end Macs the SCC chip is no longer directly connected to the main processor. An auxiliary IO processor (a Microchip PIC16C5x chip) usually handles serial IO. Apple supplies a "Compatibility Switch" control panel to disable the auxiliary processor. Softkiss examines the "HasSCC" gestalt selector to make a guess as to which mode the processor is in. If the SCC is accessed while the PIC chip is enabled then the system will hang. There is no published interface to tell if the PIC chip is enabled.

Some of the power book processors use an auxiliary processor for power management. When the OS code power management routine talks to the power manager processor it disables interrupts for long periods of time -- long enough to be a problem for MIDI input. On one of the serial channels there is a hook called while interrupts are off to collect incoming data bytes. But this hook is a hack and Apple doesn't recommend using it. Softkiss does not use the hook at this time. This has not been a problem at 1200 baud. However, hams intending to use highspeed packet (above 4800 baud) should beware.

Softkiss needs to remove the existing serial drivers for a port (.AOut and .AIn or .BOut and .Bin) and install itself. When Softkiss is closed it needs to replace the original drivers. This replace/restore is not supported by the OS. There is a remove driver call, but you can't tell which driver to put back (the serial driver may have come from ROM or the system file and it may have been patched to fix bugs at boot time). Softkiss directly manipulates the OS driver table to save and restore the old serial driver.

.AX25 driver

Softkiss installs a new driver .AX25. This driver is used to send control information to Softkiss itself. It will also be used to read and write packets for the interface to the Apple MacTCP and Appletalk.

MACINTOSH DRIVER WRITING TOOLS

dbo_printf - Screen output on the Macintosh must not be done from interrupt level as the drawing and heap management routines in the OS are not reentrant. Setting breakpoints in interrupt level code is problematic. In the past when the author really needed some information from a device driver he would dump the register to the memory mapped screen buffer and read the pixels with a magnifying glass. Fortunately, as part of Softkiss there is a debugging output routine. It writes directly to the screen memory (the screen must be in black and white 2 bit mode). dbo_printf may be called from any interrupt level and is reentrant.

driver_shell - Device drivers must have a unique driver number. Since there are only about 48 driver slots (depending on the OS version) fixed numbers are not issued outside of Apple. According to the Apple documentation drivers are opened with OpenDriver(). This will pick up the driver number stored in the resource fork (program). Attempts to dynamically set this number will trigger many of the virus prevention programs. The driver shell package that comes with Softkiss dynamically finds a free driver slot and installs a driver using the more obscure InstallDriver() call. Driver shell actually installs a stub which calls any desired code. Since Softkiss is reentrant the port A and the port B code share the same main code.

OVERALL CODE COMMENTS

Softkiss started out as a Macintosh application using no interrupts. I recommend this when starting out to access a new device. The THINK C high level debugger can be used to single step through code and examine variables. Interrupts can be added to the application. The core of the Softkiss driver can be linked into a test application for debugging or with the driver shell to make a stand-alone driver.

The source code is available from the author at no charge or from CompuServ hamnet library 9, or America Online in the Ham area.

ABOUT HIGH SPEED PACKET

Softkiss handles each interrupt individually. There is an interrupt when the sync character is detected and another when the first data byte comes in. For high speed, Softkiss should check after handling each interrupt if additional data is available. Appletalk spinloops at interrupt level to receive data at 250 kbits/s. Currently with a loopback cable, Softkiss can talk to itself at 4800 baud on a power book. The author has a pair of Kantronics D4-10 radios and intends to use the bit slicer for 19.2kb without a modem.

HIGHER LEVEL PROTOCOL STACKS

AX25

Currently MAC/NET, a port of KA9Qs net program by Tetherless Access Ltd., is used for the higher level protocol stacks. Jim Van Peursem, KE0PH, is working on an native Mac AX25 stack and terminal program called Savant.

IP/TCP

I have the interface specs to Apple's MacTCP package and will work on the interface glue to Softkiss as time allows.

NATIVE APPLETALK

Native Appletalk support would allow remote access to Macintosh file servers and printing via ham radio. I have the interface specs for adding Appletalk drivers. However, Appletalk uses dynamic node number assignment and it isn't clear how to do node number arbitration with the very dynamic nature of ham usage and marginal paths. Readers interested in helping to solve this (or any other) problem should contact the autho

References

[1] Francis, Dexter, et al "Packet on the Mac". 73 Amateur Radio. October, 1992, pp 8.
[2] Hendricks, Dewayne WA8DZP and Thom, Doug N6OYU. "KA9Q Internet Protocol Package on the Apple Macintosh". Proceedings of the 8th ARRL Computer Networking Conference. Newington CT: ARRL, 1989 pp. 83-91.