Inter-Office Memorandum

To	File	Date	March 13, 1981
From	L. Stewart	Location	Palo Alto
Subject	Alto-1822 Installation and Testing	Organization	CSL

XEROX

Filed on: [Ivy]<Alto-1822>Installation.memo

This document describes installation and testing procedures for the Alto-1822 interface.

Alto-1822 Installation Notes

This is a brief discussion of how to install an Alto-1822 interface in an Alto I or an Alto II. If you intend to control the device from Mesa, you must use an Alto II with either Mesa ROMs or a 3K CRAM.

Alto II

A standard Alto II has nearly all the required signals on slot 18. However, to install an Alto-1822 there, a few wires need to be added. If any of the J18 pins mentioned already have wires, stop and find out why! If slot 18 is already used, slot 17 or 19 will work just as well.

Alto Signal Name	From	To	1822 Signal Name
AUSYSCLK	J14-72	J18-72	AuSysClk
5ACT*	J11-102	J18-118	ITAc’
TASKB*	J10-14	J18-117	TASKB’
WAKE5*	J11-60	J18-112	WakeIT’

In principle, it is possible to wire several devices to TaskA* or TaskB*, and the Alto-1822 drives pin J18-117 with an open collector gate to permit this mode of operation. In practice, it is thought that the 1822 is the *only* device providing this capability, so it is not too useful.

Alto I

An Alto I has much less standard wiring. To install an Alto-1822 in slot 2, the following wires (at least) need to be added.

Alto Signal Name	From	To	1822 Signal Name
AUSYSCLK	J7-72	J2-72	AuSysClk
-TASKB	J6-14	J2-117	TASKB’
-5ACT	J7-102	J2-118	ITAc’
-WAKE5	J7-60	J2-112	WakeIT’
OKTORUN	J2-1	J2-11	OKToRun
-SIO	J3-41	J2-41	SIO’

In addition to the signals mentioned above, the following signals must be present at whatever slot you use:

BUS[00]-[15]
DBARC
F1[0]-[3]
F2[0]-[3]
NEXT[6]-[7]
OKTORUN
RESET

Cabling

The Alto-1822 internal cable is a 40 conductor ribbon cable with a 40 pin PC Card edge connector at the board end and a DB-37S connector at the Alto rear connector panel. Install it in the obvious way! The keyway on the Alto-1822 board occupies wires 33 and 34 of the 40 pin PC edge connector.

An Alto-1822 external cable is somewhat more complicated. Depending on the situation, it should be either a 'Local-Host' cable or a 'Distant-Host' cable. The former are for runs of up to about 30 feet, the latter for runs up to about 1000 feet. Arpa Packet Radio units use Distant-Host wiring exclusively. Refer to 'AICables.memo' and BBN-1822 for additional information.

There is a design bug in the Alto-1822 interface having to do with proper termination of the Distant-Host signals. BBN report 1822 specifies that each pair be terminated at the transmitting end, but the 1822 board as delivered does not provide termination. If distant-host service is required, the fix is to solder-tack 180 ohm 1/4 watt resistors on the bottom surface of the PC board accross the outputs of the 75114 line drivers. Refer to page 6 of the schematic diagrams.

Testing

AITest.run is a general menu-based bcpl test program for the Alto-1822. It includes both hardware test facilities and enough knowledge of protocol to exchange packets with either the Arpanet or the Arpa Packet Radio net.

Attached to this memo is an Alto screen image of the test program. When first started, the program will load Alto-1822 microcode which is linked into the program and execute a silent boot to get the 1822 task running. A polling process is also started which executes a no-op 1822 function and reports the device status in the Status area of the screen. The left portion of the status region contains the following fields:

ICount: The difference between the input control block pointers and the start of the input buffer (number of words input so far).

OCount: Same as ICount, except refers to the output buffer.

IPost, OPost, CPost: The contents of the input, output, and control post locations of the 1822 control block.

The right portion of the status region contains strings which describe the state of the IMP and Host ready relays and the state of the ''IMP was Down'' flipflop. (The *Clear IWD* button may be used to reset the latter indication if the IMP is Up.) The rest of the status region displays strings which report unusual status conditions such as Input Buffer Empty or Buffer Overrun.

The uppermost portion of the AITest.run menu contains functions which will load a packed RAM image microcode file (*Load uCode*), execute a silent boot (*Silent Boot*), and perform a sequence of tests on the hardware (*Test All*). These three functions use parameters which may be altered in the Parameter region of the menu. The *STOP!* button is used to halt the loop tests *Echo*, *CAPEcho* and *Chat* and to stop the *Listen* test if no packets are arriving. *Quit* exits the program.

The Parameter region of the menu contains numerous fields. In general, the way to change a value is to use the left mouse button to select one of the named boxes and then type in the new value, ending with <ESC> or <CR>. *Loop* and *Size* respectively control the number of packets and the packet length in words to be used in the echo tests *Echo* and *Chat*. *Contents* and *Type* offer alternative ways to control the contents of echo packets. *Command* controls the command word sent to the hardware when the *Send Command* button is pushed. Refer to AISWSpec.memo for details of some useful commands. *Boot Vec* sets up the Alto Reset Mode Register value for use with the *Silent Boot* function and the menu item immediately below contains the filename used by the *Load uCode* function. For fun, you can boot from disk by setting *Boot Vec* 177777, and pushing *Silent Boot* -- now no longer silent!

The remainder of the menu items in the Parameter field operate somewhat differently. *Interrupts* uses the three mouse buttons to separately toggle interrupt enables for 1822 Control functions, Input functions, and Output functions. These three enable bits are displayed as ‘‘c’’, ‘‘i’’, and ‘‘o’’. When activated, they print their corresponding letters in the Commentary window whenever the microcode posts status. The *Update* item both enables and disables the status polling process and sets the interval used for polling. The right mouse button toggles the process and the left mouse button sets the interval in 60Hz ticks.

The *Mode* button toggles between Arpanet and Packet Radio modes, causing the appropriate host number to appear in the *Host* window. The *Host* window can then be left-button selected to change the program’s idea of the local host number. The *IMP* field applies only to the Arpanet and should be set to the number of the local IMP.

The Low Level Functions area of the menu generates various useful command functions and sends them to the hardware. *Send Command* transmits the command word from the parameter region, it is rarely needed. *Master Reset* generates a global reset to the 1822 board. *Clear IWD* attempts to reset the hardware IMP Was Down flip flop. It will fail if the IMP still has its ready relay open. *Relay* toggles the 1822 host relay. *Test* toggles the internal board loopback function. *CAP Echo* is really a high level function. It sets up the software to receive packets up to the buffer capability of the program (16,384 bits), and transmit them back out. In essence, *CAP Echo* turns the Alto into an outward facing software loopback plug.

The High Level Functions region of the menu contains other kinds of tests. *Interrupts* really only verifies that the microcode is alive, since the Alto interrupt system is all microcode. It enables the interrupts and then generates 1822 board commands to generate them. (*Test* must be ON, otherwise bogus packets would be sent to the IMP.) *Test BLZ* checks to see whether the 1822 board and microcode act correctly when given a zero length buffer for transmission. *Scatter* and *Gather* test the scatter read and gather write functions of the ‘‘AINcode’’ version of the 1822 microcode. *Listen* is often useful. It simply enables the input logic and waits for a packet to arrive, then prints various things about it in the Commentary window. If no packet arrives, the mode can be exited by pushing the *STOP!* button. *Echo* generates and transmits *Loop* packets of length *Size* and listens for their return. Received packets are checked for equality with the transmit buffer. Correct packets print ‘‘!’’, erroneous packets print ‘‘?’’, and if nothing comes back, a timer process prints occasional ‘‘~’’s and transmits the next packet. This is a one-outstanding-packet test and does NOT use any protocol, so the packets would be very confusing to an IMP. *Check Buffers* compares the first *Size* words of the input and output buffers and prints the number of differing words. *Chat* is somewhat like *Echo* except that it understands Arpanet and Packet radio protocols and generates true echo packets addressed to the IMP or PRU according to the *Mode*. Both *Echo* and *Chat* may be halted with *STOP!*. In both cases, the packet contents may be altered with *Type* and *Contents*.

The final button, *Edit*, brings up a secondary menu which may be used for examining and altering any of the packet buffers in the program. The various buffers are the main input and output buffers, and the packet headers for Arpanet no-op messages and echo messages, and Packet Radio terminal-on-packets and echo packets. The Arpanet protocol logic uses 96 bit leaders, and the Packet Radio protocol is CAP version 5.6. To select and browse a particular buffer, use the left mouse button on, say, *I Buffer*, to bring up the first 8 words of the buffer. The middle and right buttons then step forwards or backwards througth the buffer in 8 word steps. Selecting one of the data items allows that word to be changed. The packet editor will refuse to change anything past *Size*. The *Print* button prints in the commentary the input and output buffers in 8 word chunks, and their exclusive OR, if there are any differences.

If the Edit menu is in view, the edit menu *Quit* button must be pushed to return to the main menu.

Operations

After the 1822 board is installed, pause before conecting any cables and run AITest.run. If *Test All* prints anything ‘‘Bad’’, there is no sense going any further. If there are ‘‘Control Timeout’’ messages coming from the polling process, then either the 1822 board is not plugged in, or the wakeup logic is broken somewhere. Remenber, the right mouse button with turn off the *Update* process. For the echo tests to work without the cable to the IMP, the internal loopback test *Test* must be ON. Try *Loop* 100 and *Size* 120 with *Type* Random.

Once these initial tests work, hook up the cable to the IMP or Packet radio and check out the ready relay logic. *Test* must be OFF to do this. An IMP will flap its ready relay if the host doesn’t accept a packet within 15 seconds or so. The *Listen* test is good to try next. Various Arpanet hosts may send random packets. *Listen* prints some facts about them. Packet Radio Units send Repeater-On-Packets periodically. This test will serve to tell if the input logic works. Of course, be sure that the Arpanet NCC has turned on the IMP port! AITest.run tells how to interpret incoming packets on the basis of the *Mode* switch.

Once confidence has arrived, gingerly set *Loop* 5 or so, and *Size* 20, and push *Chat.* The test program will send three no-ops of the appropriate kind, then start sending echo packets addressed to the IMP or PRU. If they come back, you win! Of course, this test works extremely well if *Test* is ON. Also try a loopback plug at the IMP or PRU end of the 1822 cable. If there is reason to suspect that the hardware works, but the protocols in AITest are obsolete, select *CAP Echo* and ask the appropriate network control center to send you some echo packets.