Macintosh IIci: Causes for IIci Incompatibilities (Part 2 of 4)

Revised: 7/9/92
Security: Everyone

Macintosh IIci: Causes for IIci Incompatibilities (Part 2 of 4)

==========

Article Created: 30 March 1990
Article Last Reviewed: 8 July 1992
Article Last Updated:

TOPIC ----------

This is part two of a four part article detailing the changes which caused the compatibility problems the Macintosh IIci faced. A significant number of Macintosh IIci compatibility problems were related to the improvements outlined below. Keep in mind that the majority of applications were not affected by these changes and that most of those that were have been updated.

DISCUSSION ----------

Leading Causes for Macintosh IIci Incompatibilities (Con't)

2) Utilization of the PMMU - No invalid addresses

Another change related to the utilization of the PMMU in the Macintosh IIci memory addressing architecture is the rejection of invalid addresses. This is perhaps the most prevalent cause for software compatibility problems on the Macintosh IIci. As described above the PMMU plays a critical role in interfacing between logical and physical memory. As part of the process of translating logical to physical, the PMMU also evaluates the "validity" of the logical address requests it receives. An address is considered valid if it falls within the valid address range and the valid address range is determined by how much physical memory is available. A 2MB configuration has a 2MB valid address range and a 5Mb configuration has a 5MB valid address range. This is determined at startup, and the PMMU tables are set accordingly. If an application requests access to an address that lies outside of this range, the PMMU generates a bus error. That all sounds logical. So why should this cause a problem?

Because on previous systems invalid addresses were not identified. When

an application requested access to an address it would go straight to RAM, which would decode whatever bits within the address it could and ignore the rest. For instance if you had 1MB of RAM installed, your valid address range would theoretically include all addresses up to 20 bits wide (2 to the 20th is 1024K), which means that memory would only be capable of decoding up to 20 bits of address information. Now let's say an application requests access to a 21 bit address that is beyond your 1 MB address range. On all systems up to the Macintosh IIci, RAM decodes the lowest 20 bits (ignoring the 21st bit) and grants access to an address within the valid address range. The fact that a bad address had been written was concealed. There is the potential for problems with this model if an application overwrites data stored in a particular address or it reads in bad data, but often the problem is never revealed.

In the case of a Macintosh IIci this errant behavior is always revealed and therefore a number of applications, CDEVs, INITs, and drivers that worked fine up until the Macintosh IIci suddenly broke. Embarrassingly enough, this change revealed problems with two internally developed Apple products, version 3.0 of the CD-ROM driver and version 2.3 of MacTerminal.

The majority of developers who have encountered this problem are aware of it and have either already made the corrections or are in the process of making corrections necessary to make their applications Macintosh IIci compatible. However this problem, which is difficult to diagnose, took many developers by surprise.

IMPORTANT NOTE: It is important to understand that both the move to non-contiguous memory and the utilization of the PMMU to capture invalid addresses presaged changes that were required for compatibility with System 7.0 and virtual memory.

Copyright 1990 Apple Computer, Inc.

Tech Info Library Article Number:5431