Tech Info Library

System 7: 32-Bit Addressing

Revised: 12/20/93
Security: Everyone

System 7: 32-Bit Addressing

==========

Article Created: 7 March 1991
Article Reviewed/Updated: 17 December 1993

TOPIC ----------

1. What is 32-bit addressing?
2. How does it work?
3. Why do I get Type 1 errors?

DISCUSSION ----------

1. What is 32-bit addressing?

32-bit addressing means that you can install and access more than 8MB of physical RAM in your Macintosh. This means you can work with very large data files, very large applications, or many applications concurrently. 32-bit addressing is most attractive to Macintosh users working with large memory-intensive problems. While virtually anyone can benefit from the large amount of memory offered by 32-bit addressing, it will immediately benefit database users, color-graphic users, CAD/CAM users, and programmers. While 32-bit addressing may seem to benefit a small percentage of Macintosh users today, users can expect to soon see powerful "general purpose" tools benefit from 32-bit addressing.

More technically speaking, 32-bit addressing lets most recent Macintosh models access the entire 1GB memory range of the 68030 microprocessor. The basic software and hardware of the Macintosh already support the 32-bit addressing model, but any Macintosh using System 6 is limited to using only 8MB of memory because 32-bit addressing was not yet fully implemented.

In System 7, 32-bit addressing is fully implemented allowing most Macintosh computers access to greater than 8MB of memory. This expanded memory is important for high-end users working with many applications, complex graphics documents, large databases, and so on.

The term "addressing" refers to the number of binary digits (bits) that

make up each memory address. Addressing directly determines the maximum amount of memory possibly available.

2. How does it work?

Binary numbers

To understand what happens with addressing, it is first necessary to understand what address bits are. The number system we're most accustomed to is the decimal system, otherwise known as the Base 10 system. In this system, all the numbers are relative to ten digits (0-9). When a number value exceeds the tenth digit, we add one to the value in the next column and and start over in the first column. Hence, when 1 is added to the number 09 we get 10. When we add 1 to 099, we get 100, so forth and so on.

Base 10 relies on there being exactly ten unique values per digit. Base 10 is not the only number system around. In the early days of computers, it was realized that it is much simpler to build data storage device such as a vacuum tube or a transistor with two possible value states(0 or 1) than it was to build one with 10 possible values(0 through 9). Therefore, someone came up with the idea of defining information in Base 2 format.

With the Base 2 number system, instead of counting your digits 0,1,2,…,9 and then adding a one to the next column, you simply count your digits 0,1 and then add a one to the next column. Hence, in the binary system the number zero in represented by a 0, the number one by a 1 and the number two by a 10, so forth and so on.

Here is a short table of some more examples:

Base 10	Base 2
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

If you expand this table out, you will see, the maximum number of values you can represent with 4 binary digits(bits) is 16, with 16 bits, 65,536, with 24 bits, 16,777,216 and with 32 bits, 4,294,967,296. This is identical to being able to represent 10,000 (0-9999) values with four decimal digits and 100,000,000 (0-99,999,999)values with 8 decimal digits.

Macintosh address range

The original Macintoshes shipped with the 68000 microprocessor. While this CPU is capable of doing 32 bit operations, it has only 24 bit addressing

capability. Which limits a 68000 based computer to 16,777,216 bytes of address range. When the Macintosh II came out with the 68020 which has full 32 bit addressing capabilities, the possible address range was increased to 4,294,967,296 bytes.

Now you're probably wondering why you can only access 8MB in 24 bit mode. The truth is you can access 16mb, but only 8mb is available for user data. The other 8 is used for hardware vectors, NuBus slots, SCSI buffers etc.

The Problem

Since the 68000 is a true 32 bit processor, it stores 32 bits of information for each memory address, but since the 68000 physically only has 24 address lines, only the first 24 bits actually count. This of course means that 8 bits are wasted.

This is where creative programmers come in. Back when the Macintosh only had a 128k of RAM, the Operating System had to go to some extreme lengths to ensure that application have enough memory to run. The Macintosh Memory manager allows blocks of memory to move, and/or be purged if the System is having trouble fulfilling a memory request. The original designers of the Macintosh OS decided to use the last three of the unused bits in a 32 bit memory address to indicate whether a block of memory can move, be purged or if the block contains a resource item.

When System 7 was introduced, the Memory manager portion of the Macintosh Operating system was modified extensively to support full 32 bit addressing. The Memory Manager no longer stores the movable, purgeable, or resource flags in the last three bits of the memory block's address, instead, the Memory Manager stores this information elsewhere. The exact location of these flags is not documented, since an application should not attempt to manipulate these flags directly.

Setting these three magic bits are at the discretion of the programmer. The Macintosh Operating System provides the programmer with the appropriate routines to set these bits. The problem is that to set these three bits, the Operating System routines have to call other routines who have to call still others etc. The net result is that using the Operating System routines to set these bits is quite inefficient when it comes to speed. Therefore, prior the introduction of System 7, some creative programmers with a need for speed, took it upon themselves to set these bits in the memory block's addresses directly thereby bypassing the overhead associated with calling the Operating System routines. Of course, the problem with doing this is that System 7 no longer stores these three bits in the address of the block of memory. Another significant programming error involves the other 5 bits of the 32 bit address. Ordinarily, these bits should remain unused and therefore, insignificant. However, some programmers, having realized that 5 bits are wasted decided to use them for their own purposes, even though Apple Developer Technical Support began warning them against this practice a full three years prior to the introduction of System 7.

3. Why do I get Type 1 errors?

The fundamental problem with setting the upper 8 bits of the address directly is that with System 7 all 32 bits of information are used for addressing. Changing the value of any of the 8 bits changes the address of the block of memory. When an application or an init tries to access the block of memory that now has an invalid address, the usual result is a Type 1 error. This occurs because the first 24 bits of an address are used to access memory locations between 0 and 16mb. The upper eight bits are used to access memory locations between 16mb and 4,096mb. Since most Macintoshes have less than 16mb of RAM, chances are this incorrect memory location is pointing to an address that does not physically exist, and this will yield a Type 1 (Bus Error).

In the event that the memory location does physically exists, then the application or init will then operate on whatever information it finds at the incorrect location. Depending on what the application or init is attempting to do, various errors may result.

Article Change History
---

17 December 1993 - Updated with techshare information from Austin reps
21 August 1993 - Revised - To include information from another article.

Tech Info Library Article Number:6919