

# MEMORY INFORMATION

by Mark E. Donaldson

## Segment and Offset Memory Addressing

In order to understand some of the other logical concepts regarding memory, it's useful to understand how the PC refers to memory addresses. For convenience, it is easier to refer to a memory address using a standard or linear address. So for example we say that the IDE hard disk BIOS usually starts at memory location C8000h. This is in fact true. However, it is not how PC processors refer to memory locations.

In x86 CPUs, **memory addresses are composed of two parts: the segment address and the offset**. These two are added together to produce the "real" address of the memory location, by shifting the segment address one hex digit to the left (which is the same as multiplying it by 16, since memory addresses are expressed in hexadecimal notation) and then adding the segment address to it. The address itself is often referred to using the notation **segment:offset**.

Because of the peculiarity of this scheme, there are in fact many combinations of segments and offsets that can result in the same linear address. Let's take **C8000h** again. The standard way to refer to this address is **C000:8000**. **To get to the linear address you take C000, shift it one digit to the left to get C0000, and then add 8000 to get C8000**. However, **C800:0000 results in the same linear address**.

## Memory Layout Overview

As a result of the design decisions made in the earliest PCS, **memory is broken into the following four basic pieces** (with some of the pieces being divided further):

**Conventional Memory:** The first 640 KB of system memory. This is the area that is available for use by standard DOS programs, along with many drivers, memory-resident programs, and most anything else that has to run under standard DOS. It is **found at addresses 00000h to 9FFFFh**. The name refers to the fact that this is where DOS, and DOS programs, conventionally run. Originally, this was the only place that programs could run. Today, despite much more memory being added to the PC, this 640 KB area remains the most important in many cases. The reason is that without special software support, DOS cannot run programs that are not in this special area.

**Upper Memory Area (UMA):** This is the upper 384 KB of the first megabyte of system memory (immediately above conventional memory). It is reserved for use by system devices and for special uses such as ROM shadowing and drivers. It **uses addresses A0000h to FFFFFh**.

**High Memory Area (HMA):** This is the first 64 KB (less 16 bytes) of the second megabyte of system memory. Technically this is the first 64 KB of extended memory, but it can be accessed when the processor is in real mode, which makes it different from the rest of extended memory. It is usually used for DOS, to allow more conventional memory to be preserved. It **occupies addresses 100000h to 10FFEFh**.

The story of the high memory area is probably one of the strangest. Here we actually have a very useful feature that is the result of a bug! The original IBM PC had only 20 address lines and so its highest memory address was FFFFFh. However, the weird segment:offset addressing makes it

# MEMORY INFORMATION

by Mark E. Donaldson

possible to generate a linear address that is higher than this number.

Take the address FFFF:FFFF. To convert this to a linear address you take the segment and multiply it by 16, to get FFFF0. Then you add the offset, FFFF. FFFF0+FFFF in hexadecimal results in 10FFEFh. There's a problem with this: that "1" at the front requires a 21st address line to represent it, and that doesn't exist on the 8088 or 8086 processors used in the first PCs. They deal with this problem by ignoring the "1". They treat the address simply as 0FFEFh. Software must be able to handle this "wrap around" of the memory addresses.

The 80286 does have a 21st address line (it has 24), and due to a bug in its design it didn't do the wrap around in the same way as the 8088 when in real mode. When it used address FFFF:FFFF and came up with 10FFEFh for a linear address, it kept it as 10FFEFh instead of wrapping it around to 0FFEFh like on the older CPUs. This allowed the first FFEFh of extended memory (100000-10FFEFh) to be accessed by the chip even while still in real mode. This block of memory is the high memory area (HMA).

There was still the problem of ensuring compatibility of the 80286 when in real mode. IBM solved this in the original AT by using spare lines in the keyboard controller chip to manage the 21st address line (which is called the A20 line because address bits are numbered starting with zero). The keyboard controller turns off the A20 line when the processor is running in real mode, to allow for full compatibility with the older PCs. It turns it back on when running in protected mode.

The use of the keyboard controller to operate the A20 line is why you sometimes see an error message relating to the A20 line when you have a keyboard problem with your PC. Microsoft developed a special driver called **HIMEM.SYS** that allowed the A20 line to be manipulated under software control. This allows the high memory area to be put to good use.

The final step in institutionalizing this former bug as an official PC feature was removing manipulation of the A20 line from the keyboard controller. Since that was originally a hack anyway, the controller was used because there was no better way to do it, after all, it has nothing to do with the keyboard. In many newer PCs there is a BIOS option to allow the chipset to control the A20 line directly. This provides a small performance increase compared to letting the keyboard controller manage the line.

**Extended Memory:** This is all the memory above the high memory area until the end of system memory. It is used for programs and data when using an operating system running in **protected mode**, such as any version of Windows. Extended memory is **found from address 10FFF0h to the last address of system memory**. (Technically, the high memory area is part of extended memory, it all depends on how you are looking at things). Extended memory is different from expanded memory (EMS), which uses bank switching and a page frame in the upper memory area to access memory over 1 MB.

A true, full protected mode operating system like Windows NT, can access extended memory directly. However, operating systems or applications that run in real mode, including DOS programs that need access to extended memory, Windows 3.x, and also Windows 95, must coordinate their access to extended memory through the use of an extended memory manager. The most

# MEMORY INFORMATION

by Mark E. Donaldson

commonly used manager is **HIMEM.SYS**, which sets up extended memory according to the extended memory specification (**XMS**). **XMS** is the standard that PC programs use for accessing extended memory. (**HIMEM.SYS** is also used to enable access to the high memory area, which is part of extended memory). In practical terms, extended memory is usually referred to as **XMS** and vice-versa, even though technically **XMS** is really a protocol for using extended memory.

## Processor Modes

Processor modes refer to the various ways that the processor creates an operating environment for itself. Specifically, the *processor mode controls how the processor sees and manages the system memory* and the tasks that use it. There are *three different modes* of operation, that resulted from the evolution of the PC.

### Real Mode

The original IBM PC could only address 1 MB of system memory, and the original versions of DOS created to work on it were designed with this in mind. DOS is by its nature a single-tasking operating system, meaning it can only handle one program running at a time. The decisions made in these early days have carried forward until now, and in each new processor, care had to be taken to be able to put the processor in a mode that would be compatible with the original Intel 8088 chip. This is called real mode.

Real mode is of course used by DOS and "standard" DOS applications. In fact, today there are relatively few simple DOS programs that just use the standard 640K that DOS makes available. Even within DOS now there are special programs available that will "extend" DOS to allow access to extended memory (over 1 MB) and faster 32-bit access. These are sometimes called DOS extenders. The protocol that describes how to make DOS work in protected mode is called DPPI (DOS protected mode interface). DOS extenders are used by most DOS games.

### Protected Mode

Starting with the 80286 chip in the IBM AT, a new processor mode was introduced called protected mode. This is a much more powerful mode of operation than real mode, and is used in all modern multitasking operating systems. The advantages of protected mode (compared to real mode) are:

- Full access to all of the system's memory; there is no 1 MB limit in protected mode.
- Ability to multitask, meaning having the operating system manage the execution of multiple programs simultaneously.
- Support for virtual memory, which allows the system to use the hard disk to emulate additional system memory when needed.
- Faster (32-bit) access to memory, and faster 32-bit drivers to do I/O transfers.

The name of this mode comes from its primary use, which is by multitasking operating systems. Each program that is running has its own assigned memory locations, which are protected from conflict with other programs. If a program tries to use a memory address that it isn't allowed to, a

# MEMORY INFORMATION

by Mark E. Donaldson

"protection fault" is generated. DOS, which normally runs in real mode, can access protected mode using DPMI (DOS protected mode interface), which is often used by DOS games to break the 640 KB DOS conventional memory barrier.

All processors from the 286 on can use protected mode. 386 and later processors can switch on the fly from real to protected mode and vice-versa; the 286 can only switch from real to protected mode once (switching back requires a reboot). ***Protected mode is also sometimes called 386 Enhanced Mode***, since it became mainstream with that family of processors.

## Virtual Real Mode

The third mode of processor operation is actually an additional capability, an enhancement, of protected mode. In essence, it emulates real mode from within protected mode, allowing DOS programs to run. A protected mode operating system such as Windows can in fact create multiple virtual real mode machines, each of which appear to the software running them as if they are the only software running on the machine. Each virtual machine gets its own 1 MB address space, an image of the real hardware BIOS routines, everything.

Virtual real mode is what is used when you use a DOS box or run a DOS game in Windows 95. When you start a DOS application, Windows 95 creates a virtual DOS machine for it to run under. Virtual real mode was introduced starting with the 386 family of processors.

## Memory Addresses and Device BIOSes

Some devices, in addition to using interrupt lines, DMA channels and/or I/O addresses, require some space in the upper memory area for their own use. The devices that use a memory area generally use it for their own BIOS, which contains code to control the device and is invoked by direct calls or calls from the internal system BIOS. These BIOSes are "mapped" into the upper memory area in particular places and the BIOS looks for them there and executes them if found. This is part of the system boot process.

There are **three standard BIOSes present in most systems and located pretty much at the same place:**

- **System BIOS:** The main system BIOS is located in a 64KB block of memory from **F0000h to FFFFFh**.
- **VGA Video BIOS:** This is the BIOS that controls your video card. It is normally in the 32KB block from **C0000h to C7FFFh**.
- **IDE Hard Disk BIOS:** The BIOS that controls your IDE hard disk located from **C8000h to CBFFFh**.

The most common add-in device to use a dedicated memory address space for its own BIOS is a **SCSI host adapter**. This **may default to C8000-CBFFFh**, which will conflict with an IDE drive that is also in the system, but **can be configured to use a different address space instead**, such as **D0000-D7FFFh**. In addition, network cards that have the ability to boot the computer over the

# MEMORY INFORMATION

by Mark E. Donaldson

network typically also use a memory area for the boot BIOS.

## Input / Output (I/O) Addresses

Input/output addresses (usually called I/O addresses for short) are resources used by virtually every device in the computer. They **represent locations in memory that are designated for use by various devices to exchange information between themselves and the rest of the PC.**

## Memory-Mapped I/O

You can think of I/O addresses like a bunch of small two-way "mailboxes" in the system's memory. Take for example a communications (COM) port that has a modem connected to it. When information is received by the modem, it needs to get this information into the PC. Where does it put the data it pulls off the phone line?

## I/O Address Space Width

Unlike IRQs and DMA channels, which are of uniform size and normally assigned one per device, sound cards use more than one because they are really many devices wrapped into one package, **I/O addresses vary in size.** The reason is simple: some devices (e.g., network cards) have much more information to move around than others (e.g., keyboards).

The size of the I/O address is also in some cases dictated by the design of the card and (as usual) compatibility reasons with older devices. Most **devices use an I/O address space of 4, 8 or 16 bytes. Some use as few as 1 byte and others as many as 32 or more.** The wide variance in the size of the I/O addresses can make it **difficult to determine and resolve resource conflicts, because often I/O addresses are referred to only by the first byte of the I/O address.**

For example, people may say to "put your network card at 360h", which may seem not to conflict with your LPT1 parallel port at address 378h. In fact many **network cards** take up 32 bytes for I/O; this means they use up 360-37Fh, which totally overlaps with the parallel port (378-37Fh). The I/O address summary map helps you to see which I/O addresses are most used, and to visualize and avoid potential conflicts.

One additional note about parallel ports. The I/O addresses used for the different parallel ports (LPT1, LPT2, LPT3) are not universal. Originally IBM defined different defaults for monochrome-based PCS and for color PCS. Of course, all new systems have been color for many years, but even some new systems still default LPT1 to 3BCh. Here is how the two different labeling schemes typically work:

Port	Monochrome Systems	Color Systems
LPT1	3BC-3BFh	378-37Fh
LPT2	378-37Fh	278-27Fh
LPT3	278-27Fh	

## I/O Address Details By Number

- **060h and 064h:** These two addresses are used by the keyboard controller, which operates

# MEMORY INFORMATION

by Mark E. Donaldson

both the keyboard and the PS/2 style mouse (on devices that use it).

- **130-14Fh, 140-15Fh:** These addresses are sometimes offered as options for SCSI host adapters. Note that these options partially overlap (from 140-14Fh).
- **220-22Fh:** This is the default address for many sound cards. It is also an option for some SCSI host adapters (first 16 bytes).
- **240-24Fh:** This is an optional address for sound cards and network cards (first 16 bytes for NE2000 cards).
- **260-26Fh and 270-27Fh:** This is an optional address for sound cards and network cards. NE2000-compatible network cards take 32 bytes. If set to use this I/O address, they will conflict with several system devices as well as the I/O address for either LPT2 or LPT3 in the 270-27Fh area.
- **280-28Fh:** This is an optional address for sound cards and network cards (first 16 bytes for NE2000 cards).
- **300-30Fh:** This is the default for many network cards (NE2000 cards extend to 31Fh). 300-301h is also an option for the MIDI port on many sound cards.
- **320-32Fh and 330-33Fh:** This is a busy area in the I/O memory map. First, 330-331h is the default for the MIDI port on many sound cards. 320-33Fh is an option for some NE2000-compatible network cards and will conflict with the MIDI port at this setting. Some SCSI host adapters also offer 330-34Fh as an option. Finally, the old PC/XT hard disk controller also uses 320-323h.
- **340-34Fh:** Optional areas for several device types overlap here, including two options for SCSI host adapters (330-34Fh and 340-35Fh) as well as network cards.
- **360-36Fh and 370-37Fh:** This is another "high traffic" area.
- **378-37Fh:** This is used on most systems for the first parallel port, and 376-377h is used for the secondary IDE controller's slave drive. These can conflict with an NE2000-compatible network card placed at location 360h. Tape accelerator cards often default to 370h, which will also conflict with a network card placed at 360h).
- **3B0-3BBh and 3C0-3DFh:** These are used by VGA video adapters. They take all of the areas originally assigned for monochrome cards (3B0-3BBh), CGA adapters (3D0-3DFh) and EGA adapters (3C0-3CFh).
- **3E8-3EFh:** There is a potential conflict here in locations 3EE-3EFh if you are using a third serial port (COM3) and a tertiary IDE controller.

# MEMORY INFORMATION

by Mark E. Donaldson

- **3F0-3F7h:** There is actually a "standard" resource conflict here: the floppy disk controller and the slave drive on the primary IDE controller "share" locations 3F6-3F7h. These devices are actually both present in many systems. Fortunately, this conflict (which exists for historical reasons) is fairly well known and compensated for, so it will not result in problems in a typical system. Note that some tape accelerator cards also offer the use of 3F0h as an option, which will conflict with the floppy disk controller.

## I/O Address Summary Map

The **table below shows the I/O addresses from 000 to 3FFh, along with the devices that typically use them.** Each row is 16 bytes and is divided into four columns; the first is for bytes 0 to 3, the second 4 to 7, the third 8 to B and the fourth C to F. So to find address 3BCh, you would look in the fourth column of row "3B0-3BFh".

Items in the table in bold print represent standard devices in a typical PC configuration. Items in regular print represent optional devices or optional locations for addresses of standard devices. Blank spaces are areas that are open. Multiple lines are used to show multiple items that go in the same address space. Where you see two or more items overlapping in the same address space, there is the potential for a resource conflict.