Élan™SC300 and ÉlanSC310 Microcontrollers Memory Management

AMD

Application Note

The Élan™SC300 and ÉlanSC310 microcontrollers contain a sophisticated memory management unit (MMU), which makes PC/AT compatibility easy to achieve. However, if you wish to stray from the standard path and do something different, there is much to be learned concerning the capabilities and limitations of the MMU. The purpose of this application note is to explain what the MMU is capable of and how it may be programmed to achieve your goals. This document supplements Chapter 2, Memory and PCMCIA Management *of the* Élan™SC300 Microcontroller Programmer's Reference Manual*, order #18470.*

INTRODUCTION

Unless otherwise noted, the features discussed in this document for the Élan™SC300 microcontroller apply equally to the Élan™SC310 microcontroller. The term "DRAM" will be used to mean either DRAM or SRAM. SRAM for system memory is an option on the ÉlanSC300 microcontroller and for the purposes of memory management, SRAM is treated identically to DRAM. The term "Local/ISA Bus" includes both Local bus and ISA bus cycles. These are treated identically for memory management. When the ÉlanSC300 microcontroller is configured in full ISA mode or in internal LCD mode, all accesses are treated as ISA bus cycles. When the ÉlanSC300 microcontroller is configured in local bus mode, the local bus is first given the cycle, and if a local bus peripheral accepts the cycle, the ISA bus never sees it. If no local bus peripheral accepts the cycle, it is stretched into a slower ISA cycle, and $\overline{\text{MEMR}}$ or $\overline{\text{MEMW}}$ is asserted.

ADDRESS DECODING AND ALIASING

Most designers are familiar with address aliasing, which simply means that if an address is only partially decoded by a device, that device will appear to exist multiple times throughout the address space. The following topics that are associated with aliasing on the ÉlanSC300 microcontroller must be thoroughly understood by the designer before attempting to use the ÉlanSC300 microcontroller memory management.

Aliasing within the 4-Gbyte Address Space of the Am386® Microprocessor

The core CPU of the ÉlanSC300 microcontroller is an Am386SX microprocessor, which supports 24 physical address lines (16 Mbyte of physical address space). The Am386 microprocessor architecture supports 32 bits of addressing, but the top 8 address bits from the CPU are effectively ignored for all memory mapping functions. From a programming perspective, this means that the ÉlanSC300 microcontroller can "see" 16 Mbyte of memory at a time. This 16 Mbyte of memory is aliased 256 times into the 4-Gbyte physical address space of the Am386 processor. Although it is true that the PCMCIA address spaces of the ÉlanSC300 microcontroller support 26 bits (64 Mbyte), these PCMCIA spaces are only accessible by translated memory management.

Behavior at Reset and SMI

After reset or when processing a system management interrupt (SMI), the CPU executes from the top of memory. The processor is in Real mode and normally can address only 1 Mbyte, but the first instruction is fetched from address FFFFF0 at the top of memory. The initial value of the code segment, CS, is F000, and the initial value of the instruction pointer, IP, is FFF0. However, the internal CPU base address associated with the code segment is FF0000 rather than F0000. The processor can remain in this top 64 Kbytes of the available system memory address space by making near calls/jumps. If a far call/jump is executed and the CPU is still in Real mode at the time of this far control transfer, the far control transfer will cause the CS base address to be set to 16 times the segment of the jump target. This is the normal Real mode behavior. Thus, the target of such a far jump must be in the lower 1 Mbyte. Many PC/AT-compatible BIOS implementations have a far jump to a target in segment F000 as the first instruction executed.

The fact that the physical address of the code executed at reset is nowhere near the physical address of the code executed at the target of the first Real mode far jump can be important for systems with boot ROMs larger than 1 Mbyte. This is discussed in the section entitled $\overline{\text{ROMCS}}$ Space and Non-Translated Memory Management.

Note that the behavior described previously applies to SMI handling as well as to reset. In both cases, you should ensure that the target of any far jump is mapped appropriately. For example, at FFFFF0, a system may

Publication# **21823** Rev: **A** Amendment/**0**
Issue Date: **July 1997**

contain a far jump to address F0000, which maps to $\overline{\text{ROMCS}}$ at reset. After booting, the processor may disable $\overline{\text{ROMCS}}$ for F0000–FFFFF and map that region to DRAM. If an SMI occurs, the jump at FFFFF0 will be executed and will now jump to F0000 in DRAM. You must ensure that the appropriate code for SMI handling remains at that location.

Special Handling for A20

Because the ÉlanSC300 microcontroller is designed to be PC/AT compatible, it contains logic to allow backward compatibility all the way to the 8088 processor. One of the 8088 features used by some software is the address wrap, which occurs from the top of the 8088 processor's 1-Mbyte memory down to the bottom of memory. The Am386 processor performs this by using an A20 control gate, which can force A20 to always be 0. Because the Am386 processor defaults to 8088-compatible mode, you must set the gate to allow A20 propagation for most applications. Most operating systems, such as MS DOS, contain code to do this. In the case of MS DOS, HIMEM.SYS contains the code and has an application program interface (API) to allow program control of A20. A20 automatically will be set to propagate if HIMEM.SYS is loaded and DOS=HIGH{,UMB} is added to CONFIG.SYS. For information about direct control over the A20 gate, look at the descriptions for the ÉlanSC300 microcontroller direct-mapped register 92h and index registers 6Bh and 6Fh in the *Élan™SC300 Programmer's Reference Manual*, order #18470. Also, refer to the application note entitled *Élan™SC300 and Élan™SC310 Microcontroller GATEA20 Function Clarification*, order #21811.

MULTIPLE MEMORY SPACES

One concept which may be foreign to some designers is that of multiple, parallel memory spaces, as opposed to a single linear space. The ÉlanSC300 microcontroller may address up to four distinct 16-Mbyte memory spaces: DRAM, $\overline{\text{ROMCS}}$, $\overline{\text{DOSCS}}$, and the Local/ISA bus. The ÉlanSC300 microcontroller additionally supports up to four distinct 64-Mbyte PCMCIA memory spaces (there are two PCMCIA sockets and each socket contains both data and attribute memory spaces). The ÉlanSC300 microcontroller, when configured in internal video mode, contains a small 32-Kbyte Display SRAM space.

The ÉlanSC300 microcontroller supports a few different specialized memory management schemes (most based on some form of backward compatibility), but they can be placed into two classes, which will be referred to as translated and non-translated memory management. In the non-translated scheme, the MMU hardware simply selects one of the distinct memory spaces for the read or write, and the memory address is passed unchanged to the hardware controlling that space. In the translated scheme, the MMU will translate the address in addition to choosing the space.

The DRAM, $\overline{\text{ROMCS}}$, $\overline{\text{DOSCS}}$, Display SRAM, and Local/ISA bus spaces are each accessible, to one degree or another, via non-translated memory management. All spaces except the Local/ISA bus space are fully accessible via use of translated memory management.

NON-TRANSLATED MEMORY MANAGEMENT

As noted in the previous section, when performing non-translated memory management, the MMU simply decides which device space will receive a given memory cycle by examining the memory address provided by the CPU. After examination, however, the address is passed unchanged to the hardware managing that space.

Memory Mapping at Reset

For most of the CPU address space, the default memory space on reset is the Local/ISA bus. The exceptions are:

- CPU address ranges 0F0000–0FFFFF and FF0000–FFFFFF are mapped to $\overline{\text{ROMCS}}$.
- Address range B8000–BFFFF maps to the 32-Kbyte Display SRAM space. This mapping occurs even if the ÉlanSC300 microcontroller boots in local bus or full ISA mode. The internal LCD controller must be disabled in order to direct these accesses to the Local/ISA bus.

Note: *The ÉlanSC310 microcontroller does not support an internal LCD controller, but a similar disabling must be performed using the bus configuration registers.*

$\overline{\text{ROMCS}}$ Space and Non-Translated Memory Management

At reset, the CPU address ranges 0F0000–0FFFFF and FF0000–FFFFFF are mapped to $\overline{\text{ROMCS}}$. This default mapping makes it easy to handle an early far jump from the reset start address at FFFFF0 to segment F000 in Real mode. This is the normal behavior for a PC/AT-compatible BIOS. (See the description of the processor behavior at reset in the previous section entitled Behavior at Reset and SMI.)

Because this is non-translated memory management, the MMU simply examines the CPU address, chooses the $\overline{\text{ROMCS}}$ space, and passes the address untranslated to the address lines. This means that the two default ranges, 0F0000–0FFFFF and FF0000–FFFFFF, will only map to identical addresses for designs that decode fewer than 20 address lines in $\overline{\text{ROMCS}}$ space (smaller than 1 Mbyte). You must be aware if the boot ROM is larger than 1 Mbyte. For

example, in a 2-Mbyte boot ROM, the top 16 bytes of the boot ROM may have to be reserved for a far jump to a location in segment F000, which will reside immediately below 1 Mbyte in the same boot ROM.

When the code has booted, you have some control over which CPU addresses cause an access to $\overline{\text{ROMCS}}$. Using the ÉlanSC300 microcontroller index registers 51h and 65h, the CPU can map to $\overline{\text{ROMCS}}$ address space in five different 64-Kbyte regions: A0000–AFFFF, C0000–CFFFF, D0000–DFFFF, E0000–EFFFF, and F0000–FFFFF. The 64-Kbyte region just below 16 Mbyte, FF0000–FFFFFF, always maps to $\overline{\text{ROMCS}}$. These are the only regions that may be mapped to $\overline{\text{ROMCS}}$ without using translated memory management.

DRAM and Non-Translated Memory Management

Before any DRAM can be used, the boot code must program the DRAM controller (refer to the tables in the Configuration Registers section of the *ÉlanSC300 Microcontroller Programmer's Reference Manual*, order #18470). Programming the controller sets the total size of the DRAM space. DRAM-space sizes from 512 Kbyte to 16 Mbyte are supported. The DRAM controller logically concatenates all the system DRAM into a single unified address space that starts at address 0. By default, CPU addresses from 0 to the top of DRAM will be mapped to the DRAM space, in preference to Local/ISA space. There are two exceptions:

- The window from 640 Kbyte to 1 Mbyte (A0000–FFFFF) is not automatically mapped to DRAM. Mapping in this region A0000–FFFFF remains what it was before the DRAM controller was enabled. (A description of the address range A0000–FFFFF is described in the Address Range A0000–FFFFF section below.) It is possible to map addresses in the range C0000–FFFFF to DRAM, also described below. It is not possible to directly map addresses in the range A0000–BFFFF to DRAM.
- The 64-Kbyte window at the top of CPU address space (FF0000–FFFFFF) is always mapped to $\overline{\text{ROMCS}}$.

The ÉlanSC300 microcontroller offers a limited amount of programming control over DRAM using non-translated memory management:

- In the region C0000–FFFFF, any 16-Kbyte region that is mapped to Local/ISA bus (that is, **not** mapped to $\overline{\text{ROMCS}}$) can be redirected to DRAM by setting bits in index registers 65h, 68h, and 69h. This feature is sometimes called ROM shadowing, because it allows shadowing of slow ROM into faster DRAM. A flag in index register 65h allows this shadowed memory to be write protected. Note that the granularity for these shadowing regions is 16 Kbyte, whereas the granularity for $\overline{\text{ROMCS}}$ mapping is 64 Kbyte. Thus, you can have a 64-Kbyte region that is a mixture of Local/ISA bus and DRAM on 16-Kbyte boundaries, but you cannot have a 64-Kbyte region that is a mixture of $\overline{\text{ROMCS}}$ and DRAM.

After the DRAM size has been set by programming the DRAM controller, any amount from 1 Mbyte to 15 Mbyte may be subtracted from the DRAM size using index register 6Fh. The DRAM start address remains at 0 and the end address is reduced. It is even possible to subtract the entire DRAM space using this method. Accesses beyond the new DRAM end address revert to Local/ISA bus space. However, the portions above 1 Mbyte can be redirected to $\overline{\text{DOSCS}}$ space, as discussed in the following section.

$\overline{\text{DOSCS}}$ Space and Non-Translated Memory Management

The $\overline{\text{DOSCS}}$ space is a 16-Mbyte space. No part of $\overline{\text{DOSCS}}$ space is automatically mapped at reset. For non-translated memory management (not using the MMS), one can program index register B8h to map any amount from 1 Mbyte minus 64 Kbyte up to 15 Mbyte minus 64 Kbyte of $\overline{\text{DOSCS}}$ space in 1-Mbyte increments. This non-translated $\overline{\text{DOSCS}}$ space always ends at address FEFFFF. As an example, if 3 Mbyte minus 64 Kbyte are mapped, the addresses D00000–FEFFFF will be mapped to $\overline{\text{DOSCS}}$. Remember, the region FF0000–FFFFFF is always mapped to $\overline{\text{ROMCS}}$, which accounts for the minus 64 Kbyte.

In all cases, the $\overline{\text{DOSCS}}$ mapping is not allowed to overlap with a range that is mapped to DRAM. Index register 6Fh can be used to limit the size of DRAM space if necessary. This DRAM limiting must be done before $\overline{\text{DOSCS}}$ mapping is enabled.

Again, because this is non-translated memory management, the MMU simply examines the CPU address, chooses the $\overline{\text{DOSCS}}$ space, and passes the untranslated address to the address lines.

Address Range A0000–FFFFF

The address range A0000–FFFFF is handled differently for PC/AT compatibility. This section summarizes the different ways A0000–FFFFF can be mapped.

At reset, the mapping for A0000–FFFFF is:

A0000–B7FFF:	Local/ISA
B8000–BFFFF:	Display SRAM
C0000–EFFFF:	Local/ISA
F0000–FFFFF:	$\overline{\text{ROMCS}}$

The following should also be noted:

- The 32-Kbyte Display SRAM region, B8000–BFFFF, can revert to Local/ISA by disabling the internal video controller, or it can be moved to B0000–B7FFF by setting the internal video controller to HGA mode.
- When DRAM is enabled, the range A0000–FFFFF is not automatically mapped to DRAM. Its mapping remains at what it was before DRAM was enabled.
- With 64-Kbyte granularity, blocks in regions A0000–AFFFF and C0000–FFFFF can be mapped to $\overline{\text{ROMCS}}$.
- With 16-Kbyte granularity, blocks in the region C0000–FFFFF that are not already mapped to $\overline{\text{ROMCS}}$ can be mapped to DRAM.
- With 16-Kbyte granularity, the four blocks in the range A0000–AFFFF and up to eight consecutive blocks in the range C0000–F3FFF can be mapped using MMSB and MMSA, which use translated memory management. This translated memory management allows each 16-Kbyte region to be mapped to any 16-Kbyte boundary in any address space except Local/ISA bus. This is discussed further in the next section.

TRANSLATED MEMORY MANAGEMENT

When the ÉlanSC300 microcontroller performs translated memory management, it translates addresses in addition to selecting the correct address space for each access. A window in the CPU address space is mapped to a particular target location in the target address space.

The ÉlanSC300 microcontroller implements translated memory management via two MMS windows: MMSA and MMSB. MMSA consists of eight consecutive 16-Kbyte regions starting at any 16-Kbyte boundary from C0000–D4000. MMSB consists of four consecutive 16-Kbyte regions starting at A0000. Each 16-Kbyte region can be mapped to any 16-Kbyte boundary in any address space except Local/ISA bus. This allows mapping to any address in DRAM, $\overline{\text{ROMCS}}$, or $\overline{\text{DOSCS}}$. On the ÉlanSC300 microcontroller only, you can also map to the four PCMCIA memory spaces. Note that MMSA and MMSB are the only means of accessing the four PCMCIA memory spaces. They cannot be accessed via non-translated memory management.

The particulars of programming MMSA and MMSB are described in chapter 2 of the *ÉlanSC300 Microcontroller Programmer's Reference Manual*, order #18470. The following should be noted:

- All of the 16-Kbyte pages in either MMSA or MMSB can be enabled or disabled by a global switch for each MMS system. In addition, each individual 16-Kbyte page within MMSA or MMSB can be enabled or disabled. When a 16-Kbyte page is disabled, addresses in that range map to whatever they would map to with non-translated memory management. For example, if MMSA starts at C0000, and page 4 (D0000–D3FFF) is disabled, then D0000–D3FFF will go to either Local/ISA bus, $\overline{\text{ROMCS}}$, or to DRAM using the rules discussed in the previous section entitled Non-Translated Memory Management.

A page can be enabled in MMSA or MMSB only if the non-translated mapping for that page is Local/ISA bus. A page cannot have a non-translated mapping to DRAM or $\overline{\text{ROMCS}}$ space. For example, if F0000–FFFFF is enabled for $\overline{\text{ROMCS}}$ mapping, and MMSA is set to start at D4000, you cannot enable page 7 of MMSA (F0000–F3FFF) because F0000–F3FFF is already mapped to $\overline{\text{ROMCS}}$. Note, however, that if F0000–FFFFF is disabled for $\overline{\text{ROMCS}}$ mapping, you could enable page 7 of MMSA and still have F4000–FFFFF mapped to DRAM. This is possible because DRAM mapping in this region can be enabled or disabled on 16-Kbyte boundaries.

Translated Memory Management in System Management Mode

System Management Mode (SMM) on the ÉlanSC300 microcontroller involves the following mappings.

- When SMM is entered via an SMI, CPU addresses 60000–63FFF map to a translated DRAM location on a 16-Kbyte boundary. The system state is saved at the bottom of this area before the SMI execution starts. The address that 60000 maps to in DRAM space is programmable using index registers A9h and AAh. This region, sometimes called SMM RAM, can thus be mapped to a DRAM location such as AC000–AFFFF, which is normally unused because AC000–AFFFF cannot be mapped to DRAM in normal (non-SMM) system operation. This allows SMM RAM to be invisible to normal system operation.
- Except for the region 60000–63FFF, the remainder of the CPU address mappings is identical to what it was before the SMI. Remember that the SMI handler execution begins at FFFFF0 and that FFFFF0 always maps to $\overline{\text{ROMCS}}$ (it is entirely independent of the mapping of the 0F0000 block in the lower 1 Mbyte). Also, remember that the 0F0000 block will not necessarily point to $\overline{\text{ROMCS}}$ if it has been reprogrammed since processor reset.

REFERENCE MATERIAL

- *Élan™SC300 Microcontroller Data Sheet*, order #18514
- *Élan™SC310 Microcontroller Data Sheet*, order #20668
- *Élan™SC300 Programmer's Reference Manual*, order #18470
- *Élan™SC310 Programmer's Reference Manual,* order #20665
- *Élan™SC300 and ÉlanSC310 Microcontrollers GATEA20 Function Clarification*, order #21811