# Using 16-Bit $\overline{\mathrm{ROMCS}}$ Designs in Élan™SC300 and ÉlanSC310 Microcontrollers

AMD

## *Application Note*


*This application note describes how to assert $\overline{\mathrm{MCS16}}$ for $\overline{\mathrm{ROMCS}}$ accesses and $\overline{\mathrm{IOCS16}}$ for I/O accesses without performing an external address decode.*


## $\overline{\mathrm{MCS16}}$ AND $\overline{\mathrm{IOCS16}}$ SIGNAL DEFINITIONS

$\overline{\mathrm{MCS16}}$ (memory size 16) is generated by a 16-bit memory expansion card when the card recognizes it is being addressed. This signal tells the data bus steering logic that the addressed memory device is capable of communicating over both data paths. When accessing an 8-bit memory device, the $\overline{\mathrm{MCS16}}$ line remains deasserted, indicating to the data bus steering logic that the currently addressed device is an 8-bit memory device capable of communicating only over the lower data path.

$\overline{\mathrm{IOCS16}}$ (I/O size 16) is generated by a 16-bit ISA I/O expansion board when the board recognizes it is being addressed. $\overline{\mathrm{IOCS16}}$ provides the same function for 16-bit I/O expansion devices as the $\overline{\mathrm{MCS16}}$ signal provides for 16-bit memory devices.

***Note:*** *The Élan™SC300 and ÉlanSC310 microcontrollers internally OR together $\overline{\mathrm{MCS16}}$ and $\overline{\mathrm{IOCS16}}$. The ORed signal is looked at by the microcontrollers on both memory and I/O accesses (including I/O accesses that are internal to the microcontroller).*

## ASSERTING $\overline{\mathrm{MCS16}}$ FOR $\overline{\mathrm{ROMCS}}$ ACCESSES WITHOUT ADDRESS DECODE

The ÉlanSC300 and ÉlanSC310 microcontrollers support 16-bit wide memory in the $\overline{\mathrm{ROMCS}}$ space. On reset, these microcontrollers begin fetching from this memory. The width of the $\overline{\mathrm{ROMCS}}$ memory is controlled by the $\overline{\mathrm{MCS16}}$ input. If the $\overline{\mathrm{MCS16}}$ signal is asserted, $\overline{\mathrm{ROMCS}}$ memory is treated as 16 bits wide; otherwise, this memory will be treated as 8 bits wide.

Is there a way for a system designer to assert $\overline{\mathrm{MCS16}}$ for $\overline{\mathrm{ROMCS}}$ accesses without doing an address decode? First, note that you cannot simply tie $\overline{\mathrm{MCS16}}$ Low, even in a system where all memory is 16 bits wide. This does not work because the ÉlanSC300 and ÉlanSC310 microcontrollers internally OR together $\overline{\mathrm{MCS16}}$ and $\overline{\mathrm{IOCS16}}$. Thus, tying $\overline{\mathrm{MCS16}}$ Low will result in all I/O accesses being incorrectly treated as 16 bits.

A better approach would be to tie $\overline{\mathrm{ROMCS}}$ to $\overline{\mathrm{MCS16}}$. $\overline{\mathrm{MCS16}}$ is then correctly asserted only during $\overline{\mathrm{ROMCS}}$ cycles. The problem with this approach is that, on power-up, the $\overline{\mathrm{ROMCS}}$ signal is internally gated with $\overline{\mathrm{MEMR}}$ and $\overline{\mathrm{MEMW}}$. This means that $\overline{\mathrm{MCS16}}$ will not be valid until the $\overline{\mathrm{MEMR}}$ or $\overline{\mathrm{MEMW}}$ signal is asserted. The timing requirements for the assertion of $\overline{\mathrm{MCS16}}$ during a $\overline{\mathrm{ROMCS}}$ cycle are basically the same as for an ISA cycle. $\overline{\mathrm{MCS16}}$ must be valid 35 ns after LA is stable. If $\overline{\mathrm{MCS16}}$ is gated with the $\overline{\mathrm{MEMR}}$ or $\overline{\mathrm{MEMW}}$ command, it will not be valid until about 90 ns after LA.

The ÉlanSC300 and ÉlanSC310 microcontrollers have a programmable option in index register B3h that allows $\overline{\mathrm{ROMCS}}$ to be enabled as a simple address decode and not gated with $\overline{\mathrm{MEMR}}$ or $\overline{\mathrm{MEMW}}$. Using this option, $\overline{\mathrm{ROMCS}}$ (and, hence, $\overline{\mathrm{MCS16}}$) will be stable when SA is stable, which is 20 ns after LA is stable, thus meeting $\overline{\mathrm{MCS16}}$ timing requirements.

Unfortunately, as previously noted, the ÉlanSC300 and ÉlanSC310 microcontrollers power up with $\overline{\mathrm{ROMCS}}$ gated with $\overline{\mathrm{MEMR}}$, and therefore the initial code fetches treat $\overline{\mathrm{ROMCS}}$ as 8 bits wide. In this mode, when fetching from a 16-bit wide memory, the ÉlanSC300 and ÉlanSC310 microcontrollers will fetch the same byte on both even and odd memory accesses. Executing the reset code under this restriction requires you to program index register B3h, so that the $\overline{\mathrm{MEMR}}$ gating can be turned off, allowing proper 16-bit reads to occur. The code example on the next page shows a reset code stub that meets this requirement. In the example, all of the code fetched before the jump to the label “fetching16” has the same byte in the even and odd addresses. Execution starts at the label reset_vector, which would be located at FFFFF0.




Publication# **21825** Rev: **A** Amendment/**0**
Issue Date: **July 1997**

## Reset Code Example

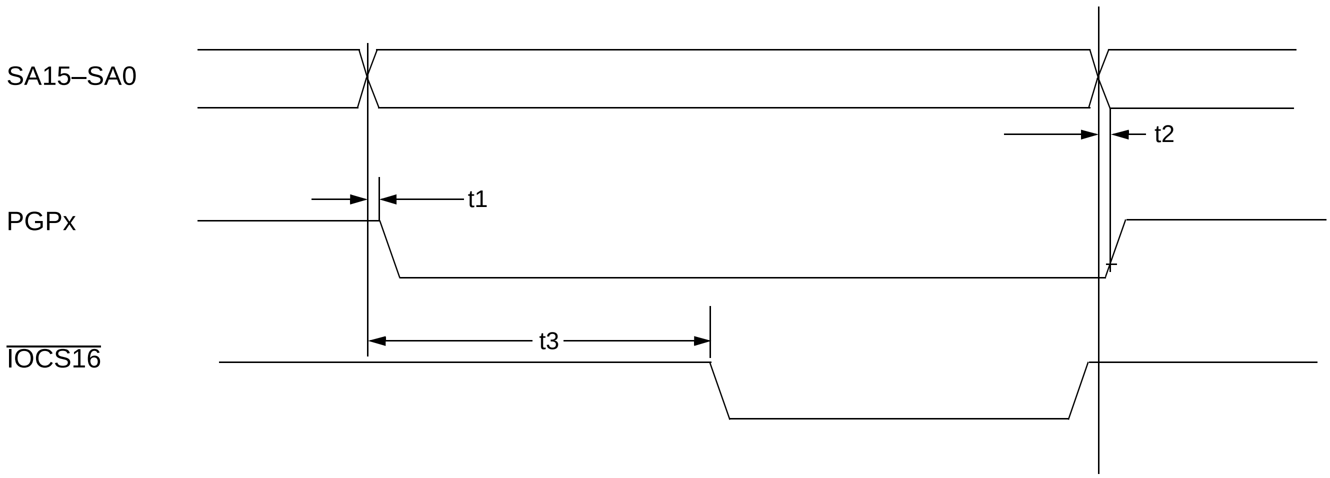
```
code segment
        assume cs: code

        org      0ffd7h

fetching 16:
        ;when we reach here, we're fetching 16 bits,
        ;so we can use normal instructions to jump to
        ;the real boot code

        db       0eah                ;jmp to real start of code
        dw       0, 0f000h           ;wherever that may be

        org      0ffdch              ;must be located at 0ffdch

        ;this code is set up so the even and odd bytes are duplicates
        ;this code just puts 04B3 in ax and outputs ax to port 22h
        ;this sets bit 2 of index register B3, enabling ROMCS
        ;to be an address decode and not gated with MEMR/MEMW
                                    ;ODD EVEN
        db       0b4h                ;B4        duplicate of mov ah opcode
fetching8:                           ;          this location must be at FFDD
        mov      ah, 004h            ;B4 04
        add      al, 0b0h            ;B4 B0     harmless, sets up next instruction
        mov      al, 0b3h            ;B0 B3
        mov      bl, 0e7h            ;B3 E7     harmless, sets up next instruction
        out      022h, ax            ;E7 22
        and      ch, bl              ;22 EB     harmless, sets up jump short instruction
        jmp      short fetching 16   ;EB EC     the fetches done at the target of this
                                                jump will be 16 bits
        db       EC                  ;EC        this code never gets executed so its
                                                contents do not matter
        nop
        nop
        nop
        nop

        org 0fff0h                   ;          reset vector
reset_vector:
        jmp      short fetching8     ;EB EB     jumps to FFDD

code ends
        end
```

## PGP PINS AS ADDRESS DECODES AND $\overline{\text{IOCS16}}$ TIMING

The general-purpose programmable PGP0 and PGP1 pins can be programmed as inputs or outputs using index register 70h bit 6 for PGP0 and index register 74h bit 2 for PGP1. PGP2 and PGP3 are output only.

The PGP3–PGP0 pins can be individually programmed as decoder outputs or chip selects for external peripherals using bits 6–0 of index registers 94h for PGP2, 95h for PGP3, 9Ch for PGP1, and 89h for PGP0. In address decode mode, bits 6–0 of these registers correspond to the SA address bits SA9–SA3, which provide address decodes from 0h–3F8h in 8-byte increments. To use the PGP3–PGP0 pins to drive $\overline{\text{IOCS16}}$, they must first be configured as address decode only in index register 91h, and then they will meet the timing in Figure 1 below.

Figure 1 shows the simulation result for timing requirements for PGPx and $\overline{\text{IOCS16}}$. For detailed timing between other signals, refer to ISA I/O 16-bit read/write cycle timing diagrams in the *Élan™SC300 Microcontroller Data Sheet*, order #18514.




***Notes:***
1. *t1: SA stable to PGP falling edge 10 ns (maximum) when programmed as address decode only (index 91h)*
2. *t2: SA stable to PGP rising edge 10 ns (maximum) when programmed as address decode only (index 91h)*
3. *t3: SA stable to $\overline{\text{IOCS16}}$ active 95 ns (maximum)*

**Figure 1. PGP Pins as Address Decodes Versus $\overline{\text{IOCS16}}$ Input Timing Requirements**

## 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 Élan™SC310 Devices' ISA Bus Anomalies Application Note*, order #20747