Apple II Technical Notes

Developer Technical Support

Apple IIe
#5: /INH Line

Revised by: Glenn A. Baxter
November 1988
Written by: Peter Baum May 1984

This Technical Note describes how a peripheral card on the Apple IIe should use the inhibit (/INH) line. This information is true for the Apple IIe only.

Using the /INH Line on the Apple IIe

Overview

One of the new features of the Apple IIe is the ability to add more memory or override existing memory from a peripheral card. This feature, which uses the /INH line on the peripheral slots, has been expanded from its original purpose on the Apple][+ of disabling the on-board ROM and allowing the language card (RAM) to reside in the same address space. The Apple IIe allows any part of memory to be replaced by memory on a peripheral card. This Note explains how a peripheral card should use the /INH line.

Uses

Presently, only a few peripheral devices use the /INH line in the IIe for memory expansion. One type of card uses /INH for RAM expansion by switching in extra language cards, while another class of cards uses it to extend the built-in 80-column ROM code by replacing it with their own ROM code. Other cards use /INH so that they can have more than one stack and zero page. Future peripheral cards can take advantage of the /INH line to do even fancier memory expansion, such as keeping multiple programs running in memory at the same time.

More memory, either ROM or RAM, can be added by mapping the memory into the same address space as existing memory. The processor can then select which memory, the on-board or the additional, it wants to use by setting a register (or soft switch). This technique of switching different blocks of memory into the same address space is called bank switching. An example of this technique for extending memory is found in the Apple][+ language card and in the bank switched memory on the IIe.

How It Works

When the /INH line, pin 32 in slots 1-7, is pulled low, all memory on the motherboard and in the auxiliary slot is disabled (including memory on the 80-column and extended 80-column cards). This action allows a peripheral card in slots 1-7 to enable its memory onto the bus.

When the 6502 reads a byte from memory, the data typically comes from one of three places: motherboard ROM, motherboard RAM, or RAM on one of the 80-column cards in the auxiliary

slot. When the /INH line is pulled low, all of the above mentioned ROM and RAM is disabled and will not drive the data bus. This disabling allows the peripheral slots to drive the bus by enabling data onto it. The 6502 will then read data from the peripheral card instead of a location on the motherboard or auxiliary slot.

During a 6502 write cycle, if the /INH line is pulled low, then motherboard and auxiliary card RAM are both disabled. A peripheral card can then read a byte off the data bus and store it.

Implementation

Because pulling the /INH line low disables all of memory, the peripheral card must be very careful when it does this. If only a small piece of memory is to be banked into a specific address space, then the /INH line should only be pulled on memory references to that address space. Otherwise the motherboard memory will be disabled and the processor will read or write to the wrong memory and the program will not work properly. For example, if a peripheral card wants to replace the zero page with memory on the card, then the /INH line should be pulled low only on references to the address space between $0 and $FF. If the /INH line is pulled during a processor instruction fetch from the monitor ROM at $F800, the 6502 will read the wrong instruction (or a floating bus) and probably crash the program.

Pulling the /INH line at specific addresses is called select decoding. The hardware on the peripheral card does this by checking the address bus of the 6502, and if the address falls in the correct range, the card pulls the /INH line low. In the earlier example of a new zero page, if the address bus was in the range $0-$FF the card would pull /INH low.

Differences: IIe vs.][+

On the Apple][+, select decoding was not necessarily needed because the /INH line only affected the ROM and not the RAM. If the Apple][+ peripheral card wanted to bank in extra language cards at the addresses $D000-$FFFF, then it could pull the /INH line and keep it low during any memory access. This action would disable the on-board ROM and not any other memory accesses such as zero page or stack. This same card would not work in the IIe, since the next instruction fetch to RAM after pulling /INH low would read a floating bus because all the memory would be disabled.

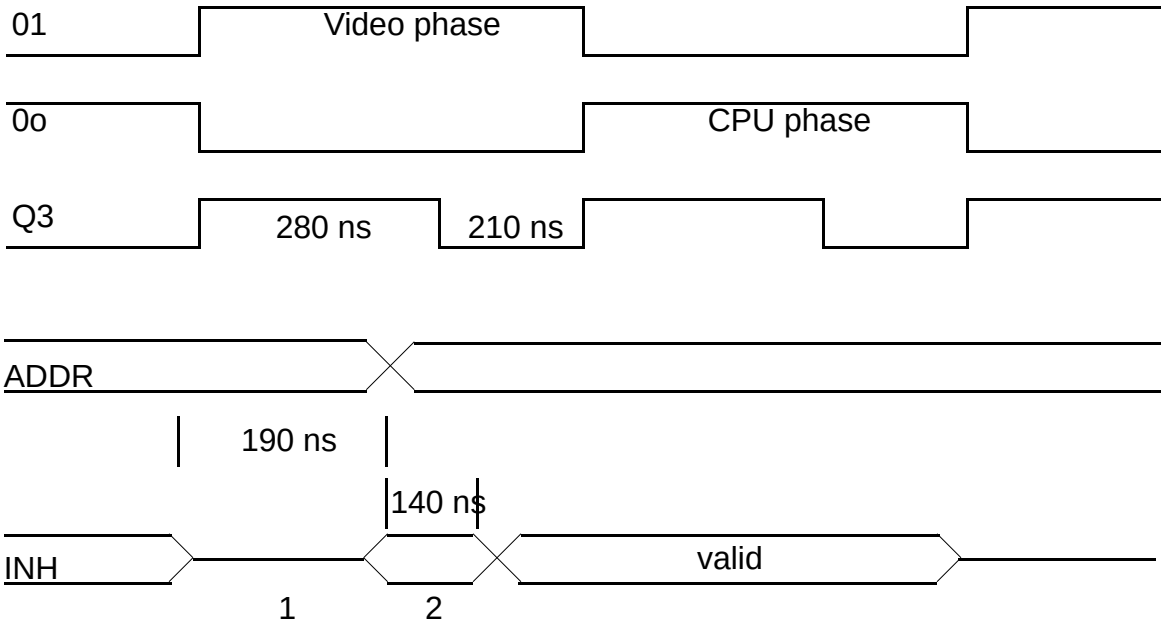
Another Feature

For those of you who love to muck around in the guts of the Apple IIe, one more feature has been added to the /INH function. The /INH line will also override DMA accesses to memory on the motherboard. This override means that if a peripheral card uses DMA to read or write to

memory, another peripheral card could pull the /INH line and process the DMA access. An example of this would be a co-processor card using the memory on a RAM card in another slot. Rather than have the co-processor write to the memory on the motherboard then have the 6502 write to the RAM card, the co-processor can write to an address that the RAM card recognizes. The RAM card could then pull the /INH line and it would be free to read or write the data bus. This technique could also be used by a co-processor to write directly to a printer card in another slot.

Timing

The peripheral card must wait for the address bus to settle, which occurs a maximum of 190 ns after the falling edge of 0o, before pulling the /INH line. (The 6502A maximum address setup time is 140 ns from 02, with a worst case 6502A skew of 50 ns from 0o to 02.) To guarantee that the RAM is disabled and a write does not accidentally take place to the motherboard, the /INH line must be pulled low within 330 ns of 0o.



1. The INH line can be pulled high at this time.
2. The INH line can be pulled low (or high) after the addresses are valid at 190 ns, but before 300 ns (from 0o).

Figure 1–INH Line Timing Signals

Circuits

Figure 2 illustrates a simple example of a circuit that can be used to implement the /INH function.

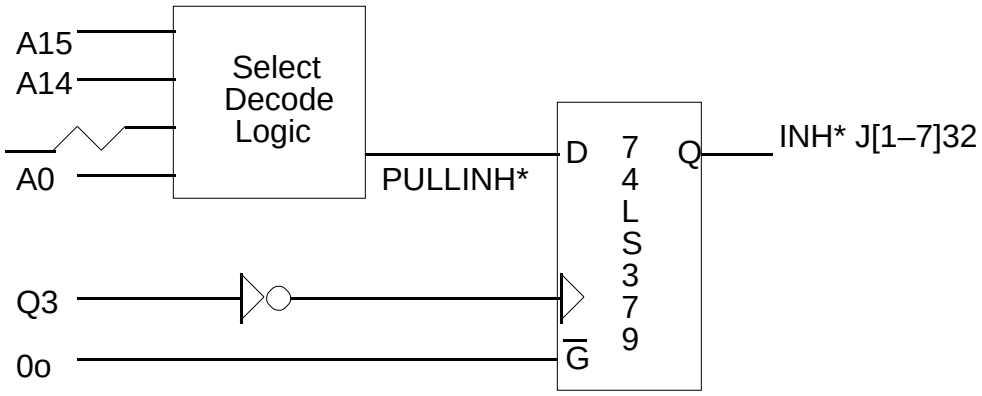


Figure 2–Circuit Implementing /INH Function

An Application

The circuit in Figure 3 can be used to replace the code in the monitor ROM, from location $FC00 to $FFFF, with custom code. Anytime the address space between $FC00-$FFFF is accessed, the /INH line is pulled low, the motherboard memory is disabled, and the circuit's 1K RAM is enabled instead. Part of this feature can be disabled and the motherboard memory can be read by keeping the switch connected to +5 volts (READDIS). Whenever the system writes to any location in the address space $FC00-$FFFF, the circuit will disable any RAM on the motherboard and instead write into the 1K RAM.

Here is a series of commands that can be used with the circuit to replace the reset vector at $FFFC and $FFFD. A new reset routine can be written that will print the screen or save the status of all the registers whenever the Reset key is pressed.

Start the system with the circuit's switch connected to +5 (READDIS). Doing so will enable the system to read the monitor ROM during startup, before the 1K RAM has been initialized.

Get into the monitor by typing CALL -151. The system prompt should now be an asterisk (*). Copy the monitor ROM into the 1K RAM with the command FC00<FC00.FFFFM. Change the reset vector so that it jumps to location $300 with this command, FFFC:0, then copy your new reset routine into memory starting at location $300. Now, set the switch to ground (READEN) so all future read accesses to $FC00-$FFFF will read the 1K RAM.

For example, if these instructions are stored in memory starting at location $300, then the system will clear the screen and continue execution in the monitor when the Reset key is pressed.

```
$300:20 58 FC    JSR HOME    (clears screen)
$303:4C 65 FF    JMP to MON (resume execution in monitor)
```

One of the problems with this circuit is that it also overrides any accesses to the language card, therefore any program that uses the language card will not work with it. The circuit does not keep track of which memory is enabled, ROM or language card RAM, in the $FC00-$FFFF space.

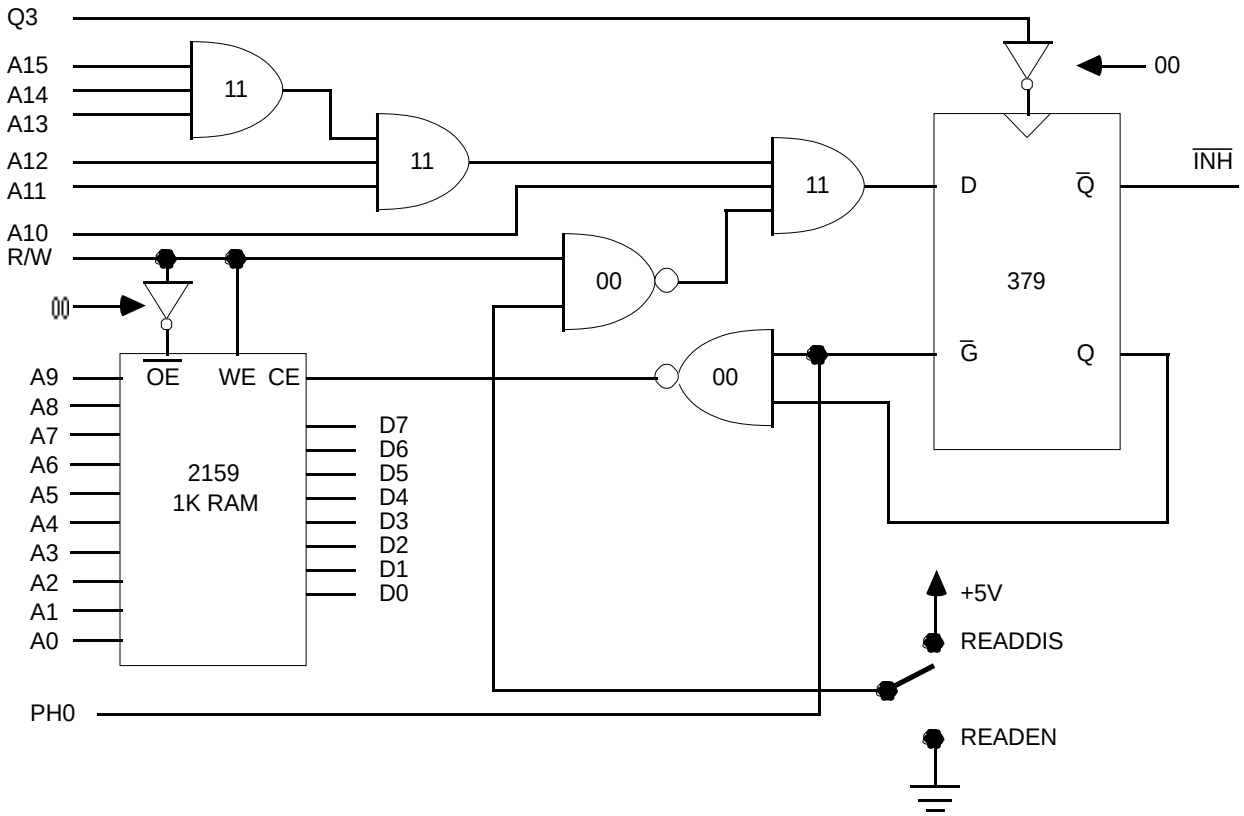


Figure 3–Circuit to Replace Monitor ROM Code

Further Reference

- *Apple IIe Technical Reference Manual*