NIBBLING at the GAME PADDLE PORT

This exploration of the Apple II family's game paddle port circuitry and software will help you design your own software for custom use of this powerful port.

by Peter Baum
Apple Computer, Inc.
MS 22-W
20525 Mariani Ave.
Cupertino, CA 95014

The game paddle port on your Apple provides an extremely versatile method of obtaining input. While it serves as a game controller for both paddles and joysticks, it has also been used for applications that range from a simple <SHIFT> key input to serial communications input. To help you get the most out of your hardware, this article will explore the basic game port circuit, how the port is read, and some of the anomalies that complicate this task.

Circuit Description

The value returned from the Apple paddles (or joysticks) is determined by a software timing loop reading a change of state in a timing circuit. Each paddle consists of a variable resistor (from 0-150K ohms) which makes up part of the timing circuit. There is a routine in the Monitor ROM, called PREAD, which counts the time until a state change occurs in the paddle circuit and translates this into a value between 0 and 255.

Figure 1 is a block diagram that shows the paddle circuit for the Apple II Plus, //e and the //c. The large block on the left illustrates part of the circuitry inside the 558 timer chip. The 558 chip consists of four of these blocks, with all four paddle trigger lines shorted together on the motherboard and activated by the soft switch at $C070. The outputs of the 558 chip run into a multiplexer, which places the appropriate signal onto the high bit of the data bus, when a paddle softswitch address in the range $C064-7 is read. The Apple //c uses a 556 timer rather than the 558 chip and only supports two paddles, 0 and 1.

The 100 ohm resistor and .022 microfarad capacitor are on the motherboard, with the variable resistor in the paddle. Each of the four paddle inputs have their own capacitor and resistor. Since these components can vary by as much as five percent from Apple to Apple, this circuit is not a very exact analog-to-digital converter. If a paddle is moved from one Apple to another without changing the resistance (turning the knob), the paddle read routine will probably calculate a different value for each machine. About the only feature of the paddle read routine on which a programmer can depend, is that the value returned will rise if the paddle resistance increases and fall if the resistance decreases.

The paddle timing circuit on the Apple II Plus and //c is slightly different than the one on the Apple //e. On the //e the 100 ohm fixed resistor is between the transistor and the capacitor, while the variable resistor in the paddle is connected directly to the capacitor. On the II Plus and //c the capacitor is connected directly to the transistor and the fixed resistor is in series with the paddle resistor.

"...this circuit is not a very exact analog-to-digital converter."

An Example of a Typical Paddle Read Routine

The timing circuit works by discharging a capacitor through a transistor; then shutting the transistor off and letting the paddle charge the capacitor by supplying current through the variable resistor. The rate at which the capacitor charges is a function of the variable resistance; the lower the paddle resistance, the greater the current and the faster the capacitor charges. When the capacitor reaches a predetermined value, it changes the state of a flip-flop. The paddle read routine counts the time it takes for the capacitor to rise and change the flip-flop.

Let's step through an example of a typical paddle read operation, as shown in **Listing 1**, the ROM routine PREAD. For now we'll assume that the capacitor has already been discharged. (Later, I'll explain when this assumption can be made and when it can't.)

The software starts by reading the softswitch at location $C070, which strobes the trigger lines on the 558 timer. This causes two events to occur: the output signal (which is read at $C064-$C067 for paddle 0-3, respectively) goes high, and the transistor turns off.

The software, after initially strobing the trigger line, executes a timing loop that reads the state of the output signal. When the output signal changes from high to low, the software jumps out of the timing loop and returns a value indicating the elapsed time. The Monitor PREAD routine consists of an 11 microsecond loop and will return a value between 0 and 255. (**Note:** The firmware listing is wrong and says the loop is 12 microseconds.) The timing loop returns 255 if the circuit takes longer than 2.82 milliseconds for the state change to occur.

Inside the 558 timer chip, when the trigger is strobed low, the comparator that feeds the set input of the flip-flop is triggered, which in turn sets the output of the 558 timer. At the same time the transistor, which has held the capacitor near ground by sinking current from it, is shut off. The capacitor can now charge up using the current supplied by the paddle. The smaller the paddle's resistance, the more current the paddle supplies and the faster the capacitor charges. After some time, the capacitor will charge to the threshold value of 3.3 volts, which is set by the voltage divider network in the 558 timer; and the comparator that feeds the reset input on the flip-flop will trigger. This sets the output signal ($C06x) of the 558 timer low, which indicates to the software that the circuit has timed out. (See **Figure 2**.)

Resetting the flip-flop turns the transistor on, which discharges the capacitor very quickly (normally less than 250 nanoseconds). That paddle can then be read again.

A Closer Look at the Hardware

The First Anomaly

Notice that the last sentence states that the *paddle* can be read again and not the *paddles*. If another paddle is read immediately after the first, it may yield the wrong value. To show this I'll step through an example of reading a second paddle immediately after finishing the first.

In this example I'll assume that the first paddle has been set with a very low resistance, while the second paddle has a high resistance. The first paddle will time out very quickly and return with a small value, while the second paddle will take longer and yield a larger value.

We start reading the paddles by testing the paddle outputs to see if they're low, which indicates that the capacitor has been discharged. Assuming that the outputs are low, the next step is to trigger the 558 timer ($C070), which turns off the transistor and allows the capacitors to charge. Since all of the trigger input lines are shorted together, all four of the capacitors will charge up, but at different rates, since the paddle resistances have been set to different values. The voltage on the capacitor for the first paddle will reach the threshold voltage very quickly, since the paddle resistance has been set low. Therefore, the timing loop will time out quickly.

At this point since the paddle resistance for the second paddle was set to a high value, its capacitor is still charging and has not yet reached the threshold value. The transistor for the second paddle is still turned off, due to the initial trigger used for reading paddle one. This means that the capacitor for the second paddle has not been discharged.

An attempt to read the second paddle now will yield only false results. The capacitor is partly charged and therefore will reach the threshold value much faster than if the capacitor had been completely discharged. If the timing loop is used, it will return with a smaller value than it would if the capacitor had been completely discharged. Notice that retriggering (reading location $C070) the 558 timer will not help, since that only keeps the transistor turned off and doesn't help discharge the capacitor. The only way for the capacitor to discharge is to let the circuit time out completely, by letting the capacitor charge until it resets the flip-flop.

To read the second paddle the capacitor must first be discharged, which is only done when the threshold value is reached and the 558 timer flip-flop is reset. The only way to guarantee that the capacitor is discharged is if the transistor is on. This condition is met when the paddle output is low. Therefore, start every paddle read by either waiting for at least three milliseconds before strobing the trigger input, or testing to make sure that the paddle output is low.

If after four milliseconds the paddle output is not low, then there is a good chance that no paddle is connected. It may also indicate that a peripheral is attached that has a larger maximum resistance than the 150K ohms used by the Apple paddles. Some peripheral devices use this technique so that more than 256 points of resolution can be determined. Of course, this requires a custom software driver and the Monitor PREAD routine can't be used.

The Apple //e Anomaly

The problem with //e paddle input is that the capacitor may not be discharged by the transistor. Typically, the transistor will discharge the capacitor in less than 250 nanoseconds on the II Plus. But on the //c, if the paddle resistance is very low, the paddle may supply enough current to keep the capacitor charged at all times.

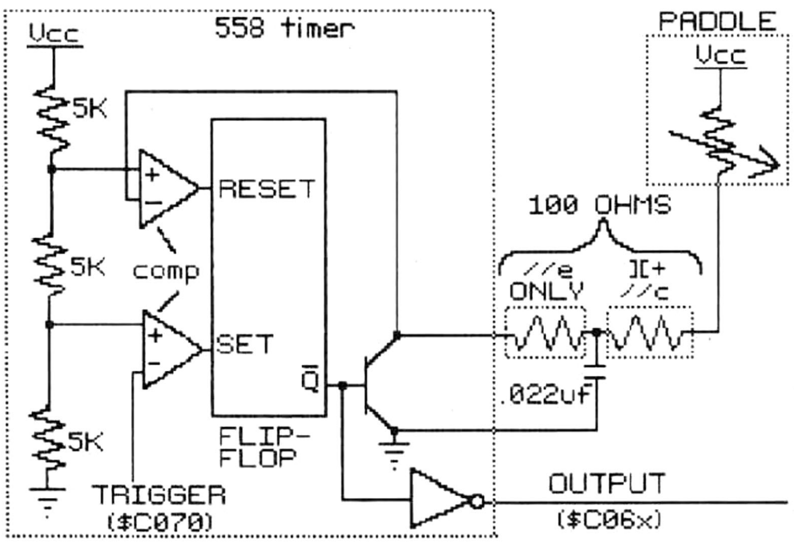
Because the fixed resistor (100 ohms) on the //e motherboard is between the capacitor and the transistor, there will be a voltage drop across the resistor if the capacitor stays charged. When the transistor is shut off by the trigger strobe, this voltage drop will disappear and the capacitor, which may be near the threshold voltage, will trigger the reset comparator earlier than it would if the capacitor had been discharged completely. The net effect of this is that the paddles will read zero on the //e when they would read a small value on the II Plus or //c.

Other circuits that expect the capacitor to discharge completely may not work properly. A circuit that attempts to simulate a paddle through active components, such as a digital-to-analog converter, may be able to provide enough current so that the capacitor never discharges and the paddle always reads zero.

Conclusion

The best way to really understand the game paddle port is to experiment with writing your own software. One interesting possibility might be a routine that reads two paddles at once. The software loop would not have the 11 microsecond resolution of the PREAD routine, but you'll find that it works amazingly well. Whatever you attempt, though, your chances of success should be much greater if you keep the actual mechanics of the port in mind.

Figure 1: Paddle Circuit for II Plus, //e and //c



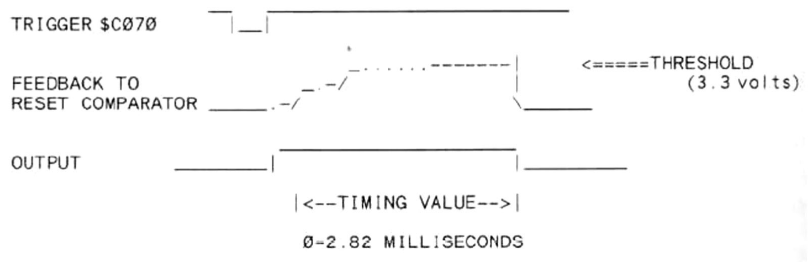


Figure 2: Timing Sequence

LISTING 1: PADDLE READ ROUTINE

```
FB1E:AD 70 C0   PREAD      LDA   PTRIG      ;TRIGGER PADDLES
                                             (PADDLE NUMBER (0-3) IN X-REG
FB21:A0 00                 LDY   #0         ;INIT COUNTER
FB23:EA                    NOP              ;COMPENSATE FOR 1ST COUNT
FB24:EA                    NOP
FB25:BD 64 C0   PREAD2     LDA   PADDL0,X   ;COUNT EVERY 11 USEC.
FB28:10 04                 BPL   RTS2D      ;BRANCH WHEN TIMED OUT
FB2A:C8                    INY              ;INCREMENT COUNTER
FB2B:D0 F8                 BNE   PREAD2     ;CONTINUE COUNTING
FB2D:88                    DEY              ;COUNTER OVERFLOWED
FB2E:60         RTS2D      RTS              ;RETURN W/VALUE 0-255
```