

# Lecture 2 Notes

*January 26, 2007*

*He's a Pinball Wizard  
He's scored a trillion more!  
A pinball wizard,  
The world's new pinball lord!*

## Introduction

A pinball machine is relatively complicated GUI to a very simple game. Its interface is as follows:

<b>Inputs</b>	<b>Outputs</b>
Playfield Switches	Solenoids (moving components)
Flipper Buttons	Flippers (driven by solenoids, but controlled separately)
Coin Slot	Lamps (Extra Ball, Game Over, etc.)
Credit Button	Loudspeaker
	Displays (4 player scores, 2 general game state)

The primary job of the game code itself, then, is to map inputs received from the user (such as the ball hitting a switch, or a coin being inserted) to their appropriate outputs (such as the score or credit-count being incremented).

This would be difficult enough to accomplish in real time, but this is far from being the software's only responsibility. The lamps, for instance, are pulse-modulated; in order to get the effect of  $6VDC$  (which the lamps require), the  $18V$  signal must be rapidly strobed on and off such that it is only enabled  $\frac{1}{3}$  of the time. If this is done too slowly, the lamps appear to flicker; too quickly, and they don't turn on at all. Similarly, the score displays cannot be given more than one digit to display at a time, and that digit will only glow for about  $10ms$ . In order to have working displays, the digits must be sent in series repeatedly. Again, the displays cannot be updated too slowly or too quickly, or they will behave unpredictably.

All of this must be handled in software (since it would have been too expensive to implement it in hardware), in real time, at same time and on the same processor as the game itself. This is the challenge before you.

## The Machine

However, let us first spend some more time examining the pinball machine you will be using.

### Displays

The *Williams System 6 Pinball Machine* has six displays: four six-digit displays (used to display each player's current score while the game is running), and two two-digit displays (used to display various and sundry pieces of information about the state of the game as a whole). These are all controlled by a display control board, which, in turn, is controlled by the CPU board. The interface presented by the display board allows the user to select a digit (by raising the voltage on one of 16 *strobe lines* corresponding to the digits), and to specify a value for that digit (in BCD). As was mentioned earlier, this must be done approximately 1000 times per second, if the displays are to function properly.

At the time that our machine was made (c. 1981), LED displays were simply not bright enough for use in a crowded and distracting game hall, so gas-discharge displays are used instead. These displays are quite bright, but they run on 100V and are prone to violent self-destruction. Most modern-day machines use either LED displays or an LCD monitor.

### The Driver Board

Aside from the displays, every element of the pinball machine is controlled from the driver board, via one of three PIA (Peripheral Interface Adapters) on that board. These chips handle the actual grunt-work of I/O (each being able to read from or write to sixteen lines of input/output), and since communication with the PIAs is handled directly through the CPU's data and address busses, the game code itself is insulated from the task of handling I/O to most elements of the machine directly. This simplifies the programmer's job quite a bit.

Since it is the driver board that directly controls the lamps, switches, and solenoids (and, in a sense, the speaker—we'll get to this later), it is the driver board that is responsible for stepping the small voltage used in the machine logic (5VDC) up to the much larger voltages required to drive the

mechanical components of the machine (ranging from 18VDC to 100VDC), using solid-state relays. The problem here is that the machine has many, many components, and, in order for each one of them to be controllable separately, each would need its own relay, capable of handling rather high voltages—and these, even now, are quite expensive. For this reason, as well as for others, the Williams System 6 machine was designed with a workaround.

Since there are approximately 64 playfield switches (the same is true for the lamps), these are arranged on an  $8 \times 8$  grid. Now, in order to read the state of one switch, one need only signal on one of the eight columns, and listen on all eight rows. If a switch in the current column is in the *on* position, then the voltage of its row will be high. Now, in order to be aware of all of the switches on the board, we need only repeatedly strobe over all of the columns, continuously reading the state of the rows. There are only two problems with this system: firstly, it is possible, if we are not strobing quickly enough, to miss a rapidly pressed and released switch, and secondly, we cannot sense more than one switch at a time, meaning that we can only use one ball.

A similar system is used to control the playfield lamps. This adds the additional complication that, in order to get the appropriate ratio to shift the voltage from 18v to 6v, we can only be strobing over three columns at a time (so that each is on  $\frac{1}{3}$  of the time).

### Solenoids and Sounds

Solenoids, despite being controlled by the driver board, are controlled directly—since there are only 12 of them, there is no need to use a grid. In fact, since the PIA controlling the solenoids is capable of controlling sixteen separate output lines, and only twelve are actually used by solenoids, the remaining four (“Solenoids” 13-16) are re-routed to the sound board. Since sound, in this machine, is entirely handled by its own, separate processor, all the game programmer needs to do is to select a sound out of the available four to be played, and the sound board will handle the rest.