Microwave Anti-Beep Mod

In 2020 I bought a Magic Chef microwave oven, model HNN1110W, and it has a very annoying "feature". When cooking ends, it produces five loud beeps at about 1Hz, and it insists on producing all five beeps even after you've opened the door. It also won't allow you to enter anything new on the keypad until after the beeping ends. I don't see how anyone could have thought this was a good idea, but there it is.

I've been thinking how I might stop the beeping, and of course the simplest approach is to just cut the lead going to the beeper. That's not really the ideal solution because it's useful to have some beeps. But it turns out that at any point during the cooking countdown you can hit Stop, and it will pause. Then if you hit Stop again, it cancels everything and returns to idle - but without the five beeps.

So the idea is to install a microcontroller that would tap into the display and the keypad, and automatically detect when the countdown has reached one second, then electronically press the Stop key. The display is seven segment, four digits, and it connects to its PC board through 13 pins, which is right for multiplexing seven segments and four digits, plus two pins to control the ":" in the middle. The keypad has 24 pads on it, and the flex cable connecting to it has 6 + 4 traces, which is some variation of rows and columns.

The display's "digit" lines swing through the full five volts, and are low when the digit is powered. However, the "segment" lines are low when idle, and only rise to about 2V when powered. So they require voltage translation to be properly read. After taking pictures of the display at various shutter speeds, and following the various lines with my scope, I confirmed that the driving algorithm consists of two multiplex cycles:

1. Beginning with the left-most (most significant) digit, the processor cycles through the four digits one at a time, diplaying segments A, B, C and D of each digit for one millisecond. Then a fifth millisecond is devoted to the ":", which has its own digit and segment lines.

2. In the second cycle the processor does the same thing again, but this time displaying the E, F and G segments of each digit, again followed by the ":".

So to determine what is being displayed, one needs to OR together the lit segments from the two cycles for each of the four digits. I only needed to detect the pattern "blank, zero, zero, one" which only occurs at the one-second point in the countdown. Only four segments need be analyzed to detect these states to the exclusion of all other possible patterns - segments A, B, D and G. And since only four segment inputs are needed, it made sense to use an on-hand quad comparator for voltage translation - a TLC374.

My microcontoller of choice was the Arduino Pro Mini - either 16MHz or 8MHz, and both work at 5V. The four digit lines are connected to D2 through D5, and the four segment lines from the comparator are connected to A0 through A3. When a digit line goes low, the processor reads the current value of the segment lines and saves it. Then after the two cycles have completed, it calculates the ORed value for each of the four digits and compares them to the target pattern.

If there's a match, D6 brings an N-channel mosfet gate high, turning the mosfet on. The row and column pins of the Stop key are connected to the drain and source of the mosfet, so turning on the mosfet has the same effect as pressing the physical Stop key.

The well-commented code for the Pro Mini is included here, as well as the schematic and pictures showing the project at various stages.

One construction note: To permit using a smaller board, there are no female headers under D7 through D13 of the Pro Mini. So those Pro Mini pins are just hanging in mid-air, not connected to anything. That permits using those protoboard rows for other things, such as the segment input lines and the mosfet. I used one-half of a protoboard that has the same pattern as a breadboard:

https://www.ebay.com/itm/265276759831