Measuring Resistance Using Digital I/O

Using a Microcontroller for
Measuring Resistance Without using an ADC.

Get more bulletins just like this one:

Click Here.

http://www.best-microcontroller-projects.com/

Table of Contents

Rights

NOTICE: You DO Have the Right Give Away This Report,

As long as you don't change anything, use SPAM or generally do something you're not supposed to do.

http://www.best-microcontroller-projects.com

Legal

DISCLAIMER AND/OR LEGAL NOTICES:

Terms of use:

All software, hardware and information found in this ebook or associated with it shall be hereafter referred to as the product. All persons using or acting on the product shall be hereafter referred to as the user.

By using the product the user agrees to abide by the following conditions.

Disclaimer:

All products are provided 'as is' and without warranty of any kind, either expressed or implied. The entire risk as to the quality and performance of the product is borne by the user.
Should the product prove defective in any respect, the user will assume the entire cost of any service and repair and will assume full liability for any indirect, special, incidental or consequential damages arising out of the use of or inability to use the product, including, without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if advised of the possibility thereof, and regardless of the legal or equitable theory (contract, tort or otherwise) upon which the claim is based.

Resources

PIC C Course

A Complete C course for PIC micros in 11 parts.

CLICK the image.

Or Click Here

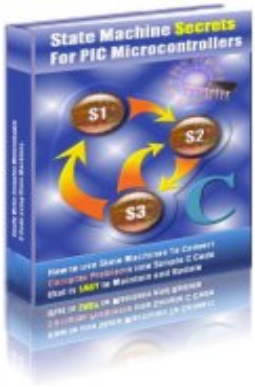
PIC Interrupt Secrets

How to Use PIC Interrupts to Maximum effect.

CLICK the image.

Or Click Here.

State Machine Secrets.



How to use state machines to create **Solutions** to complex problems including fully debugged examples – one is coding a scrolling menu on a standard LCD display.

CLICK the Image

Or Click Here.

Measuring Analogue Parameters With No ADC

Here's a trick I have been meaning to try and it could be useful for you if you either run out of ADC inputs or your device doesn't have one in the first place. It's not a true analogue measurement but allows you to measure a resistance that results in an analogue voltage e.g. components that vary their resistance in response to a physical parameter.

Quite often you will be using components that change their resistance depending on a physical parameter e.g. temperature (TCR), humidity, Light (LDR), potentiometer (POT) and the method described here will let you measure their resistances without using an ADC.

The idea is not new since it was first used on PCs to implement very cheap analogue input for joysticks using only digital inputs.

A single joystick is made from two orthogonal potentiometers 100k-470k and the trick is getting the value from the pots into the PC since, at the time, ADCs were tons of money so there was no way they would manufacture a joystick with ADCs - just for game playing!

It's done using capacitors.

Measuring A Resistance

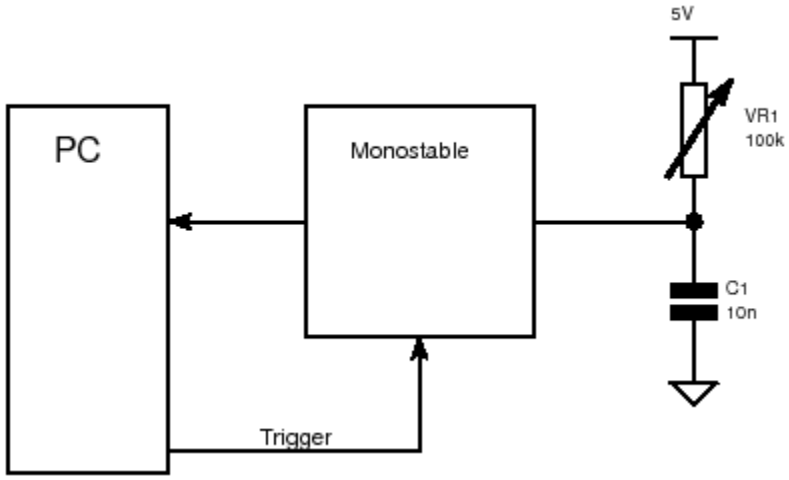
As any engineer will tell you their first experience of (difficult at the time) equations is when you solve the differential equations for charging and discharging a capacitor and you figure out the RC time constant Tor.

That time constant is the key since if you keep the capacitance the same and vary the resistance then the time taken for charging the capacitor will vary in proportion to the value of the resistance.

If you choose the capacitance and max. pot value appropriately then you can cycle through measurements quickly enough (for game playing) so that it appears that the pot is continuously read.

All you need is a way of discharging the capacitor and then apply a known voltage to the capacitor via the pot.

The Basic PC Joystick Interface
(reads one of the four potentiometers -
some game pads have two joysticks)



PC Joystick Method

This is the way that the PC joystick was read (although now more reliable methods are used) and it employs a monostable which when triggered discharged the capacitor. The monostable then sets its output low until the capacitor voltage reaches a threshold at which point the output goes high.

Therefore the time taken for the capacitor to charge is proportional to the value of the pot so the value of the pot can be estimated. By measuring the duration of the monostable signal you obtain a value that is proportional to the value of the resistance.

For the PC this was not a spectacular success for the following reasons:

- PC computer **speeds** were different from each other.
- Port **interfaces** were different from each other.
- Problems with **OS** differences.
- Big **tolerance** differences in components (different joystick manufacturers).
- Computer intensive **polling** of the port.
- Takes a **long time** to reach max pot value.

...and although the joystick worked it required calibration for each computer and each game, and required slightly different software to cope with different PC hardware and OS.

This gives you an idea of why this method is not for commercial usage.

But you won't have this huge problem because you will NOT be running the software on a multi-purpose platform so all you should be concerned about is the component tolerances (capacitor/potentiometer change with temperature and time (aging)) and power supply tolerance.

Note: You may need to calibrate depending on your application.

Calibration

One problem is that you may need to calibrate the system. And this is for two reasons:

1. Unknown inaccurate power supply.
2. Inaccurate pot/capacitance.

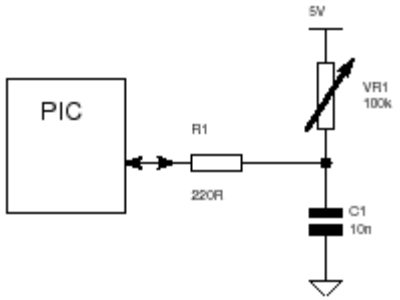
Note: Just remember to check and test and don't use this technique in a commercial system (where you run out thousands of boards because you will get into trouble) - alternatively reduce the resolution required e.g. for a 5V system measure 25%,50%,75%,100% of the supply etc. then the tolerance will be less important.

Conversion

Although the value is proportional it is on an exponential curve i.e. it's not linear so you need to use a table to convert it into resistance values or use a calculation in the code to convert from exponential.

Applied to the PIC

The difference between the PIC and the PC is that the PIC digital pin can be configured as an input or as an output on the fly (so you can discharge the capacitor by setting the pin as an output and at logic low) and then set it as an input and by using the standard PIC input voltage levels (2V - logic high) as the threshold you don't need any extra ICs.



Operation

The procedure for using it is:

1. Change the pin to an output.
2. Set the output low (to discharge the capacitor) for >2ms.
3. Set the pin to an input.
4. Start the timer.
5. Stop the timer when the input goes high.
6. Read out value from timer which is proportional to resistance.

By changing the digital input to an output (with the output set low) the capacitor will be discharged to zero current limited to 5V/220 = 23mA. (you could probably get away without using the 220R since the internal pin circuitry limits the current sink to 25mA - actually I forgot it when testing and it works fine without it - it's really a protection if the pot is set to zero resistance - I just tested it using a selection of resistors so I did not have that problem).

Then changing the pin to an input - it effectively becomes isolated from the circuit since very little current is drawn by an input pin - the capacitor will charge from nearly zero (nearly zero because of the 220R resistor - the input can't quite reach zero - this will be low enough to register as logic zero) to 5V via the potentiometer. As the input voltage crosses the 2V (ish) threshold for logic high the PIC input will be logic 1.

Note: For a more accurate (repeatable) threshold choose a Schmitt trigger input or use an internal comparator.

Measuring this time from reset to logic 1 will give a value proportional to the value of the potentiometer (one arm of the pot = variable resistor).

For the 100k & 10n components Tor = 1ms which is 63% of the time to charge

to the final voltage and this gives some idea of the order of time to be detected by the microcontroller.

The program outputs two values:

- A confidence loop counter (just a variable that increments)
- The Timer1 value.

Note: If you use Vista - putty.exe (google for it) is a suitable replacement for hyperterminal. Hyperterminal was removed from Vista!

RS232 Settings

You should set hyperterminal or putty to the following settings:

Baud	2400
Data Bits	8
Parity	None
Stop bits	1
Flow control	None

RS232 output

The terminal output looks like this:

```
39353
115
```

The upper value is the confidence output and lower is Timer1 value. This is kept at the same position on the screen using VT100 codes (I really hate the screen scrolling continuously - then again there's no history).

Something to Notice

If you remove the resistance completely then the code is always waiting for the capacitor to charge and it never will! This is an example of the type of problem that can be created by an unforeseen circumstance. Although this example is

trivial it demonstrates that if you don't allow for every different external or internal state the code could hang. The code is forever stuck unless the state changes. In this case it does not matter because the code is a demonstration and not critical but the way around it is to use the watchdog timer to force a reset.

By changing the code i.e. using a more complex project having a user interface it would be possible to allow the user to stop the test or put up a warning message. At the very least it would allow the user to interact with the system before it gets stuck again.

Device

The device used is the 12F675 (with the ADC turned off) this is just a device I had handy but you can run the code on any PIC chip.

Compiler

The compiler used is the MikroC compiler which lets you use it free for the 1st 2k Hex output.

Code Download

Download the code here: Click Here To Download