Debouncing Switches

Mechanical switches are one of the most common interfaces to a uC.

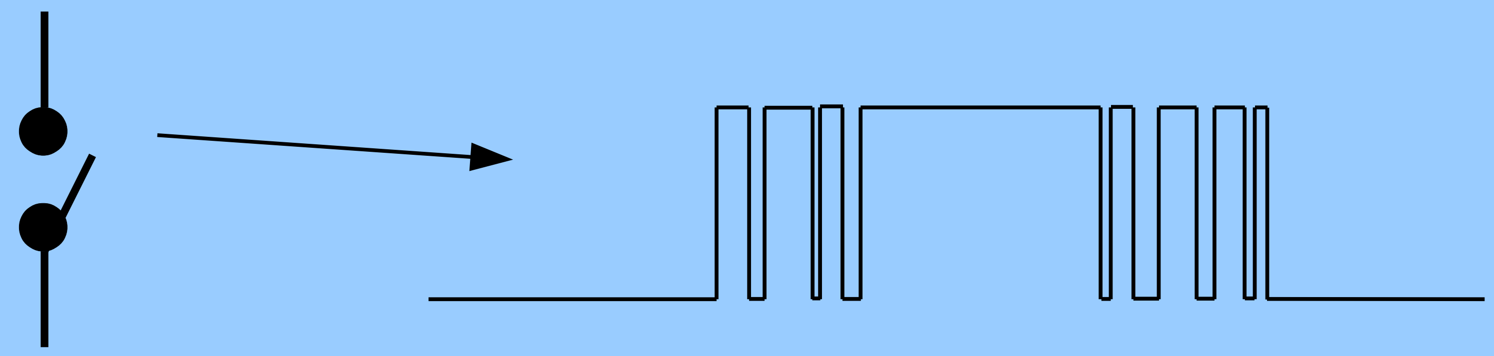
Switch inputs are *asynchronous* to the uC and are not e*lectrically clean*.

Asynchronous inputs can be handled with a *synchronizer (2 FF's)*.

Inputs from a switch are electrically cleansed with a switch *debouncer*.

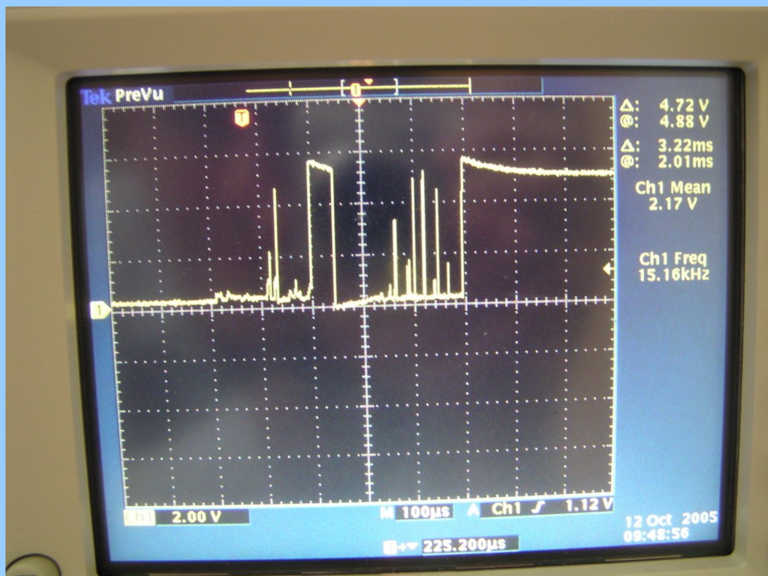
What is switch bounce?

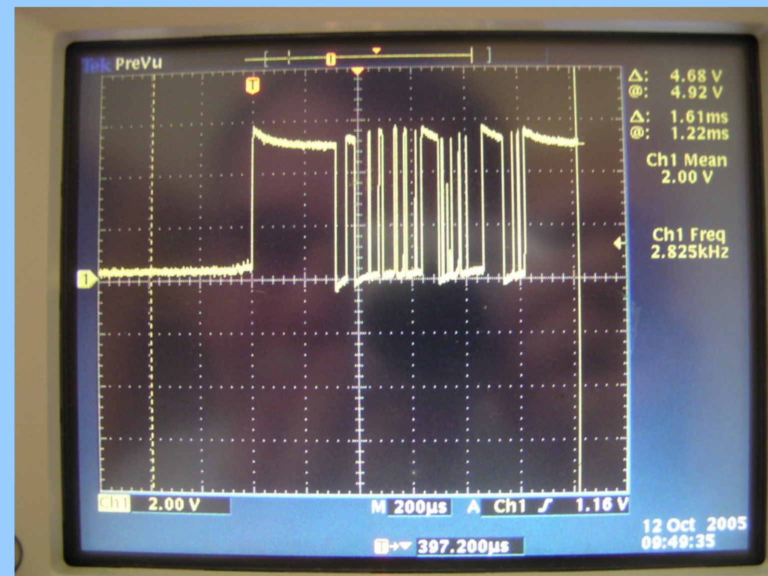
-The non-ideal behavior of the contacts that creates multiple electrical transitions for a single user input.

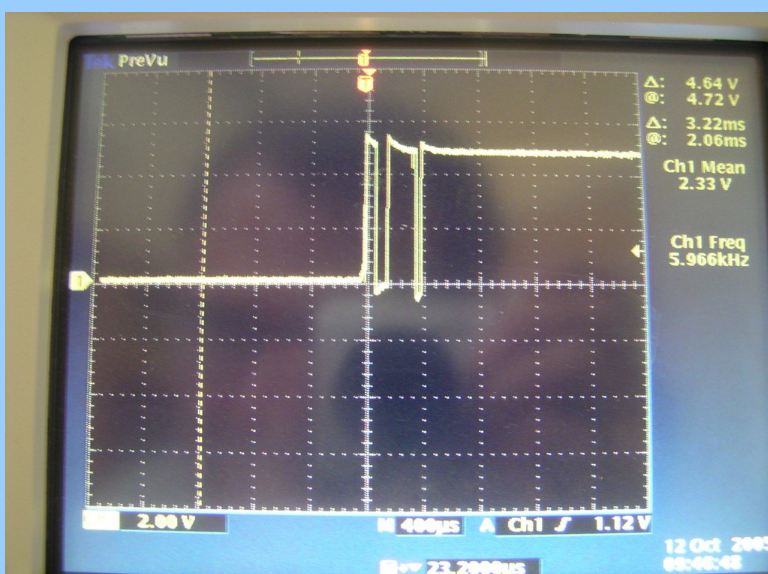


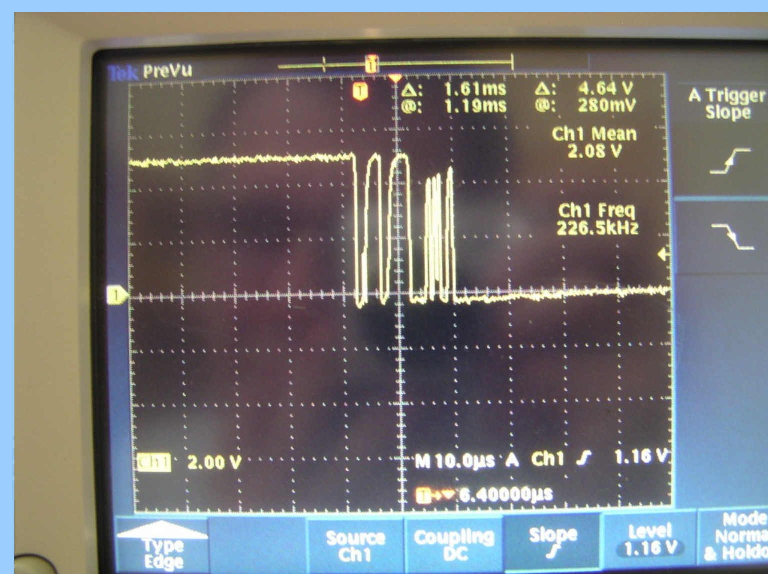
Debouncing Switches

Switch bounce from a single depress/release of mega128 pushbutton switches









Debouncing Switches

The problem is that the uC is usually fast enough to see all the transitions
- uC acts on multiple transitions instead of a single one

The oscilloscope traces showed bounce durations of 10-300us
- our mega128 uC runs at 62.5ns per instruction
- a 10uS bounce (short) is ($1x10^{-5}/62.5x10^{-9}$) 160 instructions long!
- a 100uS bounce could be sampled as a valid true or false 100's of times
- results are incorrect behavior as seen by user

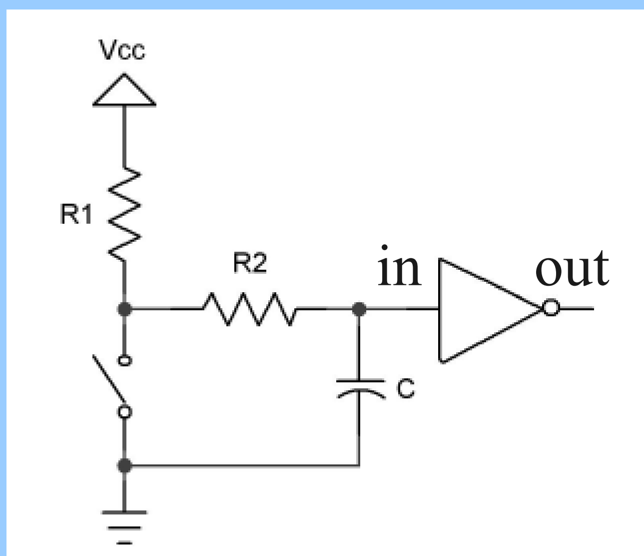
Characteristics of switch bounce:
- nearly all switches do it
- the duration of bouncing and the period of each bounce varies
- switches of exactly the same type bounce differently
- bounce differs depending on user force and speed
- typical bounce frequency is .1-5ms

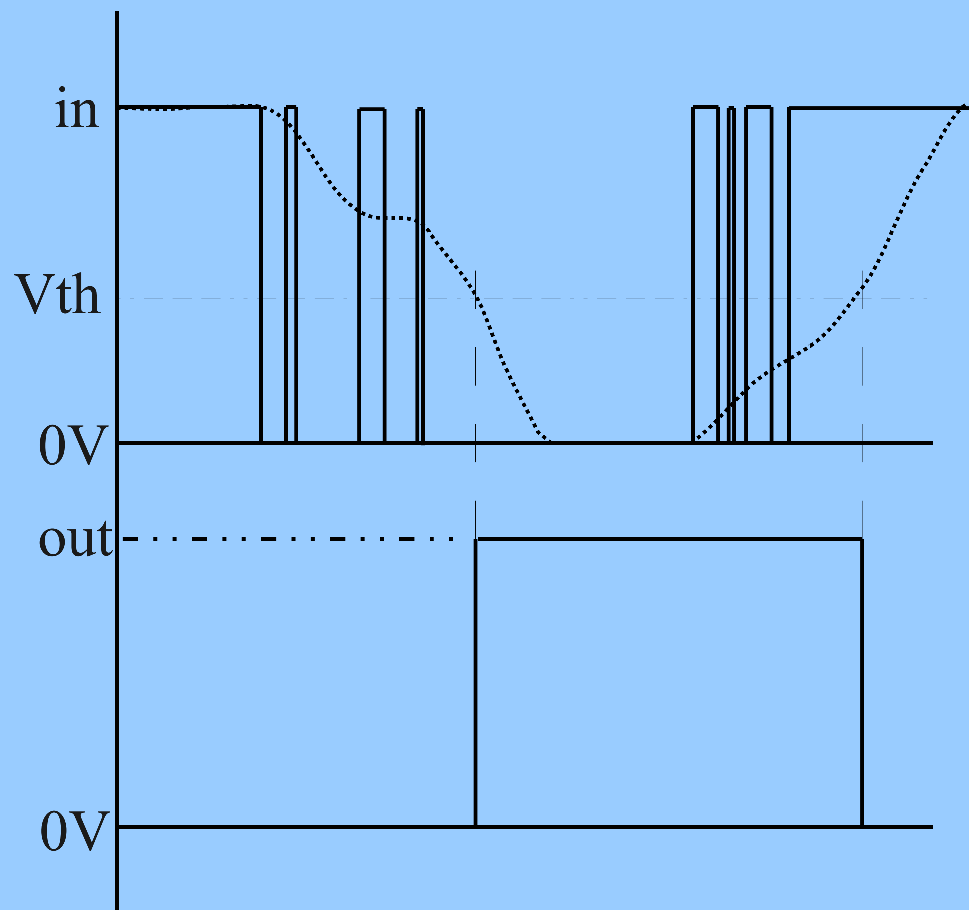
Effective switch debouncing can also reject EMI and static charge effects
- EMI can be periodic (so don't sample synchronously!)
- false triggering on static electricity is like a single random input

Debouncing Switches

Solutions

- Analog filtering
  - usually an RC delay to filter out the rapid changes in switch output
  - task is to choose R and C such that the input threshold is not crossed while bouncing is still occurring

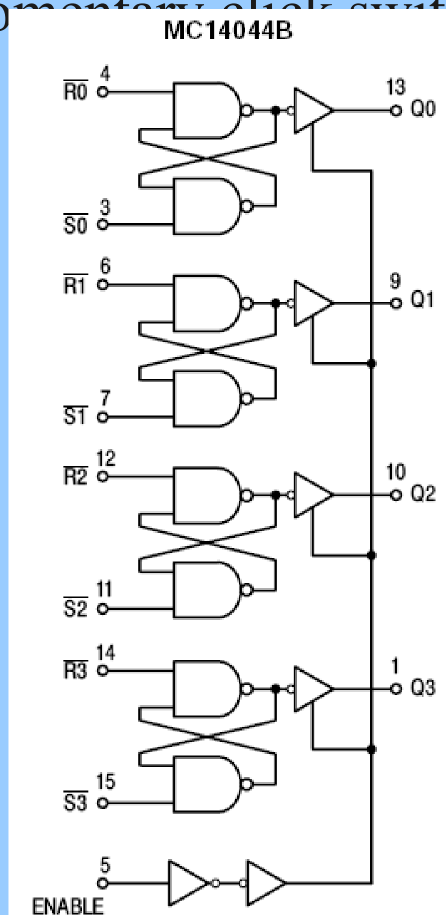


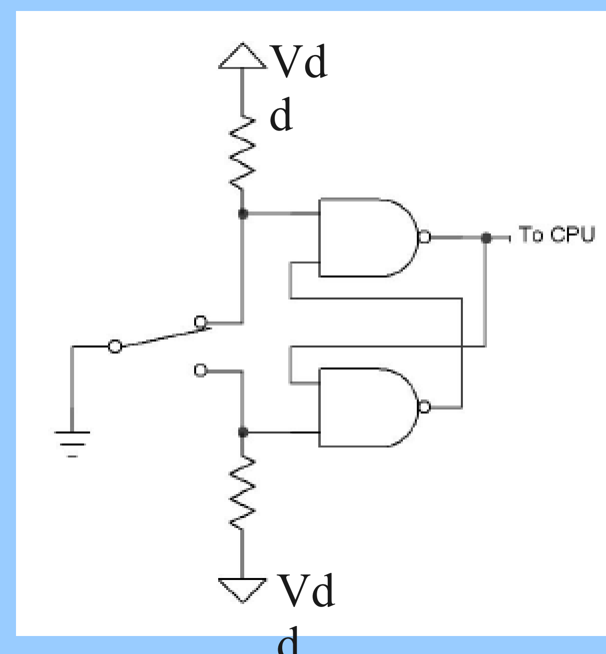


Debouncing Switches

Solutions

- Cross coupled gates (MC14044)
  - logic gates lock in one transition with a single-pole, double-throw switch
  - both switch ($3.69) and chip ($0.38) are expensive
  - momentary click switches (mega128 board) are ($0.12)





Debouncing Switches

Solutions:

- Software
  - need to minimize CPU usage
  - independent of clock speed
  - do not connect to interrupt pins, only programmed I/O
    - multiple interrupts will tie up processor
  - don't scan synchronously to noisy devices
  - identify initial switch closure quickly (100mS max)

- Two approaches (of many)
  - Count based

    risky!

    (identify initial closure AND wait AND check for same value) OR

    (identify initial closure AND check for same value for X cycles)

    safer!

    - watch for
      - CPU speed dependencies (use constant defines)
      - loop execution times (suggesting the use of timer interrupts)
  - Digital filter based
    - mimics an analog filter with first-order recursive low pass filter
    - includes a software schmitt trigger
    - good EMI filtering, quick response

Debouncing Switches

Solutions:

-Count based

-from Gansel's “Guide to Debouncing”

-routine called from timer interrupt or delay loop

-check Port D, bit 2, pushbutton depression, (bit will be grounded)

-returns “1' only once per button push, “*pulsed output*”

-looks for a falling edge

```
int8_t  debounce_switch()  {
  static uint16_t  state = 0;  //holds present state
  state = (state << 1) | (! bit_is_clear(PIND, 2)) | 0xE000;
  if (state == 0xF000) return 1;
  return 0;
   }
```

	value of `state`	
first pass after reset:	1110 0000 0000 0001	return 0
second pass after reset:	1110 0000 0000 0011	return 0
after 12 false passes:	1111 1111 1111 1111	return 0
after 7 true passes:	1111 1111 1000 0000	return 0
after 12 true passes:	1111 0000 0000 0000	return 1
after many true passes:	1110 0000 0000 0000	return 0
after 5 false passes:	1110 0000 0001 1111	return 0

Debouncing Switches

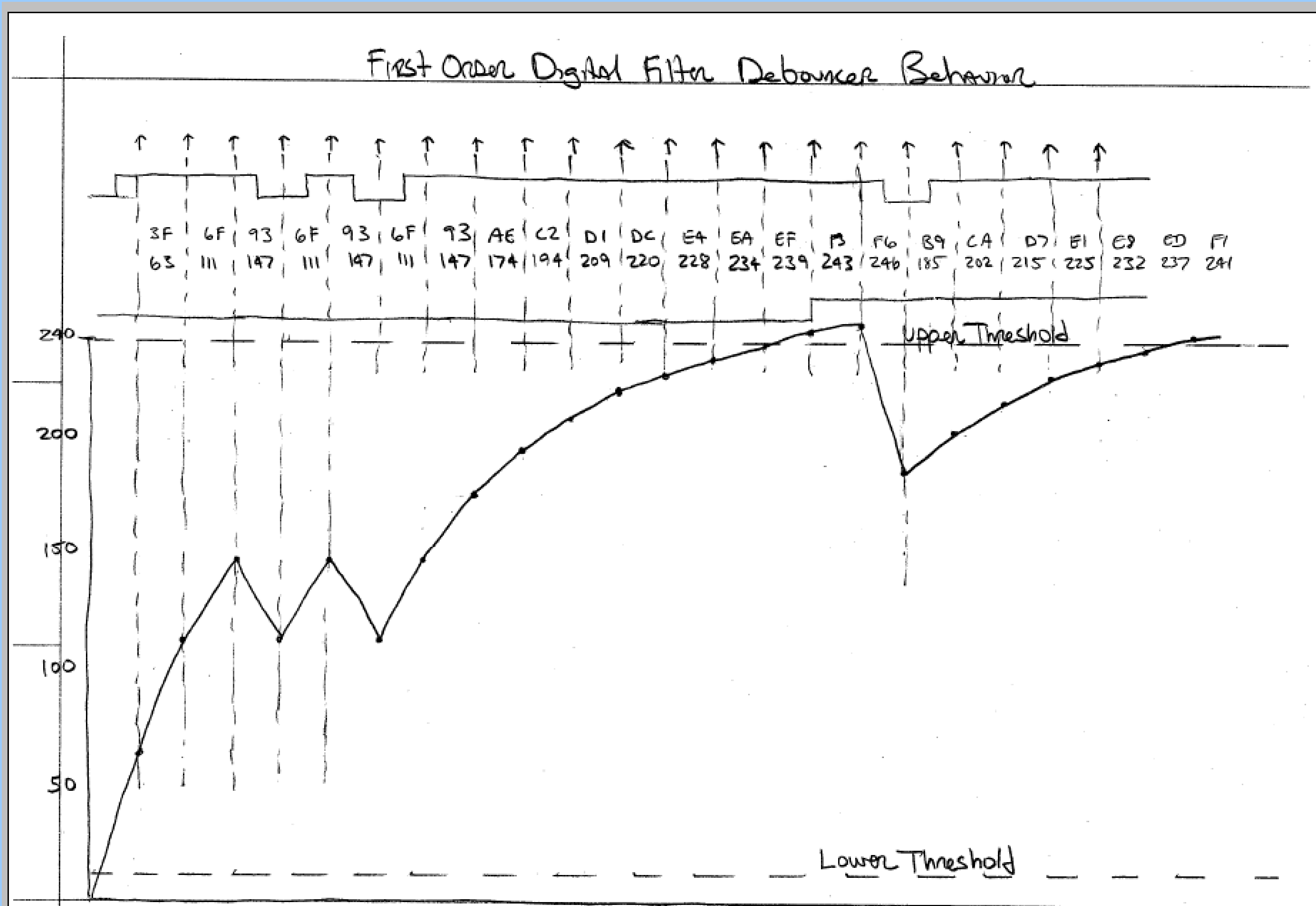
Solutions:

- Digital filter based
  - acts like analog RC filter followed by schmitt trigger
  - nearly continuous output like an analog circuit
  - 0.25= 0x3F, 0.75=0xC0, 1.0 = 0xFF

```
uint8_t output=0;  //external variable indicating switch state

uint8_t  debounce_switch2()  {
  static uint8_t  y_old=0, flag=0;
  uint8_t temp;

//digital filter part  y_old = x_new*0.25 + y_old*0.75
  temp = (y_old >> 2);    //this gives y_old/4
  y_old = y_old – temp;  //do (y_old*0.75) by subtraction

//if button is pushed, add 0.25 (3F) of new value (1.0)
  if(bit_is_clear(PIND, 2)){y_old = y_old + 0x3F;}  //

//software schmitt trigger
  if((y_old > 0xF0)&&(flag==0)){flag=1; output=1;}
  if((y_old < 0x0F)&&(flag==1)){flag=0; output=0;}
}
```

Debouncing Switches

Behavior of the digital filter debounce with schmitt trigger



Using Others Code

I just showed you several code snippets that you can use.

You should understand how they work.

You *may* (wink, wink) probably be tested on them in the future.

I offer other pieces of code in the coming weeks.

Learn what they do before you use them.

In general, don't use code from somewhere else without knowing how it works.

An exception would be a well used, well tested library of functions (libc).

Debouncing Switches

Types of debouncer output

Sometimes we want a *continuous* output, e.g., organ keyboard.

Other times we want a *pulsed* output, e.g. increment hour alarm.

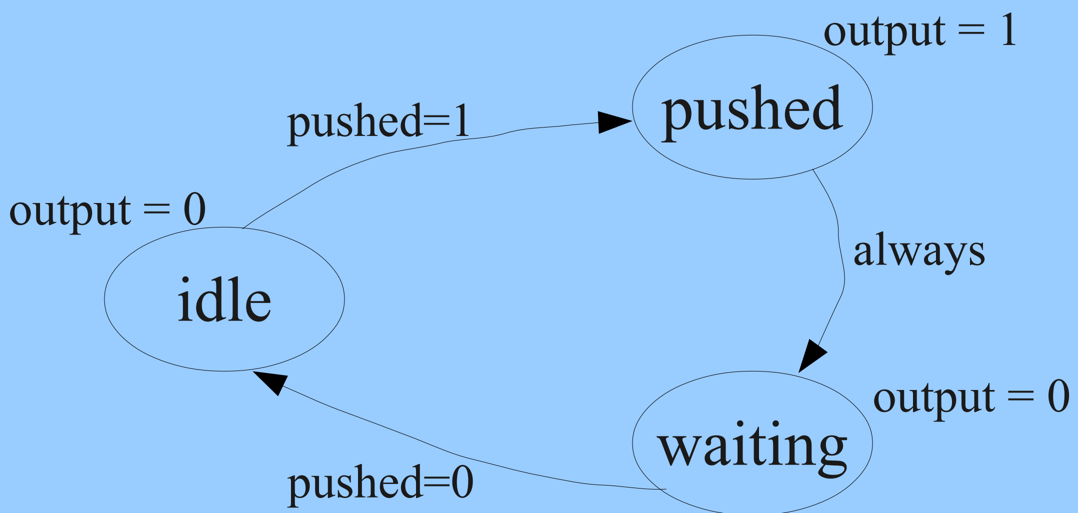
The first counting algorithm (*Gansel*) gives a pulsed output.

The digital filter algorithm gives a continuous output.

Debouncing Switches

Converting between types of debouncer output

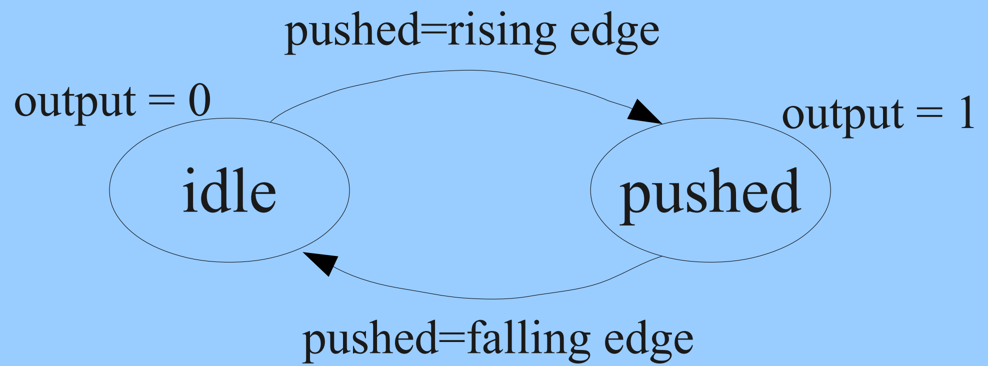
To get a *pulsed* output, from a continuous debouncer:



Debouncing Switches

Converting between types of debouncer output

To get a *continuous* output, from a pulsed output:



Note that this requires sensing rising and falling edges.