MSP430 Advanced Technical Conference 2006

Enabling Single Touch Capacitive Sensing with MSP430

Zack Albus
MSP430 Applications Engineer
Texas Instruments

Agenda

- Overview of Single Touch Applications
- System-Level Careabouts
- MSP430 Implementations
- Keys, Sliders & Demos
- Summary

Applications of Touch Sensing

- **Alternative to mechanical switches**
  - Low cost
  - Longer life
- **Flexible user interface**
  - Simple buttons
  - Sliders
- **Adaptable**

- **Useful in…**
  - Consumer electronics
  - Appliances
  - Residential control
- **… and almost anywhere a switch is currently used**

Touch Sensing Overview

- **Different technologies**
  - Optical, Resistive, Capacitive, Strain,…
- **All detect change in system**
- **Optical**
  - Expensive
  - Complex system design
- **Resistive**
  - Require sensor material that changes R when touched
  - Relatively low cost, but is an additional element to the BOM
- **Capacitive**
  - Can be implemented on PCB directly
  - Flexible sensor size & shape
  - Cost is a function of the PCB and any externals needed

Capacitive Methods

- **Charge transfer technology**
  - Quantum Research Group patented solution
  - Fixed function ICs that measure charge transfer from one sensor C to another
  - Stimulus signal and measurement integrator
- **Capacitive measurement via ADC**
  - Stimulus signal impacts capacitive sensor element, resulting voltage is measured by ADC
  - ADI implementation using a 16-bit Sigma-Delta to perform C-to-Digital conversion
- **Relaxation Oscillator**
  - Creates oscillator dependent on sensor C variation & measures frequency
- **RC Charge/Discharge**
  - Using high frequency clock, times charge and/or discharge times for sensor element with varying C

MSP430 Capacitance Measurement

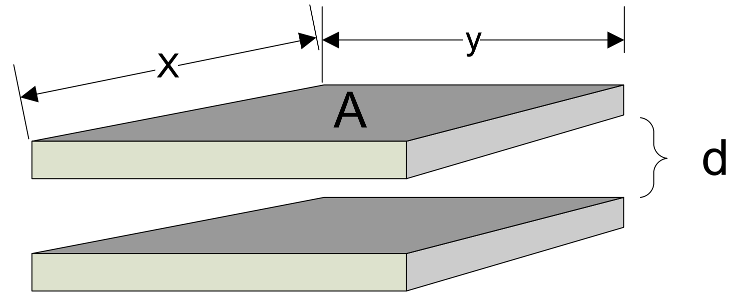
- **Change in capacitance due to physical proximity of a finger or other conductive object**
- **Method 1:**
  - Create oscillator dependent on capacitance of the sensing element
  - Measure freq change when sensor C is changed by touch
- **Method 2:**
  - Measure R-C charge/discharge where R is constant and the sensor element capacitance changes due to touch

Agenda

- Overview of Single Touch Applications
- System-Level Careabouts
- MSP430 Implementations
- Keys, Sliders & Demos
- Summary

Capacitive Fundamentals

- **Base capacitance created by PCB mechanics**
- **Capacitance change due to changing parasitics**
  - Finger touch proximity (or conductive other source)
- **Minimize base capacitance**
  - Limit parasitics
  - Limit sensor size
- **Maximize impact of change**
  - Match sensor & finger areas for greatest delta-C
  - Minimize distance between sensor and finger
- **Sensitivity**

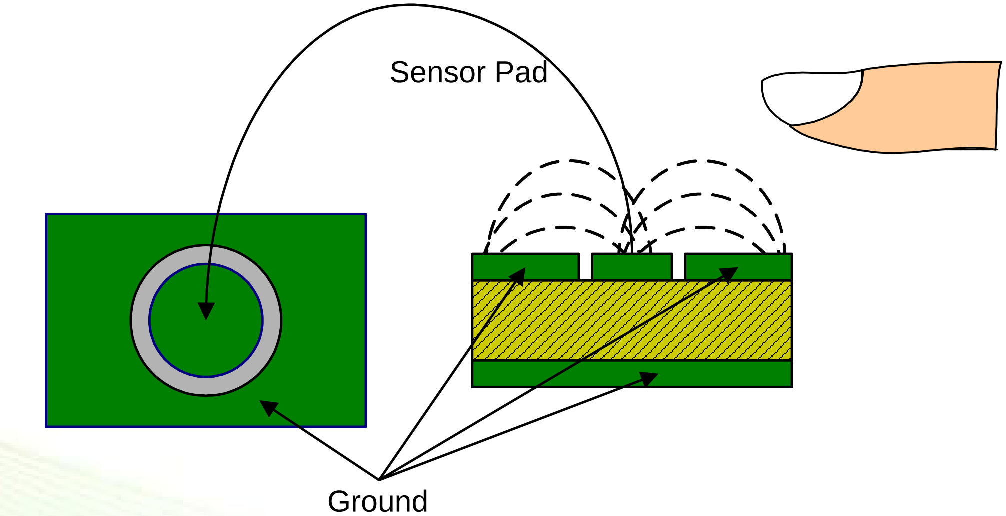


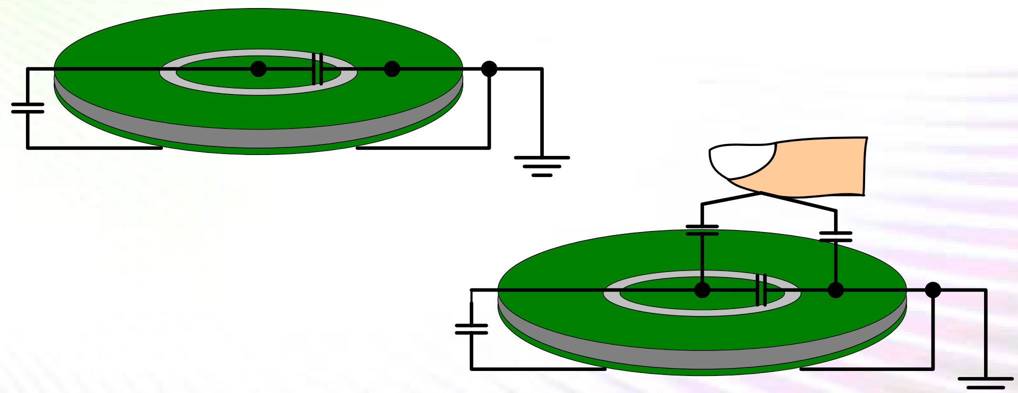
$$C = \frac{\varepsilon_0 \varepsilon_r A}{d}$$

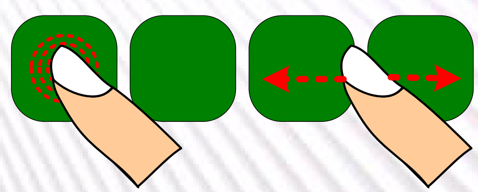
Material	***Dielectric Constant*** (ε_r)
Vacuum	**1 (by definition)**
Air	**1.00054**
Polyethylene	**2.25**
Paper	**3.5**
Pyrex glass	**4.7**
Rubber	**7**
Silicon	**11.68**

Capacitive PCB Sensor

- Copper pour on PCB makes a good sensor element
- ~10-20mil spacing between sensor & adjacent elements
- Size pads to maximize finger overlap for max delta C
- Simple pads can also be good sliders
- For true sliders, sizing pads such that more than one is touched at a time helps determine position



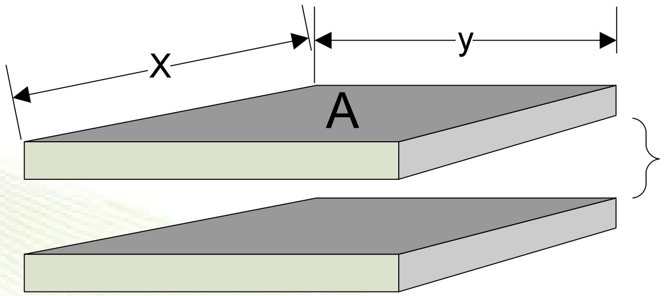




PCB Thickness

- **Material and thickness matters**
  - Goal 1: Small base C
  - Goal 2: Stable base C

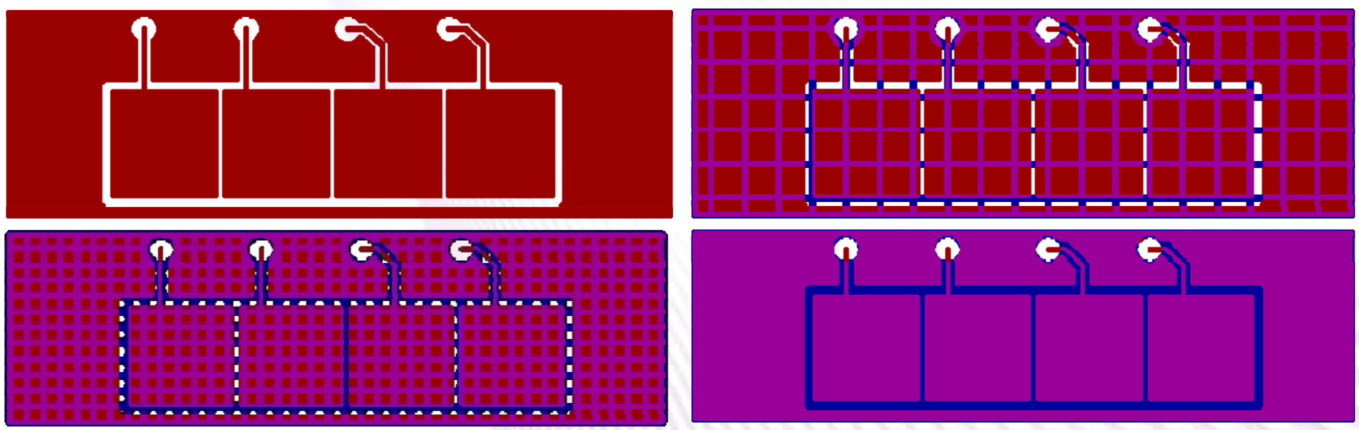
$$C = \frac{\varepsilon_0 \varepsilon_r A}{d}$$



- **As d decreases, the base capacitance increase**
- **For a given sensor size and insulator thickness, the delta C created by a touch is fairly constant**
- **This change is a smaller percentage of the base C as d goes down**
- **Thinner PCBs require more care in insulator selection and thickness**

Layout & Grounding

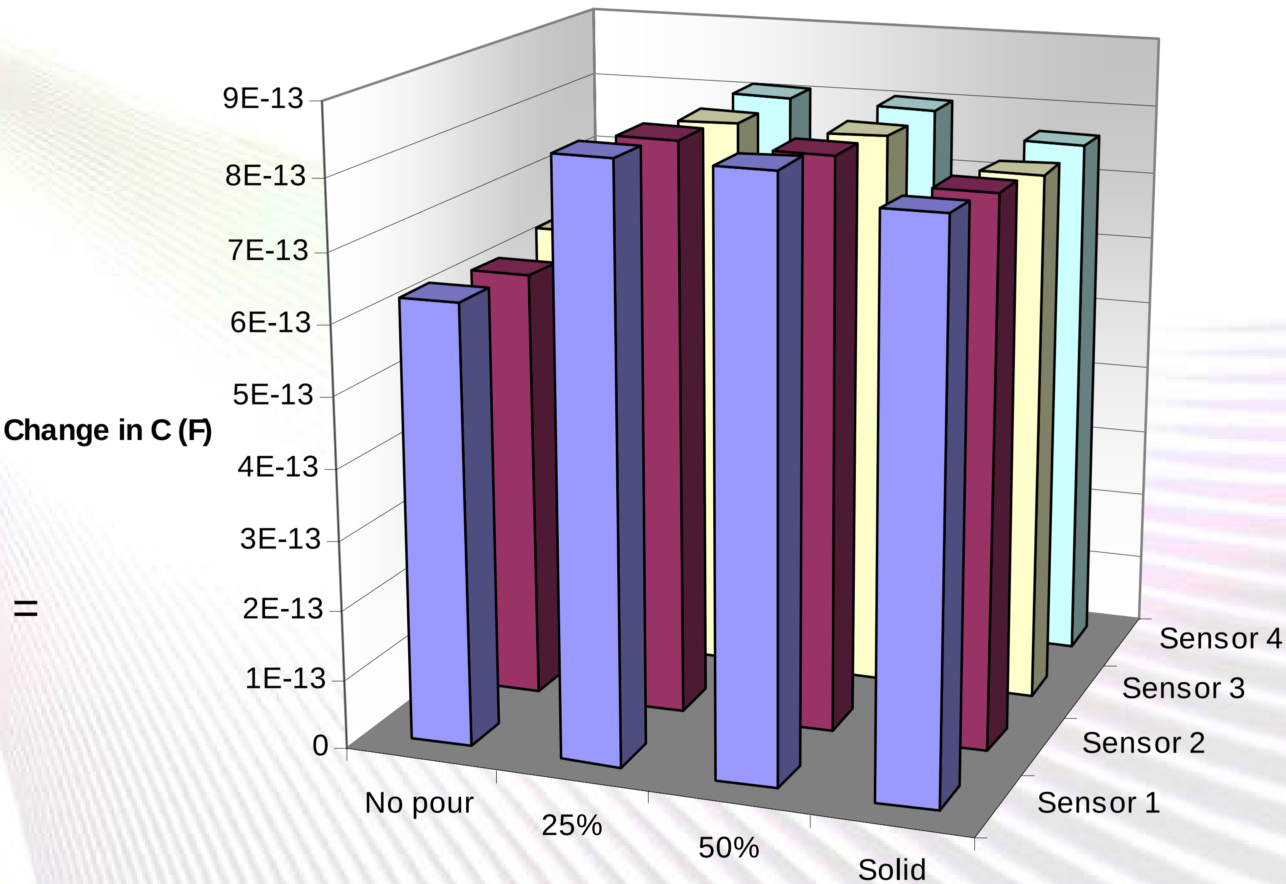
- **Minimize noise & signal coupling with solid ground pour on sensor side of PCB**
- **Hatch pour underneath sensors if possible**
  - Solid pour ok for noise, but increases base capacitance (larger A)
  - No pour has no increase in base capacitance but no noise benefits
  - A hatch of 50% is a good compromise



Sensors & Ground Influence

- **Tradeoff between PCB ground pour under sensors and sensitivity**
- **No Pour**
  - Low base C
  - Small delta C
- **25-75%**
  - Base C increases
  - Larger delta C
- **Solid Pour**
  - Large base C
  - Harder to influence change = lower delta C

Delta C vs. Pour
(8x8mm sensor on 1.5mm FR4)

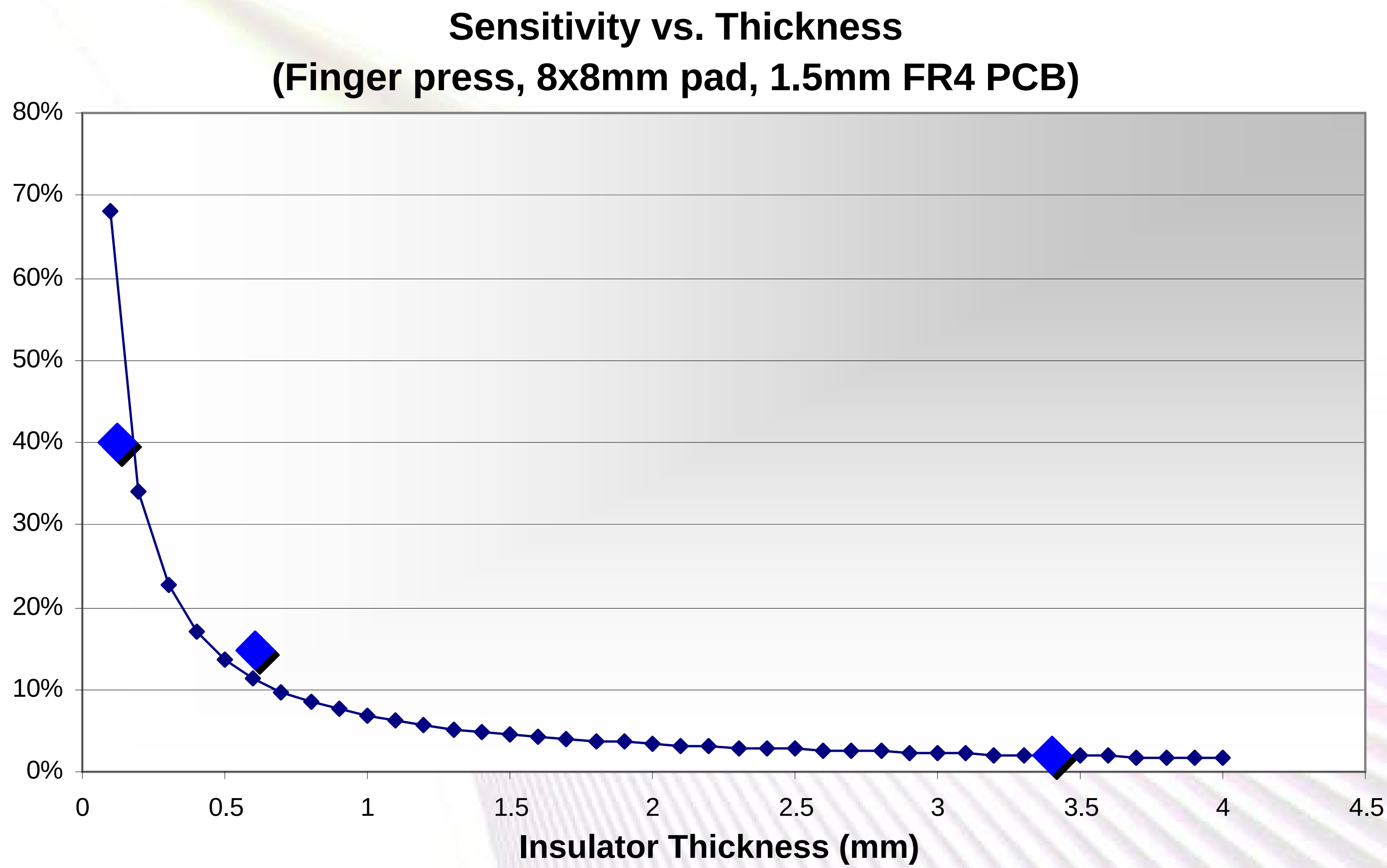


Insulators & Assembly

- **An insulator is usually needed between PCB and user**
- **Insulator material must be non-conductive**
- **Thin is better**
  - C is inversely proportional to the distance between the conductors
- **No air should be present between insulator and the sensors on the PCB**
  - C is proportional to the dielectric constant
- **Use adhesives to secure sensor and insulator**
  - Nonconductive adhesives, air-free
  - Those which tolerate temperature and humidity changes well are recommended

Insulator Spacing

- Achievable sensitivity is inversely proportional to insulator thickness

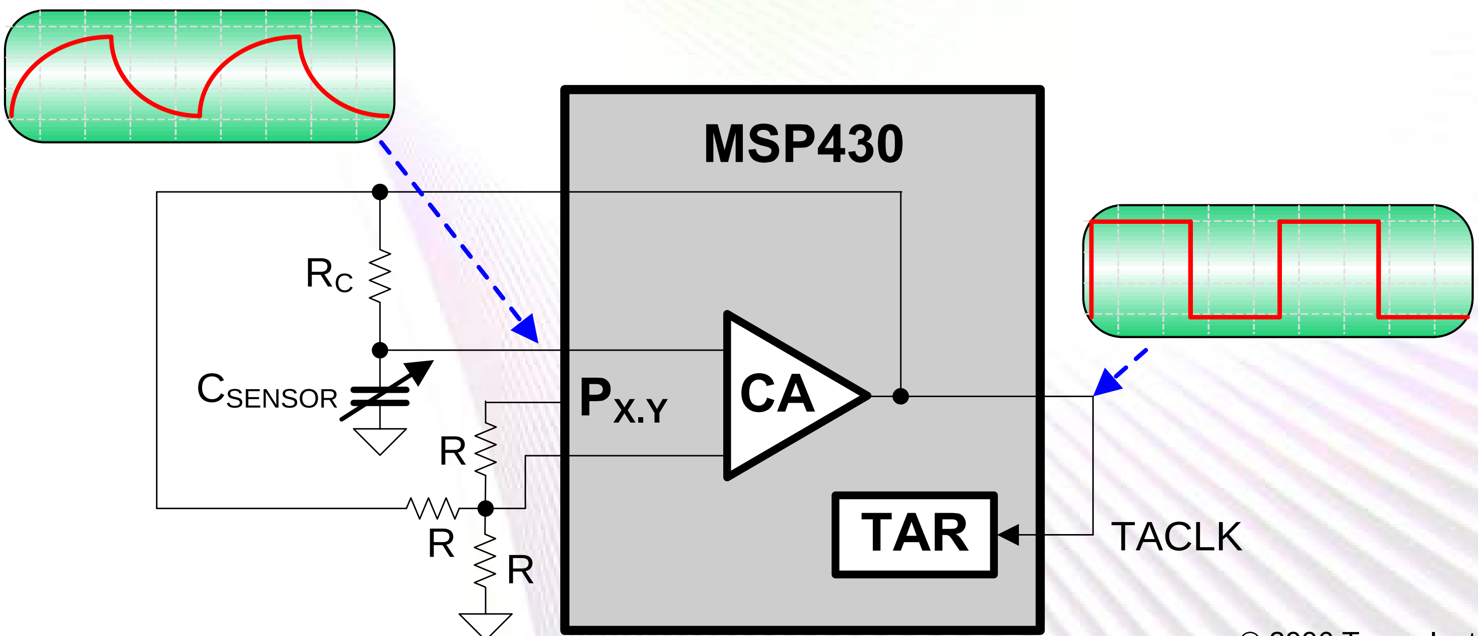


Agenda

- Overview of Single Touch Applications
- System-Level Careabouts
- MSP430 Implementations
- Keys, Sliders & Demos
- Summary

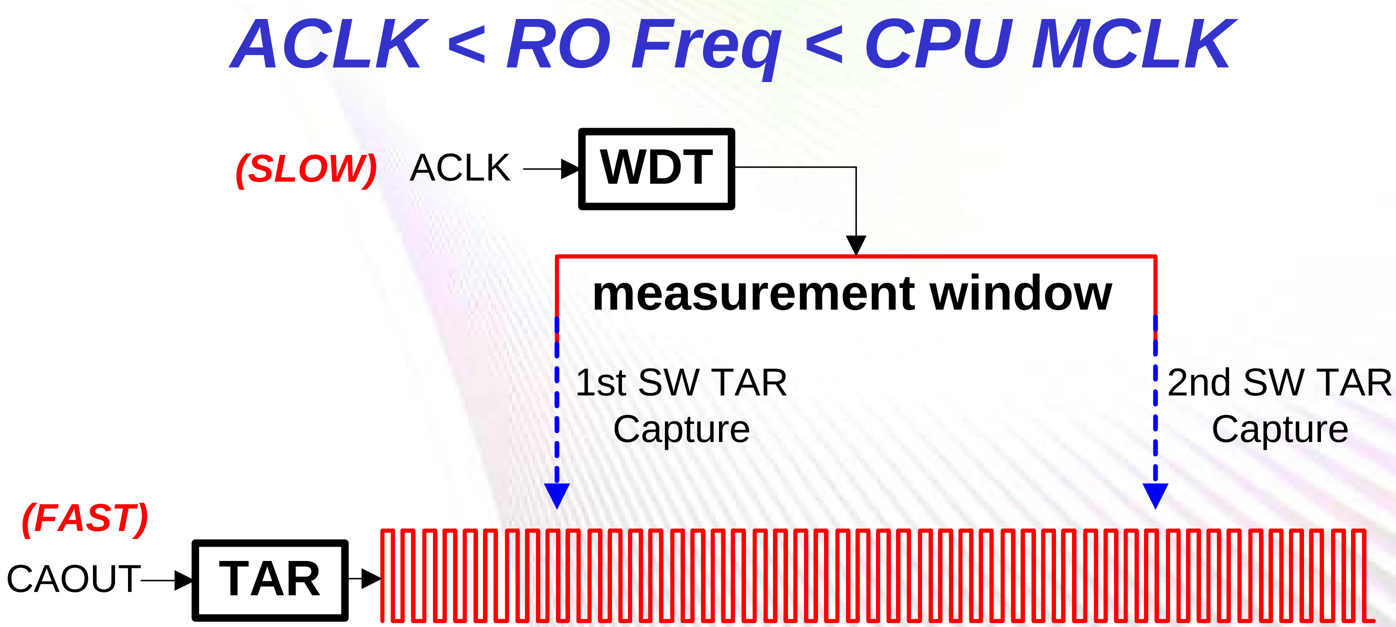
RO System Overview

- Osc created using comparator with frequency a function of sensor capacitance
- Charge/discharge limits set by R's (1/3 Vcc & 2/3 Vcc)
- Freq is 1/[1.386 x R_c x C_sensor]
- delta C => delta f



RO Frequency Measurement

- Slow interrupt defines window for measurement
- Faster RO periods are counted via Timer_A
- CPU clock speed used to eliminate ISR s/w capture latency error



Measurement Relationships

- **Usable counts increase with measurement time**
- **Using VLO/64 for ACLK & DCO_cal/32768 for SMCLK**
  - (100K R ~ 625kHz f_RO)

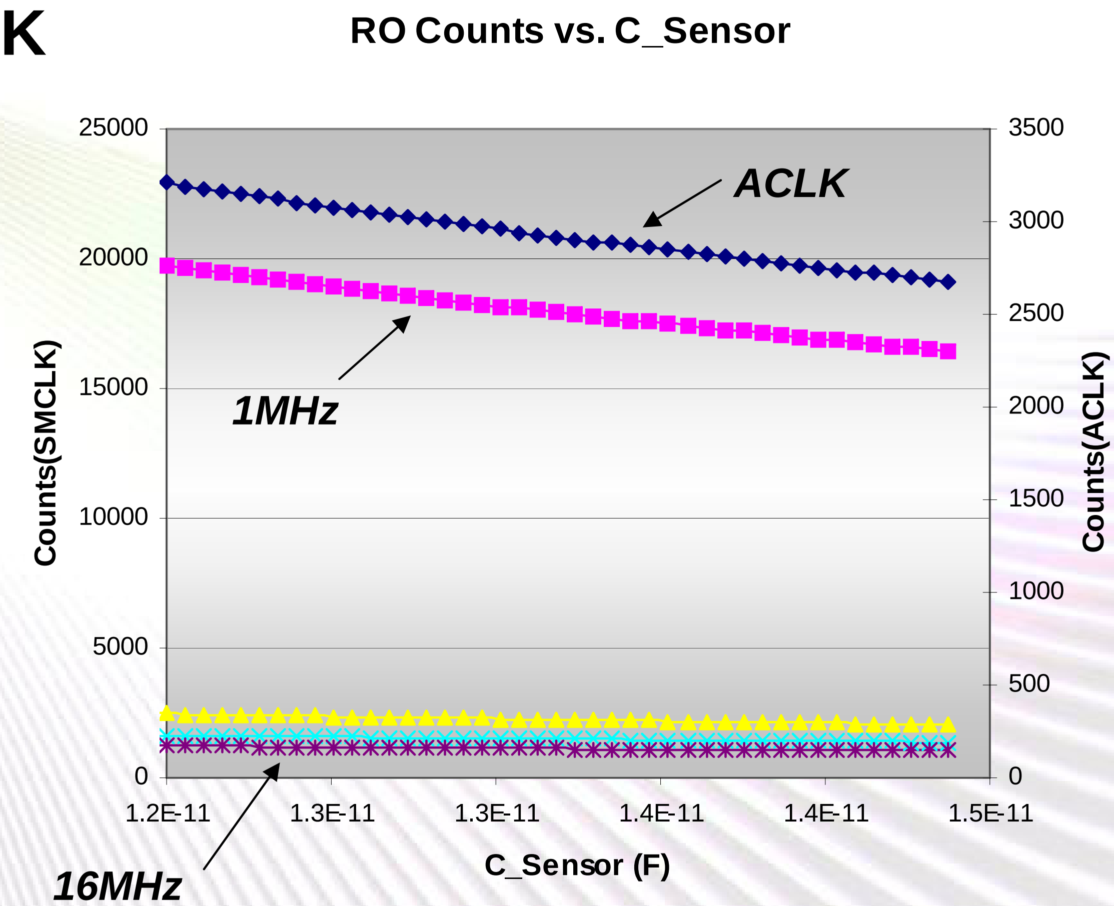
$$f_{RO} = \frac{1}{1.386 \times R \times C}, t_{RO} = \frac{1}{f_{RO}}$$

$$t_{window} = \frac{1}{f_{ACLK} / DIV_{ACLK} / DIV_{WDT}}$$

$or...$

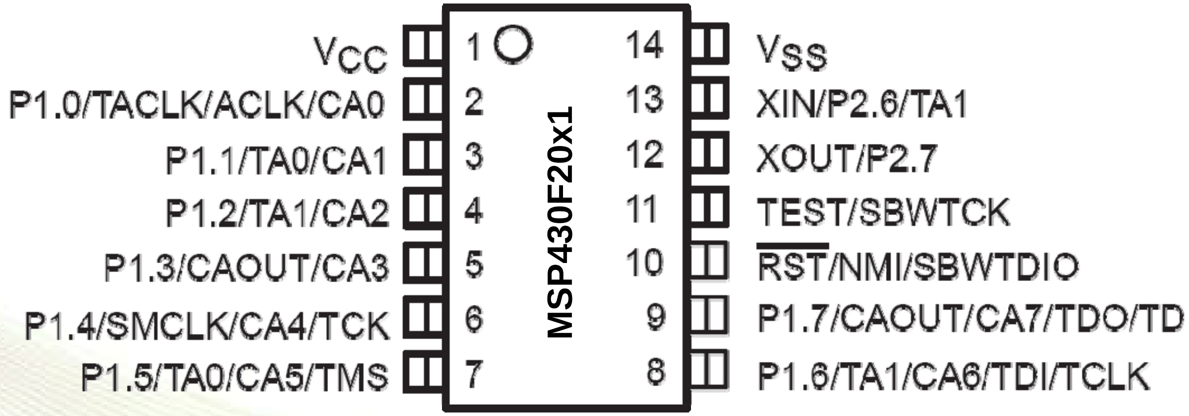
$$t_{window} = \frac{1}{f_{DCO} / DIV_{SMCLK} / DIV_{WDT}}$$

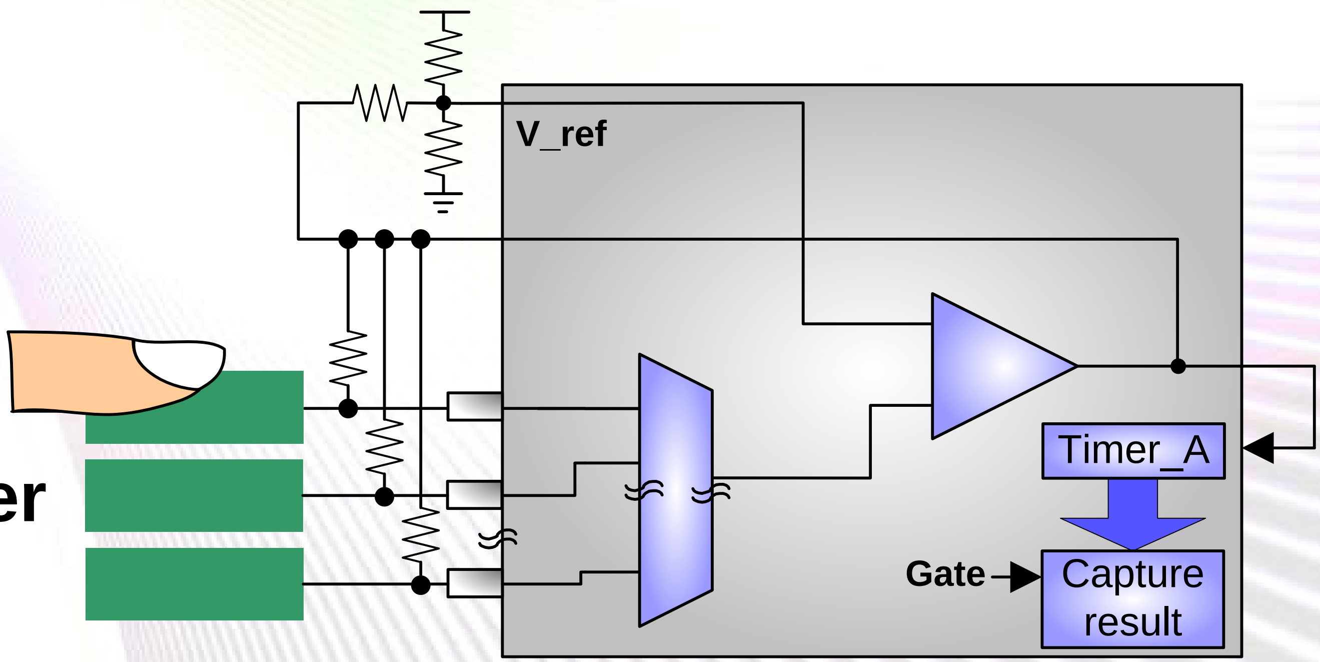
$$counts = \frac{t_{window}}{t_{RO}}$$



Complete RO System

- Requires Comp_A+ (needs mux input for multiple sensors)
- One external R per sensor, three for reference feedback
- External connection to TACLK
- Power Vref ladder via port pin for ULP

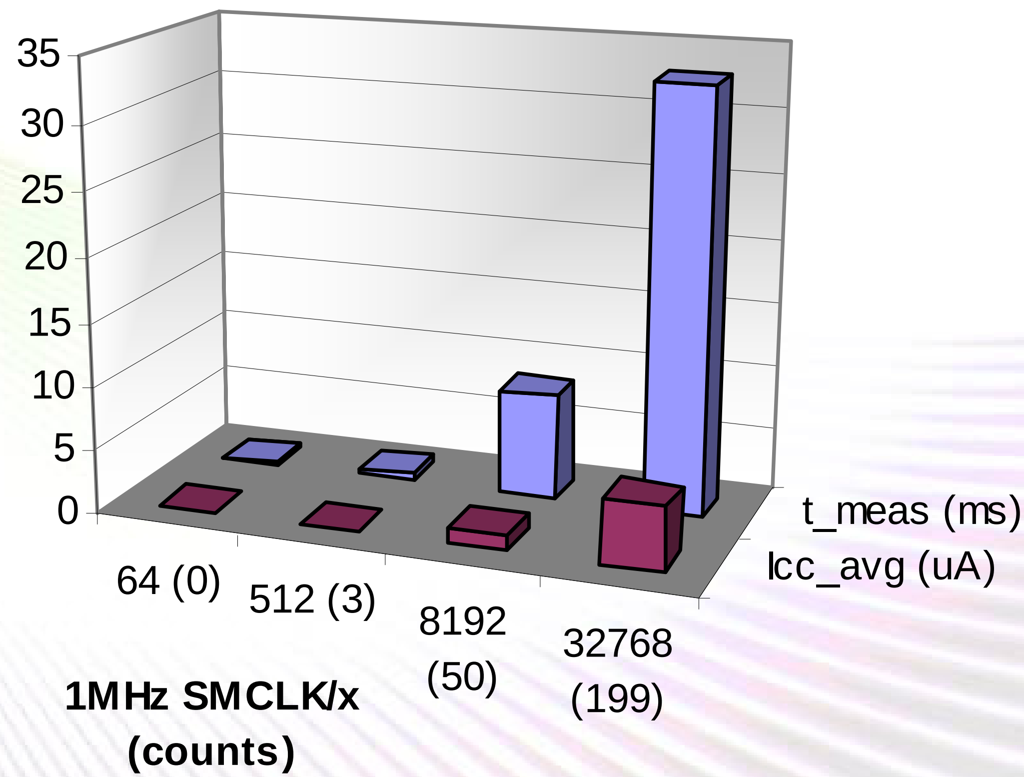




RO Current Consumption

- **Longer t_measure = more counts**
- **Also means higher average Icc**
  - DCO: ~85uA @ 1MHz
  - Comp_A+: ~45uA
  - CA Vref: Vcc/(1.5R) (for 100k R's, ~20uA)
- **Define t_measure for adequate counts for application**
  - Bigger delta C, smaller t_measure can be used
  - Design to fewest counts needed for lowest current

Current & Measurement Time vs. Measurement Window (1% C_delta)

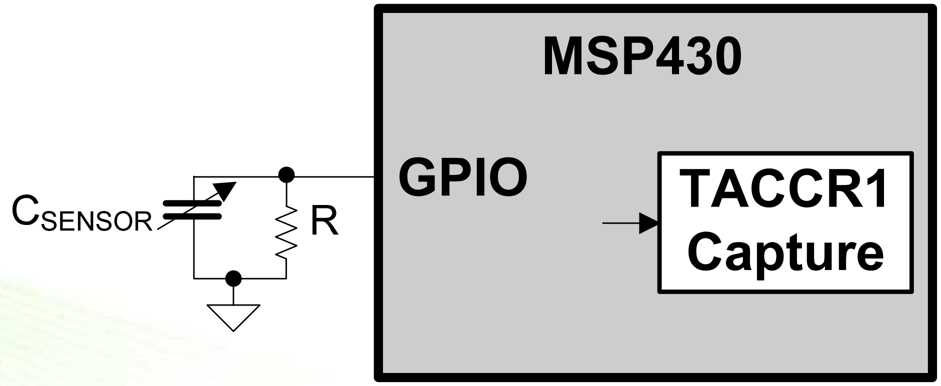


RO Tradeoffs

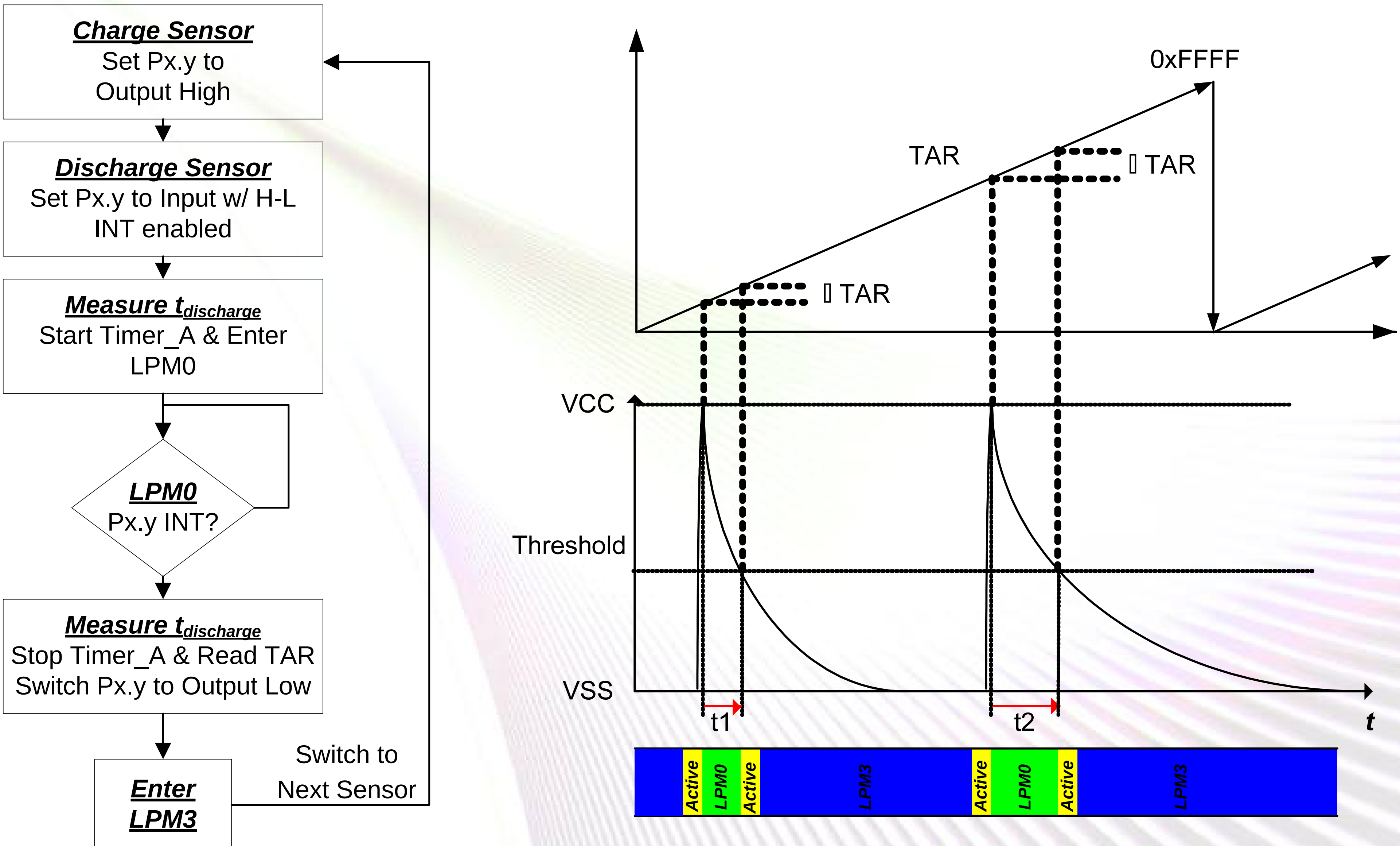
- Needs Comp_A+ input mux for multiple sensors
- Sensors used limited by usable CA+ mux inputs
- External R's needed to setup CA+ reference
- External CAOUT to TACLK required
- Good noise immunity: freq vs. voltage
- Programmable measurement time
- No high speed clock needed
- Measurement time dependent influenced by Vcc & Temp (VLO & DCO)

RC System Overview

- **RC discharge time measured using interrupt on GPIO**
- **P1.x/P2.x GPIOs used**
- **Port pin used to charge sensor cap and measure discharge time**
  - GPIO = Output high (charge C)
  - GPIO = Input (discharge C)
  - GPIO INT on low threshold
- **Timer_A used to measure discharge time of C_sensor**



RO Measurement Cycle



Measurement Relationships

- **Usable counts increase with increased reference clock**
- **Using ACLK = VLO & SMCLK = DCO_cal**
  - 5.1Mohm R

$$V(t_{rc}) = Vcc \times e^{\frac{-t}{RC}}, V(t_{rc}) = Vcc \times [1 - e^{\frac{-t}{RC}}]$$

$$V_{IT-} = Vcc \times e^{\frac{-t_{\text{discharge}}}{RC}}, V_{IT+} = Vcc \times [1 - e^{\frac{-t_{\text{charge}}}{RC}}]$$

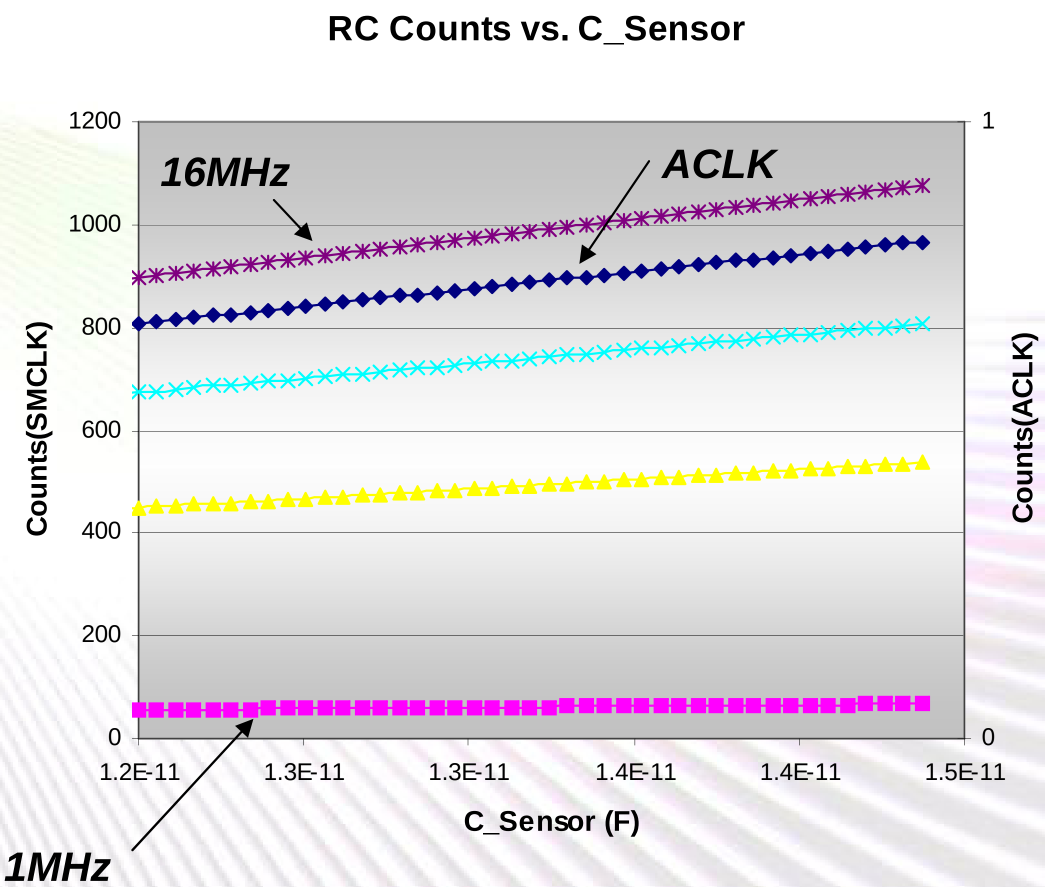
$$V_{IT-} = 0.4 \times Vcc, V_{IT+} = 0.6 \times Vcc$$

$$t_{\text{discharge}} = -RC \times \ln(0.4), t_{\text{charge}} = -RC \times \ln(1 - 0.6)$$

$$t_{CLK} = \frac{1}{f_{DCO} / DIV_{SMCLK}}$$

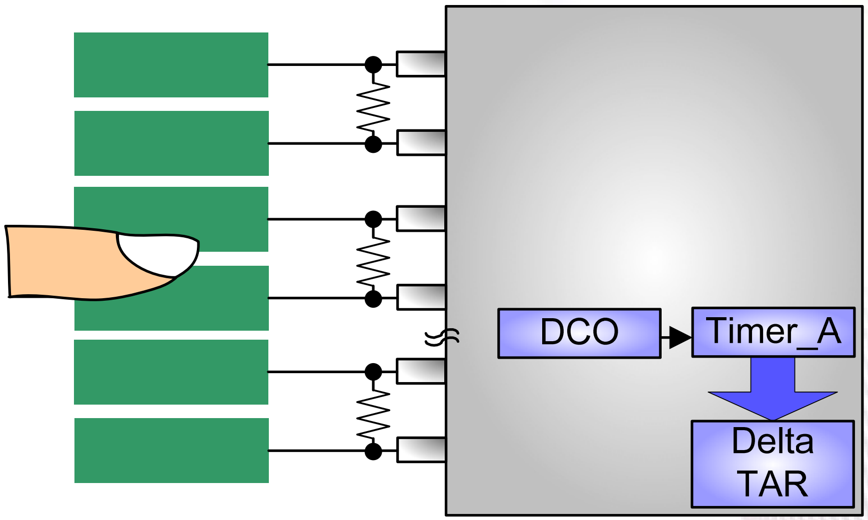
$$counts_{\text{discharge}} = \frac{t_{\text{discharge}}}{t_{CLK}}, counts_{\text{charge}} = \frac{t_{\text{charge}}}{t_{CLK}}$$

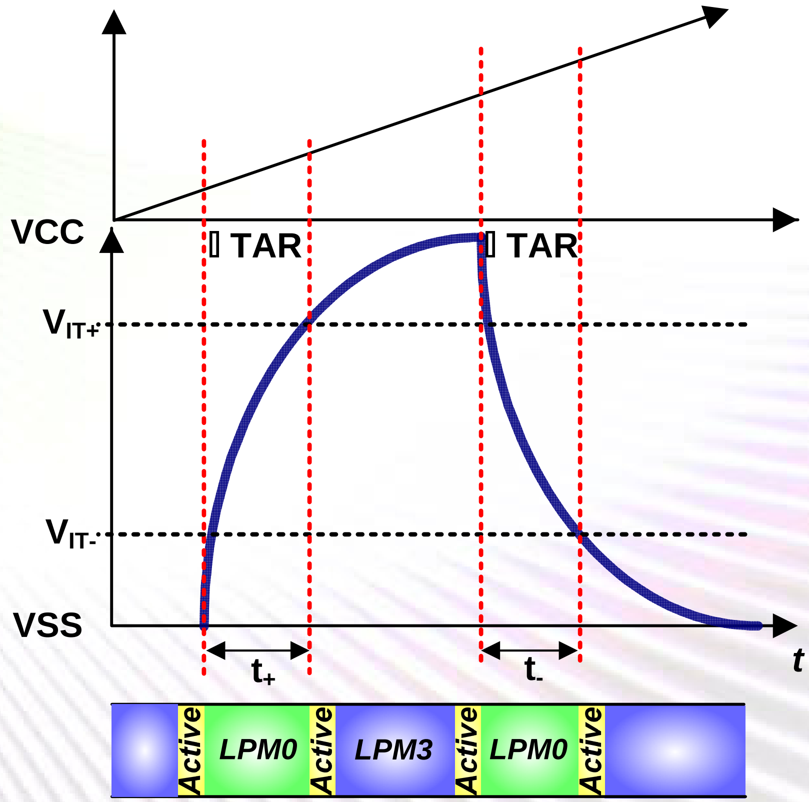
$$counts_{avg} = \frac{counts_{\text{discharge}} + counts_{\text{discharge}}}{2}$$



RC Optimizations

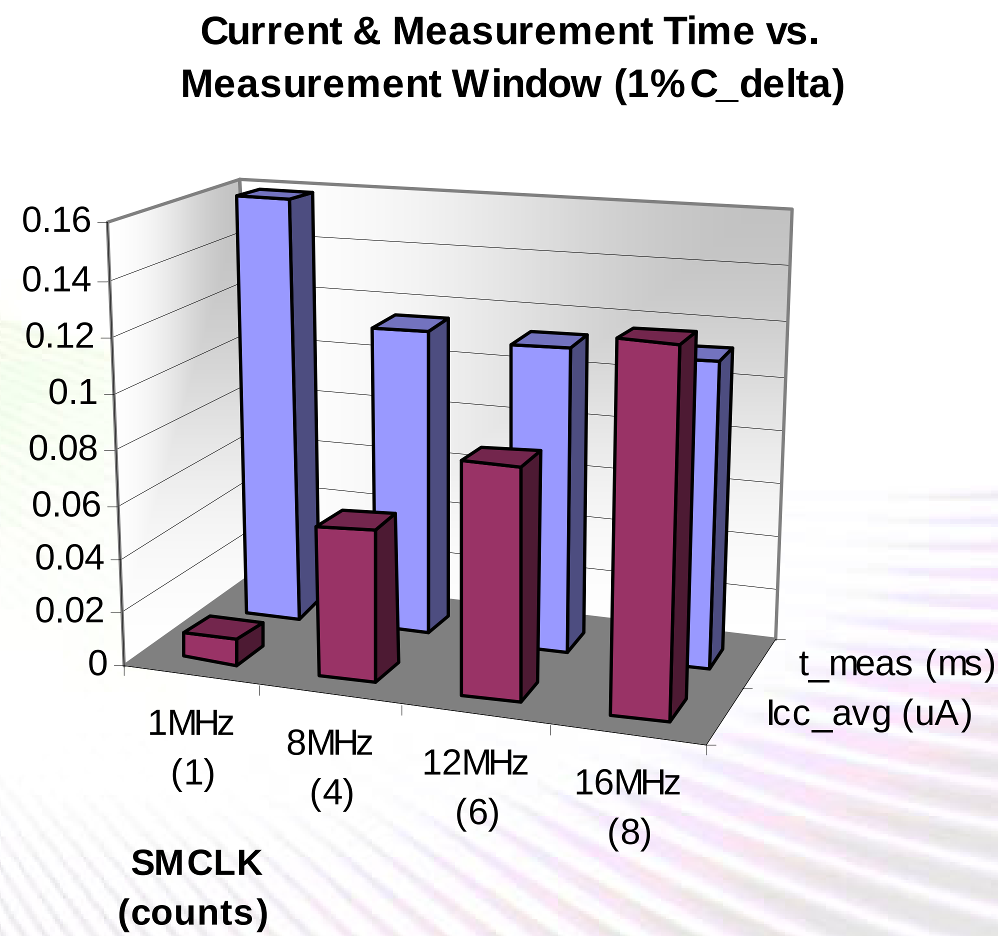
- **Two sensor elements can share a single R**
- **Each sensor can be charged, then discharged for an average result: better noise rejection**





RC Current Consumption

- **t_measure is constant: ~2*t_RC_charge**
  - R = 5.1Mohm
  - Counts TACLK
- **Average Icc depends on**
  - Tau = RC
  - DCO current consumption
- **Set TACLK for adequate counts for application**
  - Bigger delta C, lower f_DCO can be used
  - Design to fewest counts needed for lowest current



RC System Careabouts

- **Requires interrupt enabled GPIO for measurement**
- **One pin per sensor, shared resistor per two sensors**
- **R is Mohm's (5.1M)**
  - With pF C, large R required for a measurable charge/discharge time
- **Low pin leakage of MSP430 ideal for the methodology**
- **Noise rejection aided by charge/discharge average**
- **Measurement window is fixed by RC charge/discharge time: high freq reference clock needed to "count"**
- **Measurement counts dependent on Vcc & Temp (DCO)**

Agenda

- Overview of Single Touch Applications
- System-Level Careabouts
- MSP430 Implementations
- Keys, Sliders & Demos
- Summary

Touch Sensor Careabouts

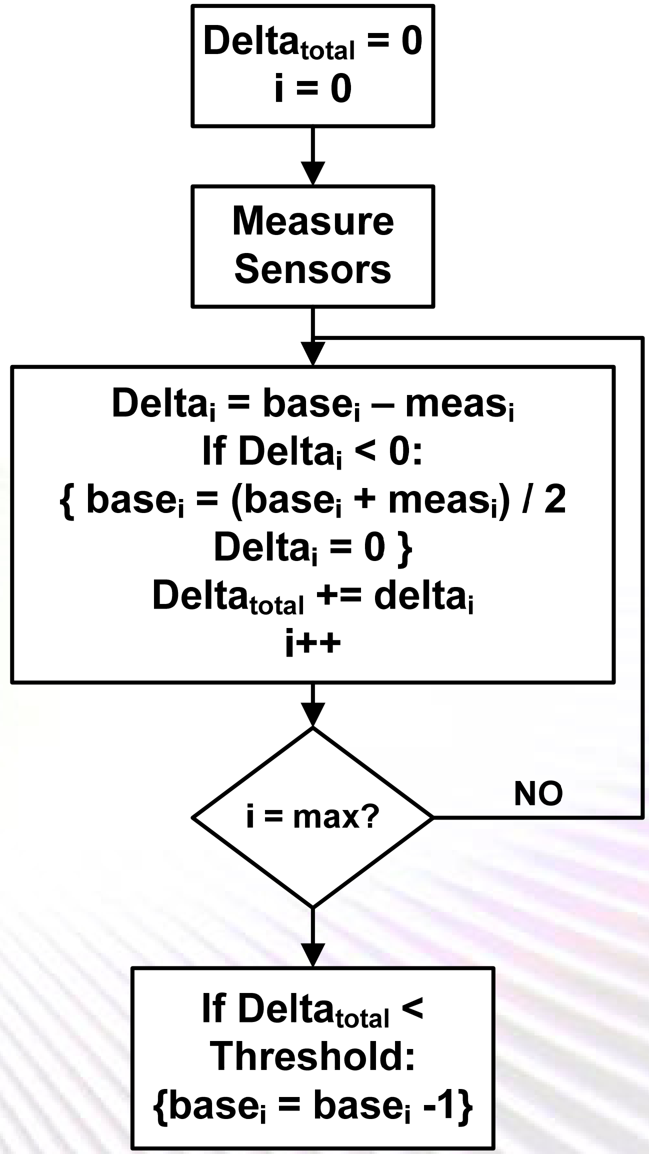
- **What is the application:**
  - A switch replacement?
  - Position detection? (e.g. slider)
- **Threshold: Establish a "usable" limit**
  - Can it be reached?
  - Enough noise margin?
  - Tolerant to manufacturing changes?
- **Filtering: Noise coupling**
  - Given large R in RC method, noise can easily couple in
  - Multi-result averaging: RC charge/discharge method
  - RO method inherently immune due to multiple cycles per measurement
- **Tracking: Baseline capacitance can shift**
  - Periodically adjust base capacitance count set-point
  - Take care to exclude a "touched" sensor result from any tracking algorithm

Tracking C_base

- **C_base measurement result can change over time**
  - Humidity effects
  - Temperature
  - Component tolerances
  - Voltage drift
- **Failure to track this change adequately can result in false key events or inability to detect events**
- **Algorithm basics:**
  - Adjust for a decreasing C rapidly, e.g. on each measurement, since this is not a function of sensor excitation
  - Adjust for increasing C very slowly as this may be due to a finger hovering over a key, not just C_base drift
  - Exclude an increasing C adjustment when any keys are pressed as it may be caused by the user, not C_base drift

Example: C_base Tracking

- **Adjust base result quickly when cap decreases**
  - Ex: re-average with current result
- **Adjust base result slowly when cap increases**
  - Ex: adjust by 1 with each measurement
  - Only adjust if no keys are pressed
- **Set “Threshold” level low enough that the sum of all key deltas will be greater if any key is press**
  - Alternatively, can adjust on per key basis
- **Note: sign of delta calc changes for the two methods**
  - RO: counts decrease when key excited
  - RC: counts increase when key excited

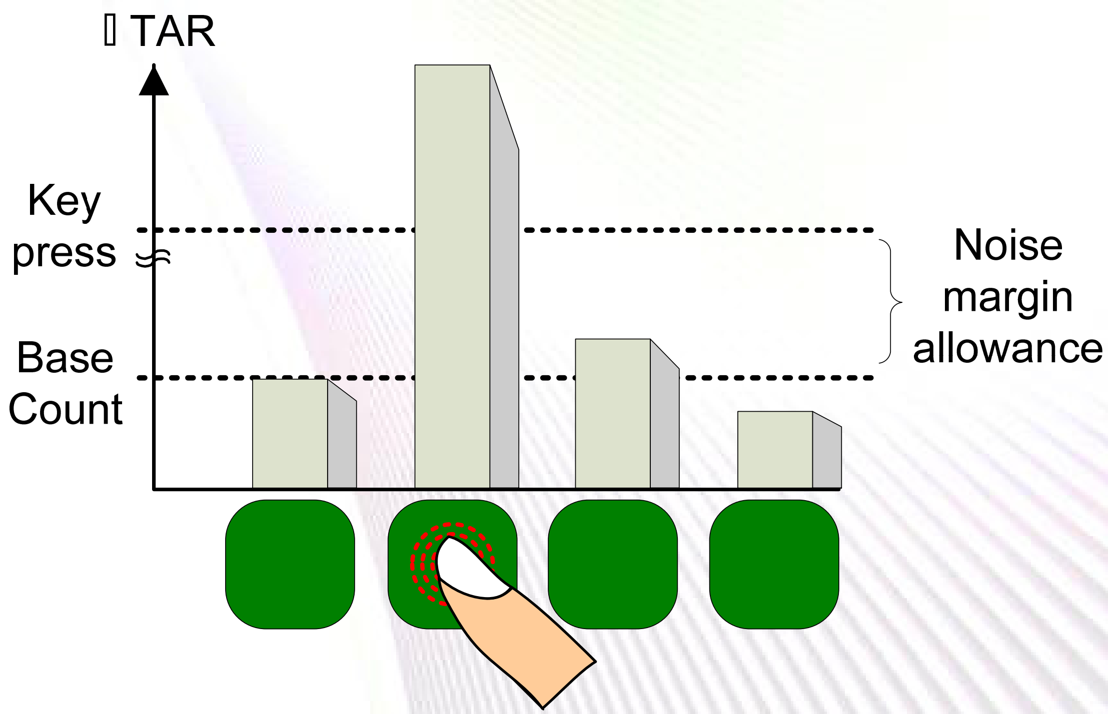


Data Filtering

- **Measurement results often noisy due to a number of factors including voltage supply**
- **When enough counts can be measured, simply throwing away the LSBs may be good enough**
  - Works ok for simple key press detection
- **A low pass filter of each key result will more adequately remove any unwanted noise and help stabilize the results, especially when measuring position on a slider**
- **Critical when counts are at a premium in the system due to constraints such as the PCB, insulator and power budget**

Key Press Detection

- **Measurement Flow**
  - Step 1: Establish a base count measurement
  - Step 2: Set a key press count threshold
  - Step 3: Scan keys
- **Set detection threshold ~50% of maximum count delta expected from the given implementation**



Key Pad Current Consumption

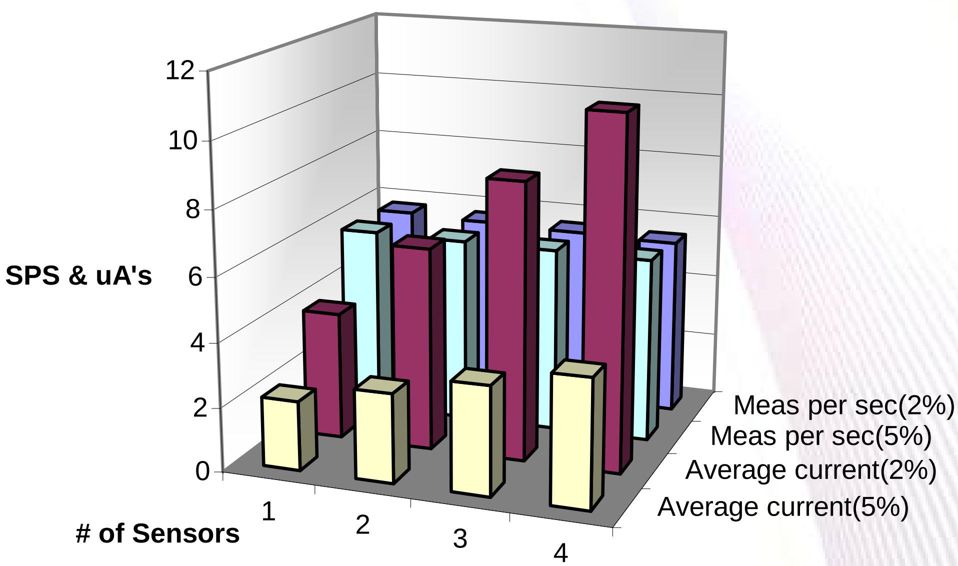
RO Method

- **Use smallest t_meas (lowest SMCLK) for needed counts**
  - ΔC 5% 1MHz, WDT= SMCLK/1/512
  - ΔC 2% 1MHz, WDT= SMCLK/4/512

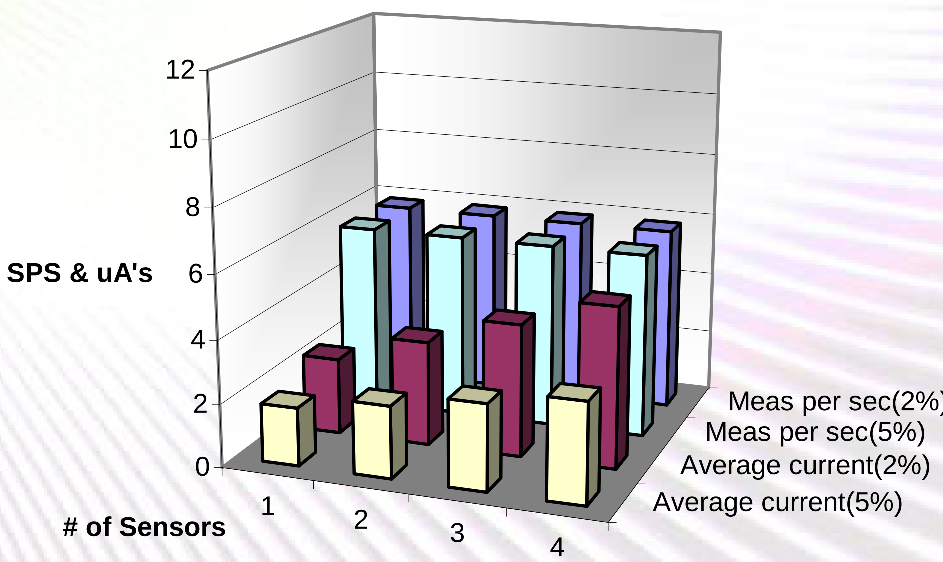
RC Method

- **Use lowest TACLK for needed counts**
  - ΔC 5% 8MHz TACLK
  - ΔC 2% 16MHz TACLK

Sensor Switch Application- RO
Current & SPS vs. Sensor Count (~20 counts)



Sensor Switch Application- RC
Current & SPS vs. Sensor Count (~20 counts)



Demo: ULP Key Detection

- **RC measurement flow**

```
// RC Method: Measurement Excerpt
...
P1OUT &=~(BIT0+BIT1+BIT2+BIT3); // everything is low
P1OUT |= active_key;    // Charge the sensor
_NOP();_NOP();_NOP();   // short time for hard pull high
P1IES |= active_key;    //-ve edge trigger
P1IE |= active_key;
P1DIR &= ~active_key;   // set the active key to input
timer_count = TAR;      // Take a snapshot of the timer
LPM0;
meas_cnt[i]= timer_count;
... // Now repeat with charging cycle and average results
```

```
// Port ISR
...
timer_count=TAR-timer_count; // Get charge/discharge time
...
```

Demo: ULP Key Detection

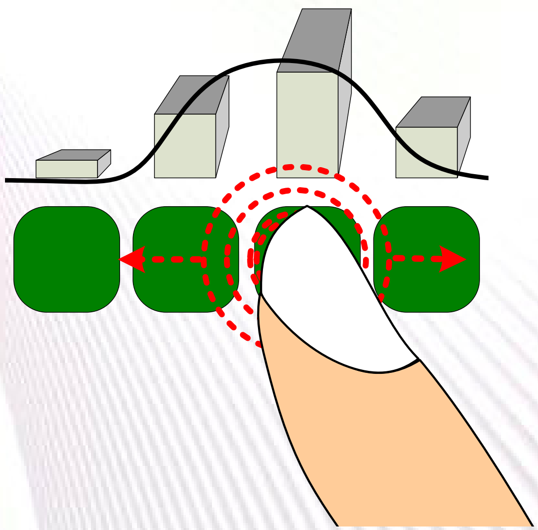
- RO measurement flow

```
// RO Method: Measurement Excerpt
TACTL = TASSEL_0+MC_2;                    // TACLK, cont mode
TACCTL1 = CM_3+CCIS_2+CAP;                // Pos&Neg Capture
CACTL1 |= CAON;                           // Turn on comparator
for (i = 0; i<Num_Sen; i++)
{switch (i)
  {case 0: // Sensor 1
           CAPD = CA_Ref+S_1; // Disable I/O:CA1 ref, 1st sensor
           CACTL2 = CA_1+CA_2;// CA1 ref, CAx sensor
           break;
   ...}}
WDTCTL = WDT_meas_setting; // Set duration of sensor measurement
TACTL |= TACLR;            // Clear Timer_A TAR
LPM0;                      // Wait for WDT interrupt
meas_cnt[i] = TACCR1;      // Save result
```

```
// WDT ISR
...
TACCTL1 ^= CCIS0;                    // Create SW capture of CCR1
...
```

Slider Scanning

- **Measurement Flow**
  - Step 1: Establish a base count measurement
  - Step 2: Set a key press count threshold
  - Step 3: Scan keys
  - Step 4: Calculate position based on counts for each key
- **Apply linear weighting algorithm**
- **Filter noise counts for jitter-free operation**



Position

- **Establish design to steps/sensor required**
  - Sensor size
  - Insulator thickness
- **Smoothly linearize steps across the slider**

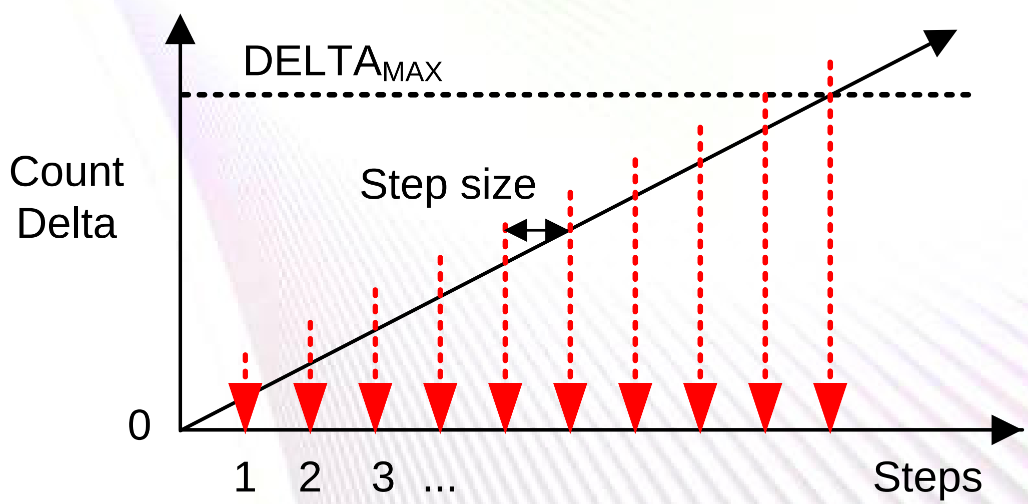
Set max delta expected
Set steps per key (steps$_{KEY}$)
Step size = max delta / steps$_{KEY}$
(slider steps = steps$_{KEY}$ * #keys)

↓

Get key delta & limit to max value
position$_{KEY}$ = delta / step size

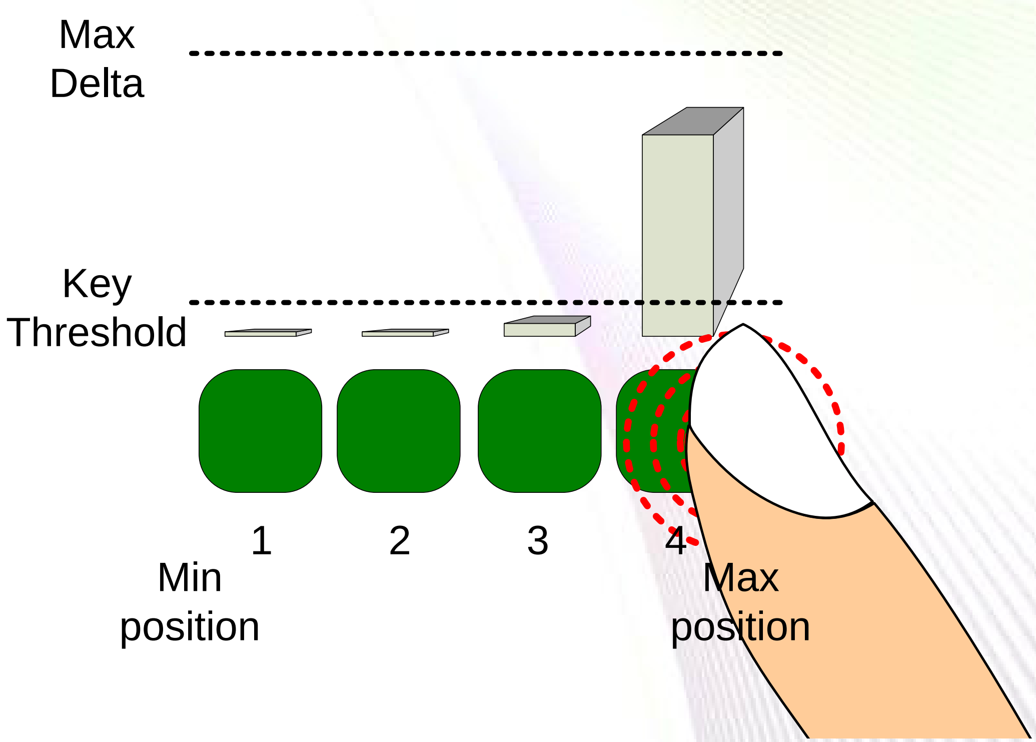
↓

If KEY pressed: Slider position = position$_{KEY}$ + steps*weight$_{KEY}$
(0, 1, 2, 3...)



Endpoint

- **Handle end-point touch**
  - Press beyond max
  - Movement beyond max
  - Movement from max



If end KEY pressed AND last position = max

↓ Y

If end-1 $position_{KEY}$ < = last end-1 $position_{KEY}$

↓ Y

position = max position
(finger moving beyond max)

If end KEY pressed AND last position = 0

↓ Y

If end-1 KEY not pressed

↓ Y

position = max position
(finger approaching from max)

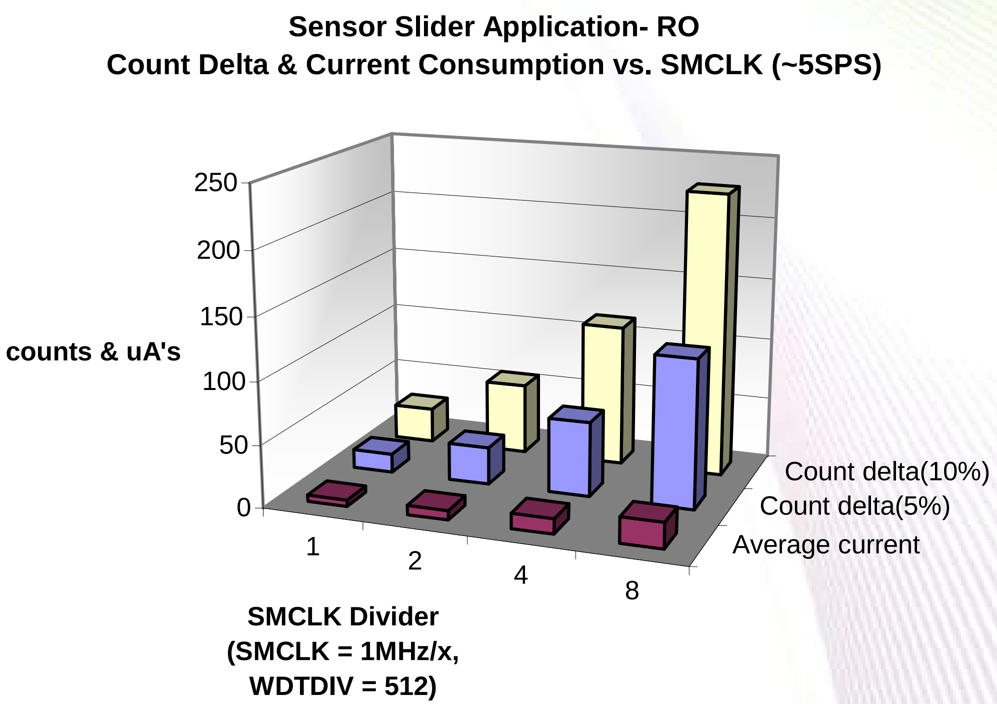
4-key Slider Current Consumption

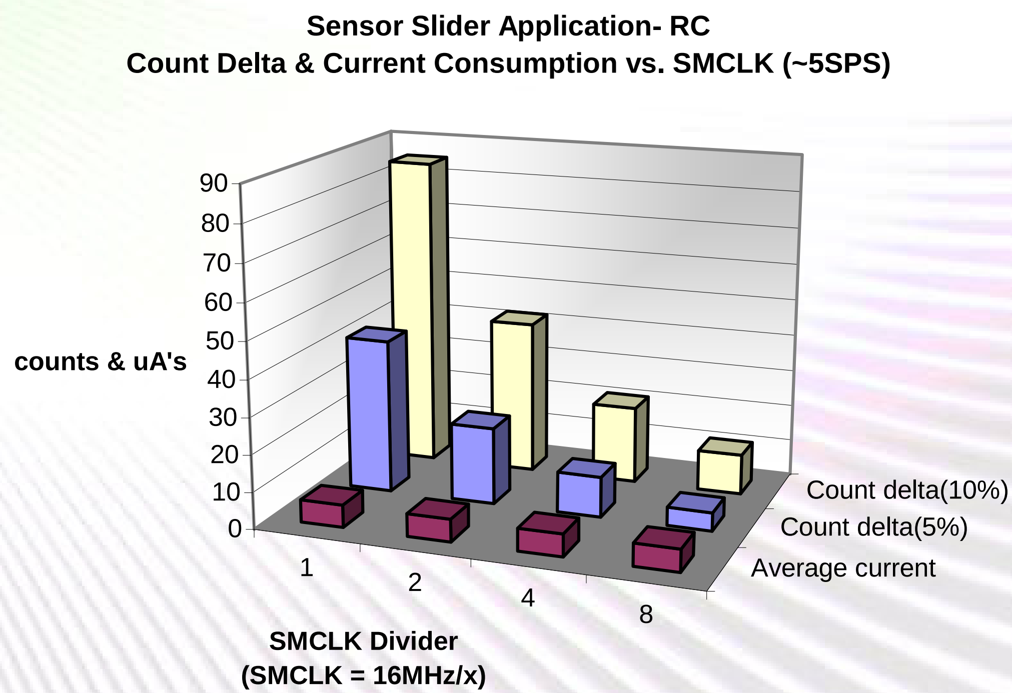
RO Method

- **t_meas user programmable**
  - Larger window = more counts
  - Define smallest window for needed counts, use lowest DCO for window

RC Method

- **t_meas is fixed by RC**
  - Faster TACLK = more counts
  - Don't divide TACLK, set = to fastest DCO required for needed counts





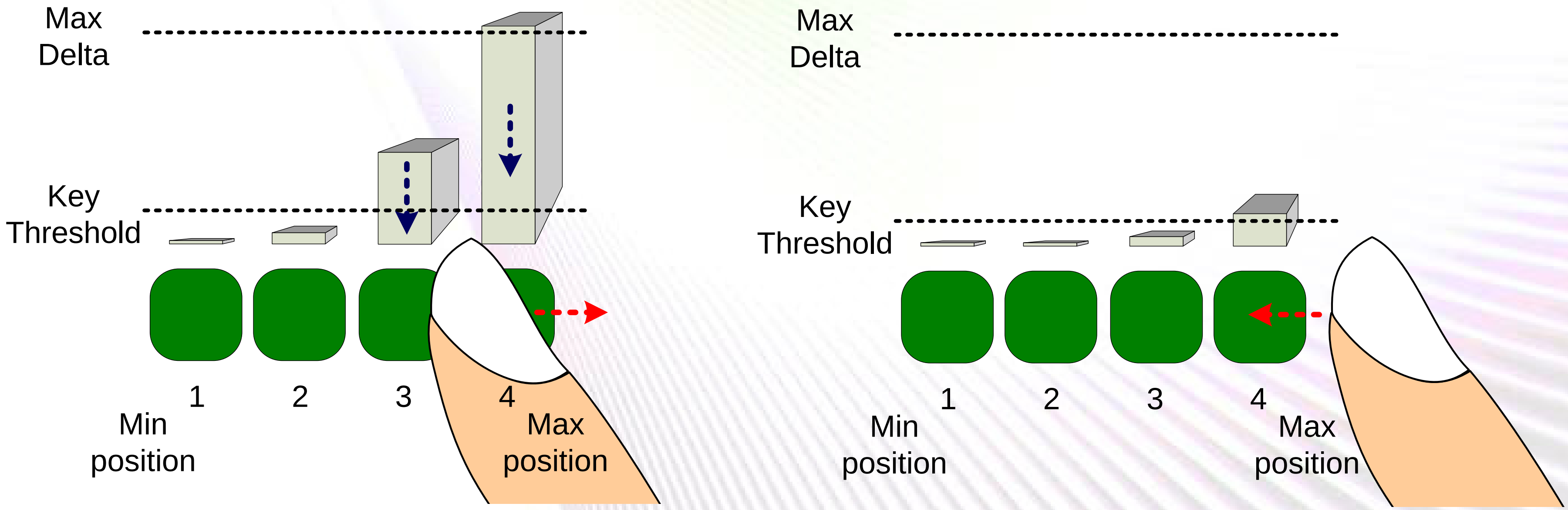
Demo: ULP Slider Detection

```
// Sensor slider definitions
#define Num_Sen     4    // # of sensors
#define KEY_lvl     5    // min count for a "key press"
                          // Must be less than step_size

#define max_cnt     100  // Set below actual max delta expected
#define num_steps   16   // How many steps per key?
#define step_size   (max_cnt/num_steps) // Step size for position
...
if (delta_cnt[i] > max_cnt)  // count exceeds preset upper delta
 delta_cnt[i] = max_cnt;     // limit to set point

key_pos[i] = delta_cnt[i]/step_size;  // individual "position"

if (key_pos[i] > 0)           // If the key is "pressed",
 position = key_pos[i] + num_steps*(i); // Pos=0-16, key weight
```

- Determine legitimate number of steps for a given application
- Linearize across all sensors for entire slider span

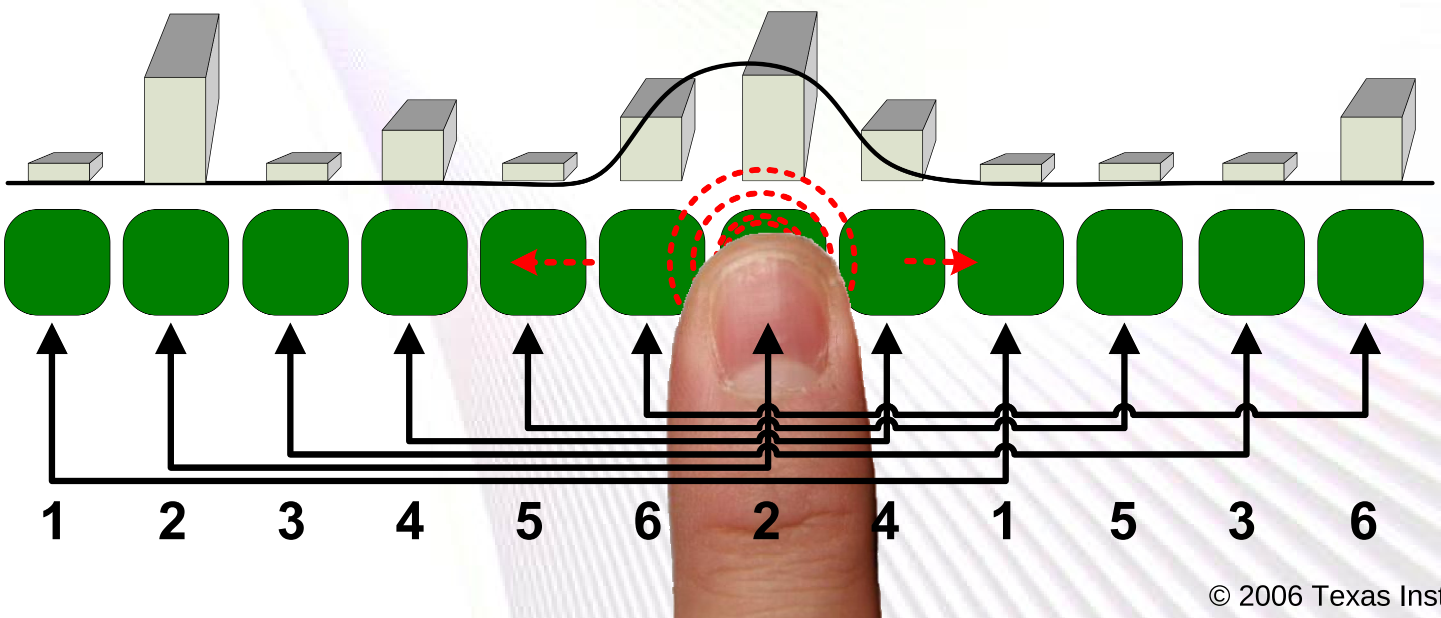
Demo: ULP Slider Endpoint

```
// Handle max end of slider
if (key_press[3] && position_old == Num_Sen*num_steps)
{if (key_pos[2]<key_pos_old[2] || key_pos[2]==key_pos_old[2])
  position = Num_Sen*num_steps; // moving beyond the max
}
else if (key_press[3] && position_old == 0 && !key_press[2])
 position = Num_Sen*num_steps; // approaching from max
```



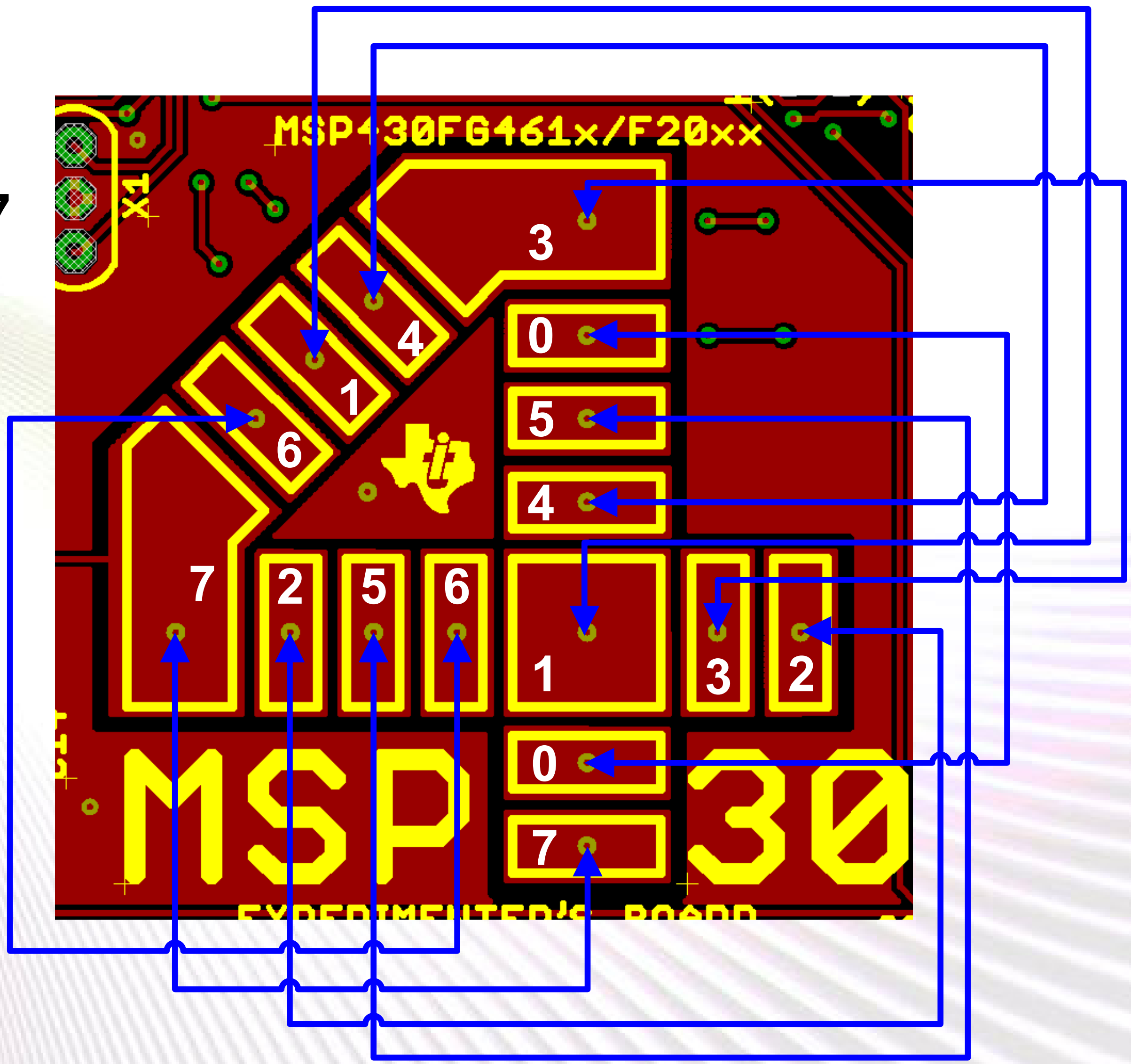
Multiplexed Sliders

- **Multiplex sensors for better pin:sensor ratio**
  - Increases base capacitance
  - Measured delta C will be lower
- **Mux for unique pattern for each position**
- **Multiple sensors should be excited for proper location & direction detection**



ATC2006 Touchpad Interface

- 8 port pins used
- 2x8 = 16 sensors
- 0-7: P1.0-P1.5,P2.6,P2.7



Agenda

- Overview of Single Touch Applications
- System-Level Careabouts
- MSP430 Implementations
- Keys, Sliders & Demos
- Summary

Summary

- **Capacitive touch sensing can be an attractive option**
  - …for existing switch replacement
  - .. and more: potentiometer replacement, multi-position switches
- **MSP430 RO Method**
  - Works in Comp_A+ devices
  - Number of independent sensors limited by CA+ mux
  - Needs 1 external R per sensor + reference ladder
  - Sensitivity limited by current consumption, flexible measurement time
- **MSP430 RC Method**
  - Can be implemented on any MSP430
  - Up to 16 independent sensors (16 interruptible GPIOs)
  - Single external R per two sensors
  - Sensitivity limited by on-chip max clock frequency, fixed measurement time
  - Lowest power implementation

IMPORTANT NOTICE