Microchip MASTERs 中国技术精英年会 2018

C19H05 TNG23

挑战触控设计解决方案，快速完成触控项目

MICROCHIP

触摸和手势解决方案
MASTERs 2018

C19L07 TNG14 使用Microchip 1D/2D/3D触摸技术设计独一无二的用户界面！

正在上的课程

C19H05 TNG23 挑战触控设计解决方案，快速完成触控项目

MICROCHIP MASTERs 2018

主讲人及助教

- 资深嵌入式解决方案工程师**Jim Li**
- 资深嵌入式解决方案工程师**Vincent Liu**
- 主任嵌入式解决方案工程师**Alan Yao**
- 主任嵌入式解决方案工程师**Jerry Wang**

课程目标

完成本课程之后，您将能够...

- 了解如何克服常见的设计挑战，以实现稳健的低功耗触摸设计
- 熟悉评估EMC的要求，以实现辐射量降低且能够在电噪声下工作的触摸设计
- 创建可在有水/潮湿环境下工作的1D（按钮和/或滑动条和滚轮）电容式触摸设计

课程目标

完成本课程之后，您将能够...

- 使用**mTouch®**和**QTouch®**库中的触摸**API**和相关参数在基于**MCU**的设计中实现稳健的触摸功能。
- 使用**MCC**和**START**生成项目，以使用**mTouch®**和**QTouch®**模块化库实现触摸按钮。
- 使用**Data Visualizer**工具为**mTouch®**和**QTouch®**设计调试和优化触摸参数。
- 学习用于实现防水和防噪声的稳健触摸设计的高级调节设置。

MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

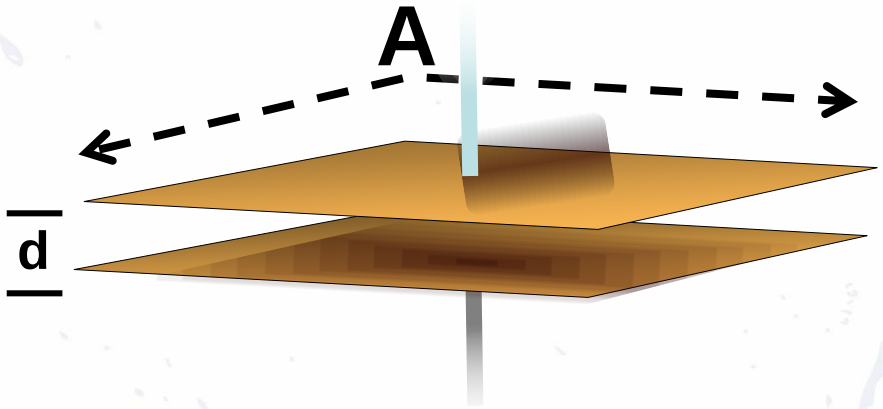
课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

什么是电容式传感

什么是电容式传感？

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



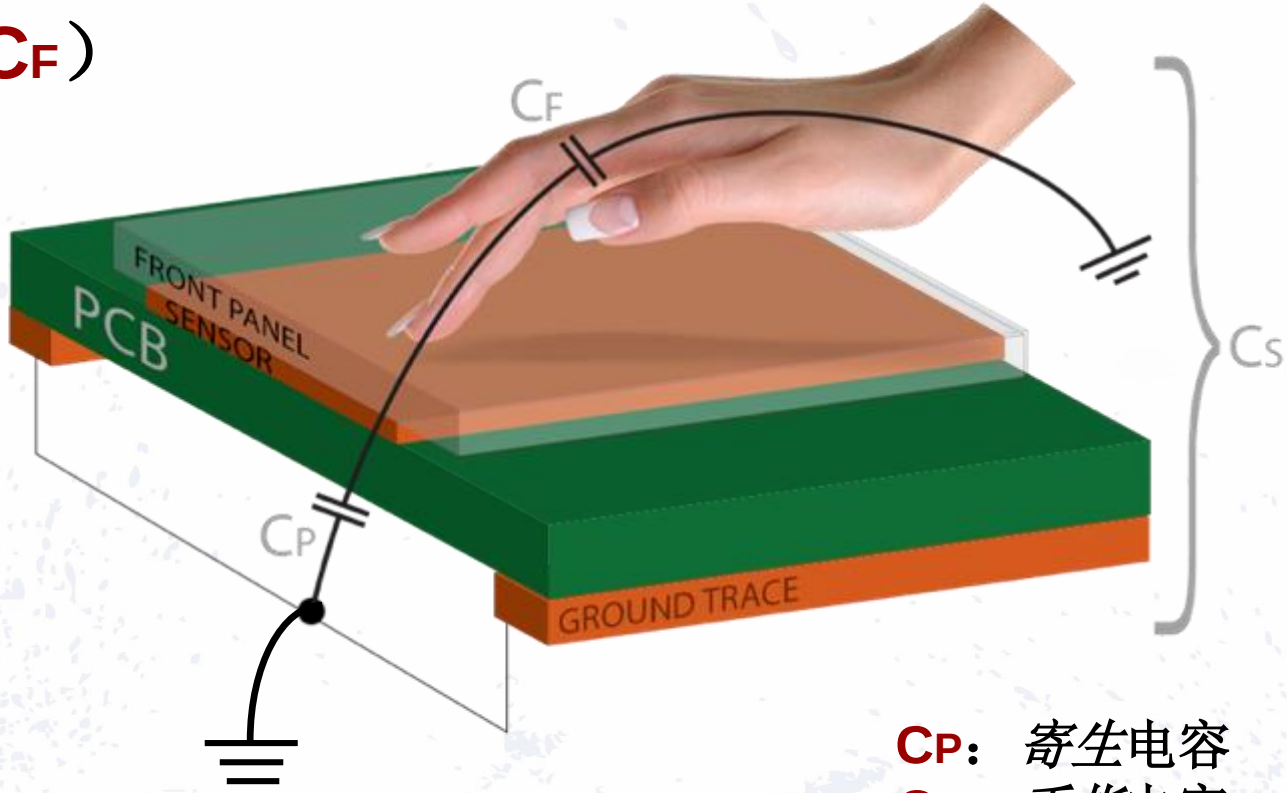
C	电容（F）
ε_0	自由空间的介电常数（8.854 pF/m）
ε_r	相对介电常数（无单位）
A	板面积（m^2）
d	板间距离（m）

材料	ε_r
空气	1
聚乙烯	2.25
聚苯乙烯	2.4 – 2.7
玻璃	4 – 10
FR-4	4.8
水	80

电容式触摸传感器

引入手指会产生并联电容

（C_F）



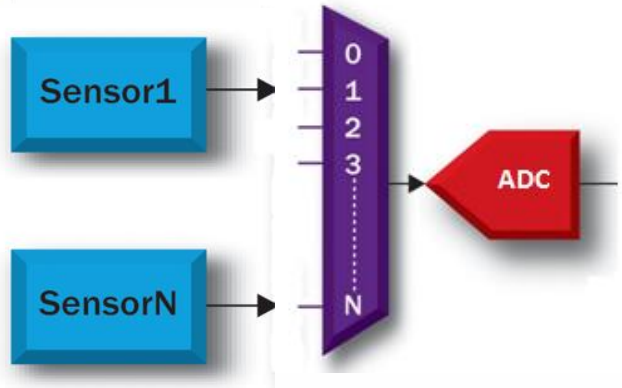
C_P：*寄生*电容

C_F：*手指*电容

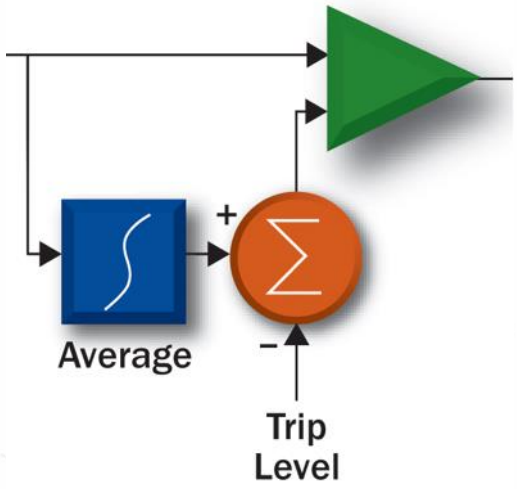
C_S：总*传感器*电容

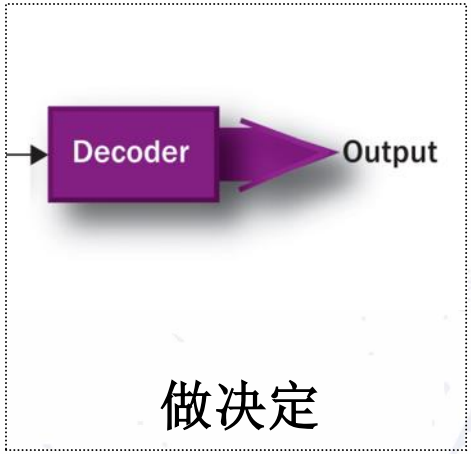
传感器电容（C_S）= C_P + C_F

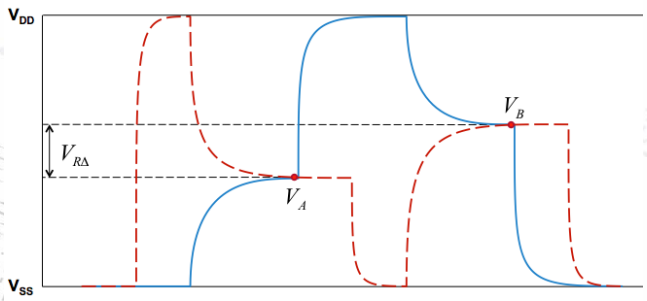
电容处理

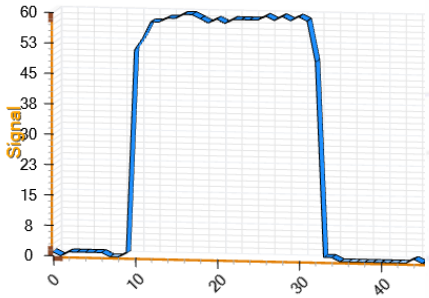


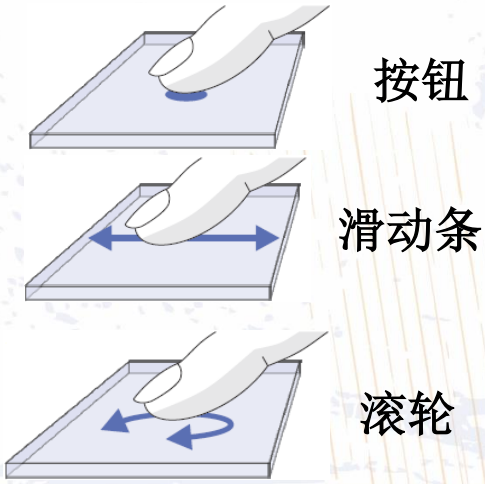
信号采集









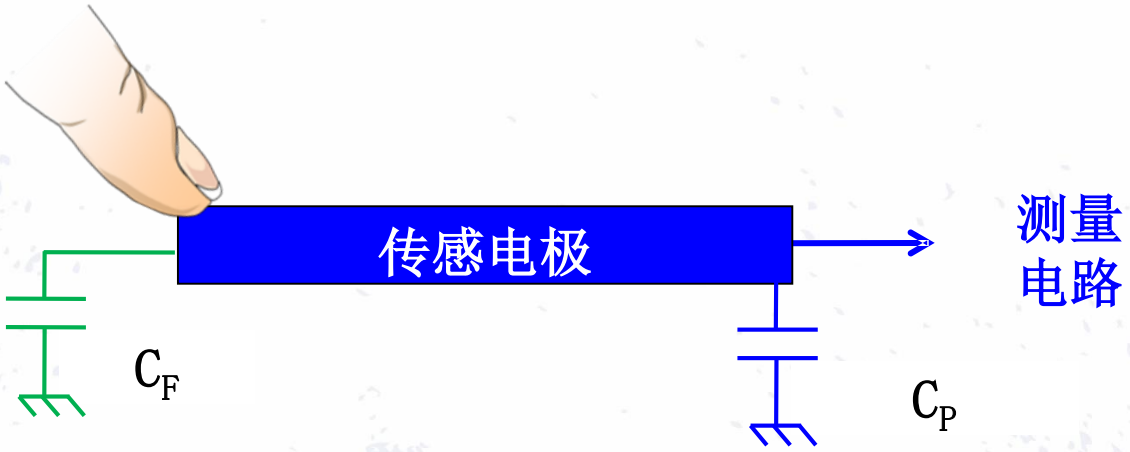


课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

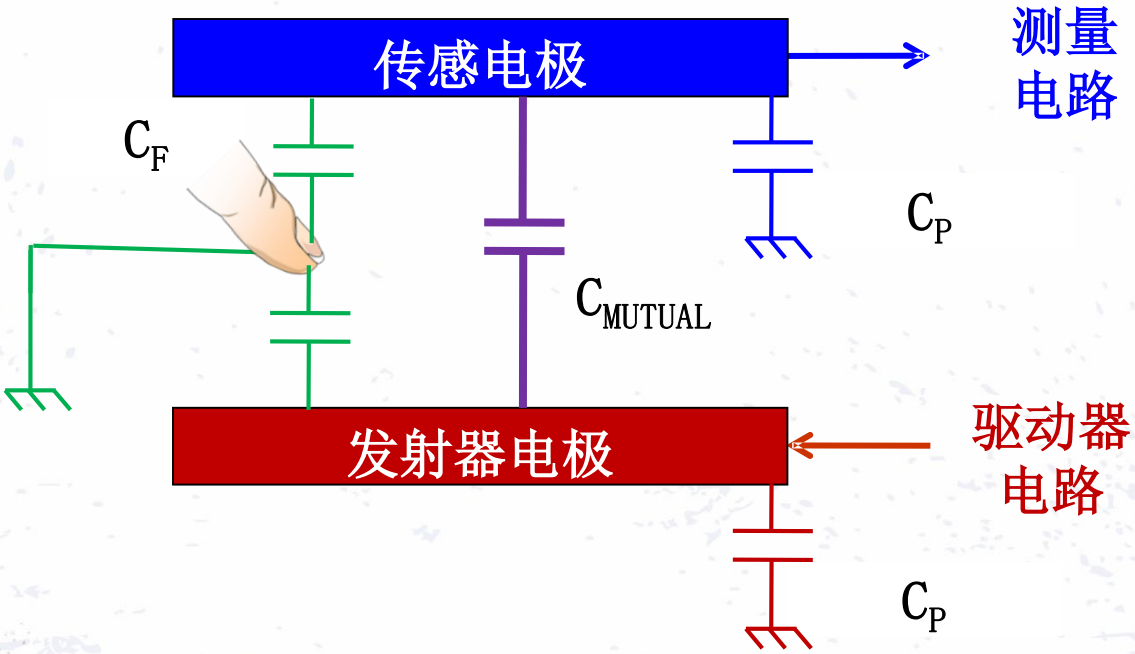
自电容

- 电极的自电容是其施加到测量电路上的相对于电路地的电容负载。



互电容

- 互电容是接收器和发射器电极之间的电容耦合。

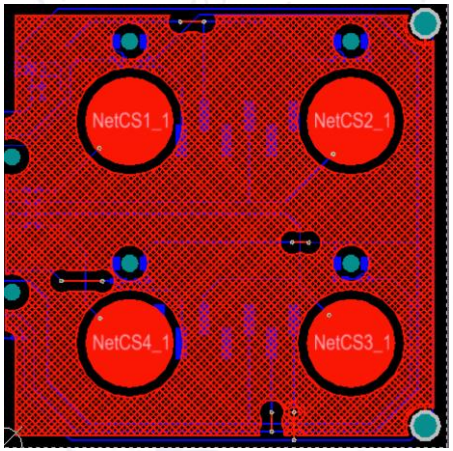


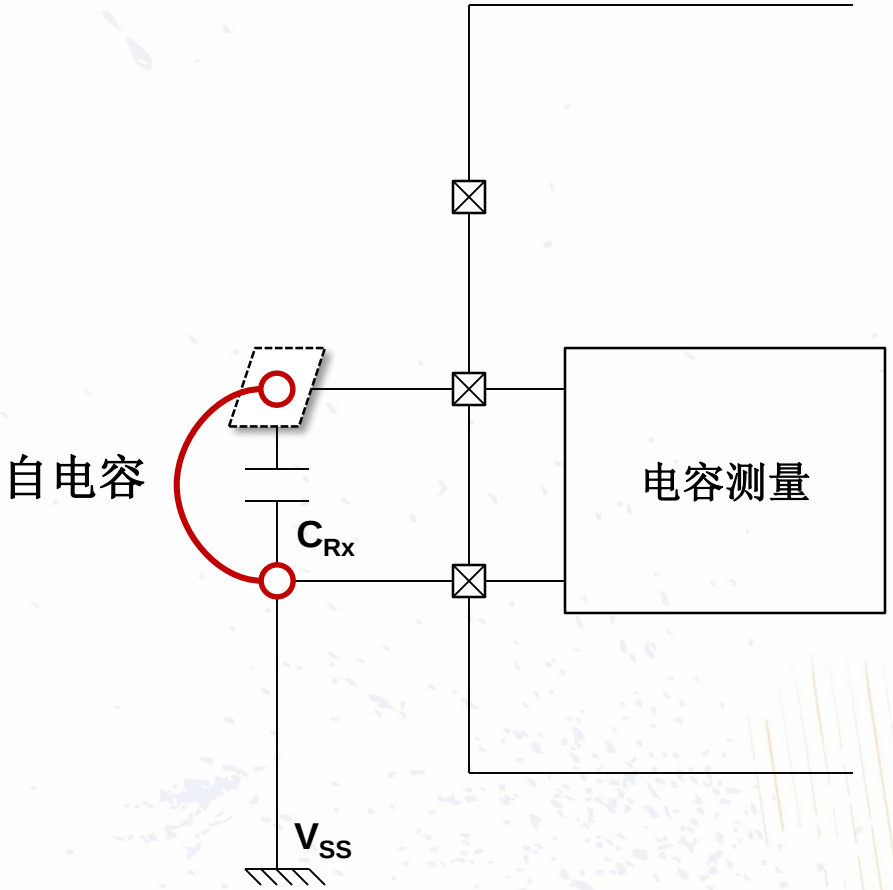
课程安排

- 电容式传感电路模型基础
  - 自电容
  - 互电容

自电容

PCB示例

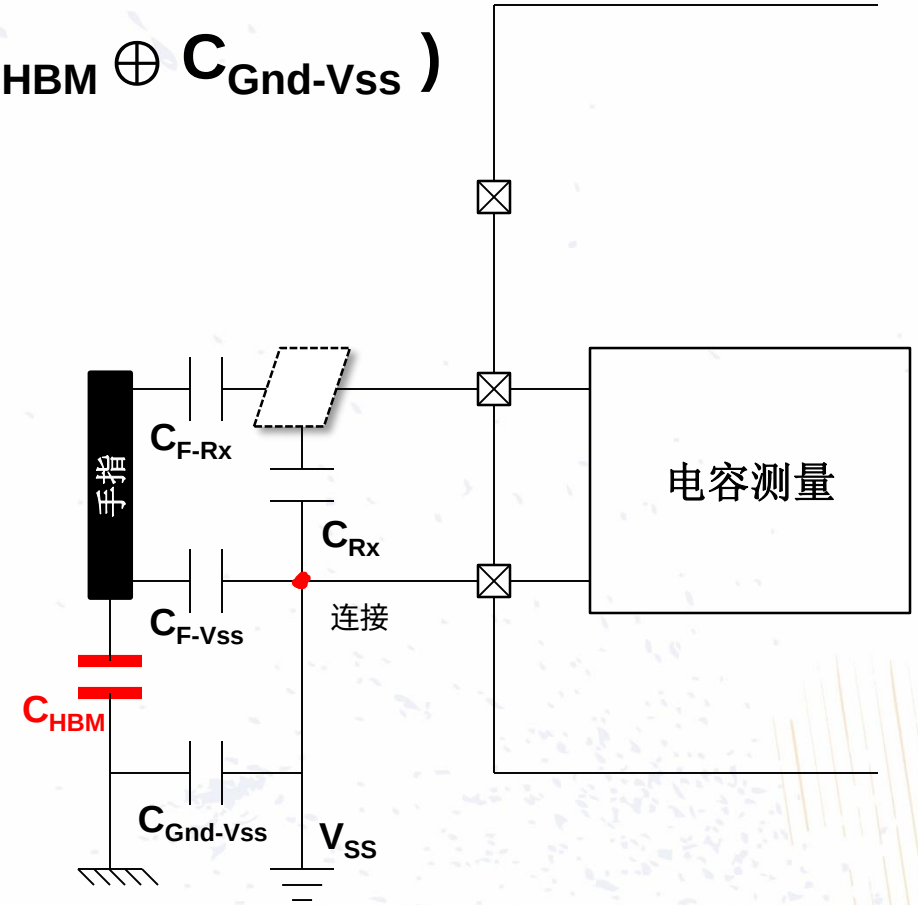




自电容

$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM} \oplus C_{Gnd\text{-}Vss})$$

- C_{HBM}：100-200 pF
- $C_{F\text{-}Rx}$：传感器设计
  - 传感器尺寸
  - 覆盖材料和厚度
  - 典型值0.25 pF-0.5 pF
- $C_{F\text{-}Vss}$：PCB上的接地区域
  - 0.5 pF-100 pF，基于用户采用何种方式耦合到器件。
- $C_{Gnd\text{-}Vss}$：器件接地方案
  - 电池供电
  - 干线电源供电



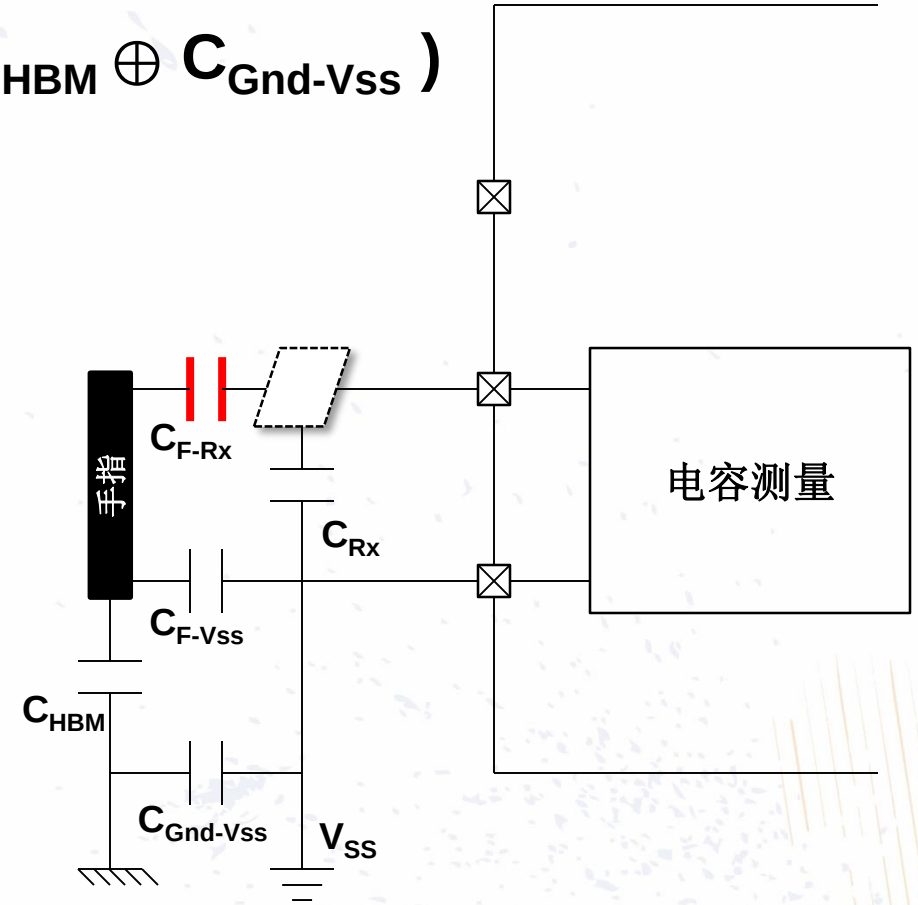
⊕表示串联电容

C1 ⊕ C2 = C1*C2/(C1+C2)

自电容

$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM} \oplus C_{Gnd\text{-}Vss})$$

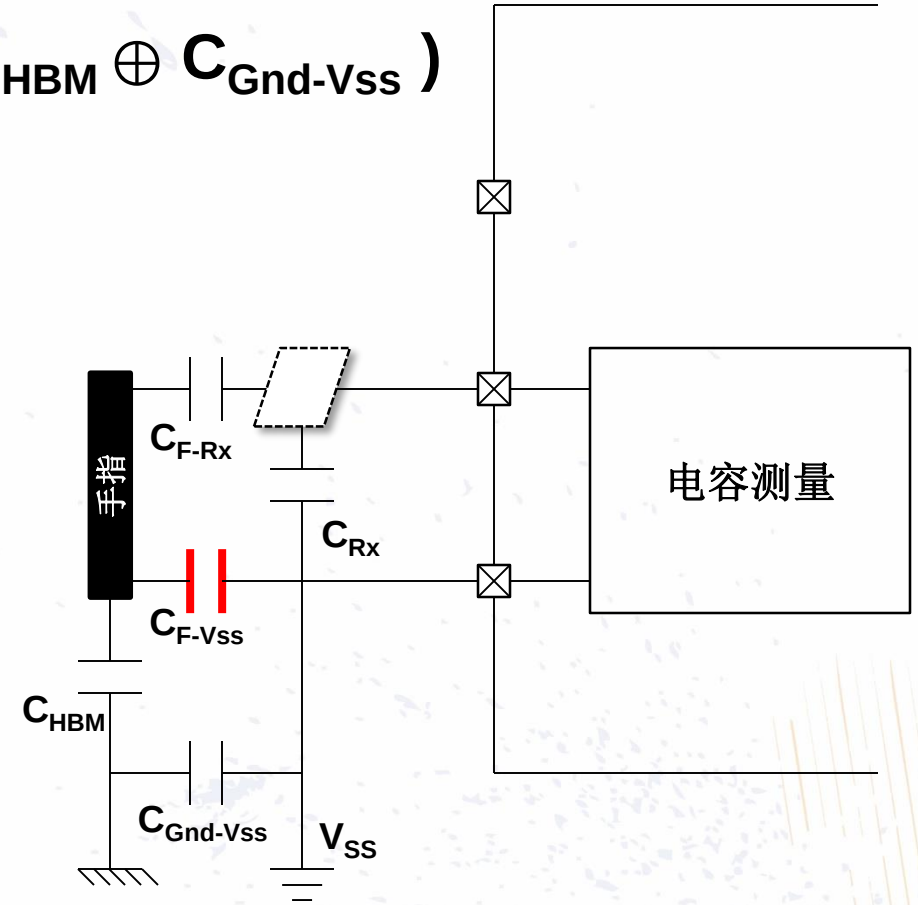
- C_{HBM}：100-200 pF
- $C_{F\text{-}Rx}$：传感器设计
  - 传感器尺寸
  - 覆盖材料和厚度
  - 典型值0.25 pF-0.5 pF
- $C_{F\text{-}Vss}$：PCB上的接地区域
  - 0.5 pF-100 pF，基于用户采用何种方式耦合到器件。
- $C_{Gnd\text{-}Vss}$：器件接地方案
  - 电池供电
  - 干线电源供电



自电容

$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM} \oplus C_{Gnd\text{-}Vss})$$

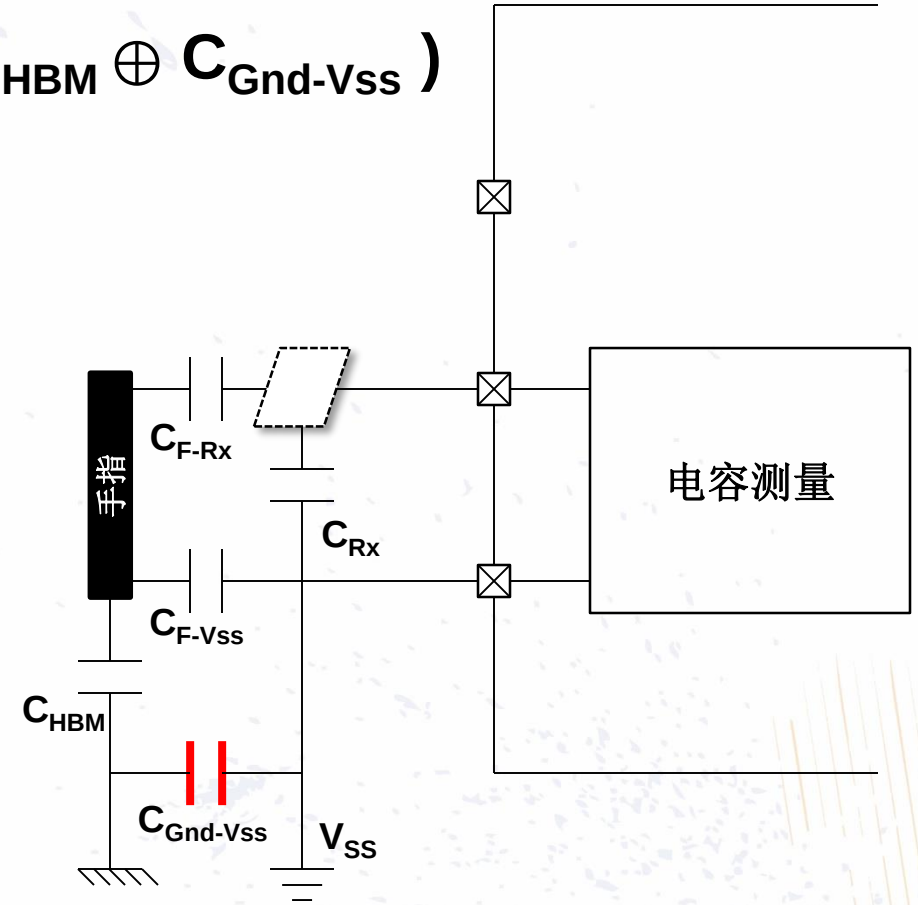
- C_{HBM}：100-200 pF
- $C_{F\text{-}Rx}$：传感器设计
  - 传感器尺寸
  - 覆盖材料和厚度
  - 典型值0.25 pF-0.5 pF
- $C_{F\text{-}Vss}$：PCB上的接地区域
  - 0.5 pF-100 pF，基于用户采用何种方式耦合到器件。
- $C_{Gnd\text{-}Vss}$：器件接地方案
  - 电池供电
  - 干线电源供电



自电容

$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM} \oplus C_{Gnd\text{-}Vss})$$

- C_{HBM}：100-200 pF
- $C_{F\text{-}Rx}$：传感器设计
  - 传感器尺寸
  - 覆盖材料和厚度
  - 典型值0.25 pF-0.5 pF
- $C_{F\text{-}Vss}$：PCB上的接地区域
  - 0.5 pF-100 pF，基于用户采用何种方式耦合到器件。
- $C_{Gnd\text{-}Vss}$：器件接地方案
  - 电池供电
  - 干线电源供电



自电容

共用接地系统

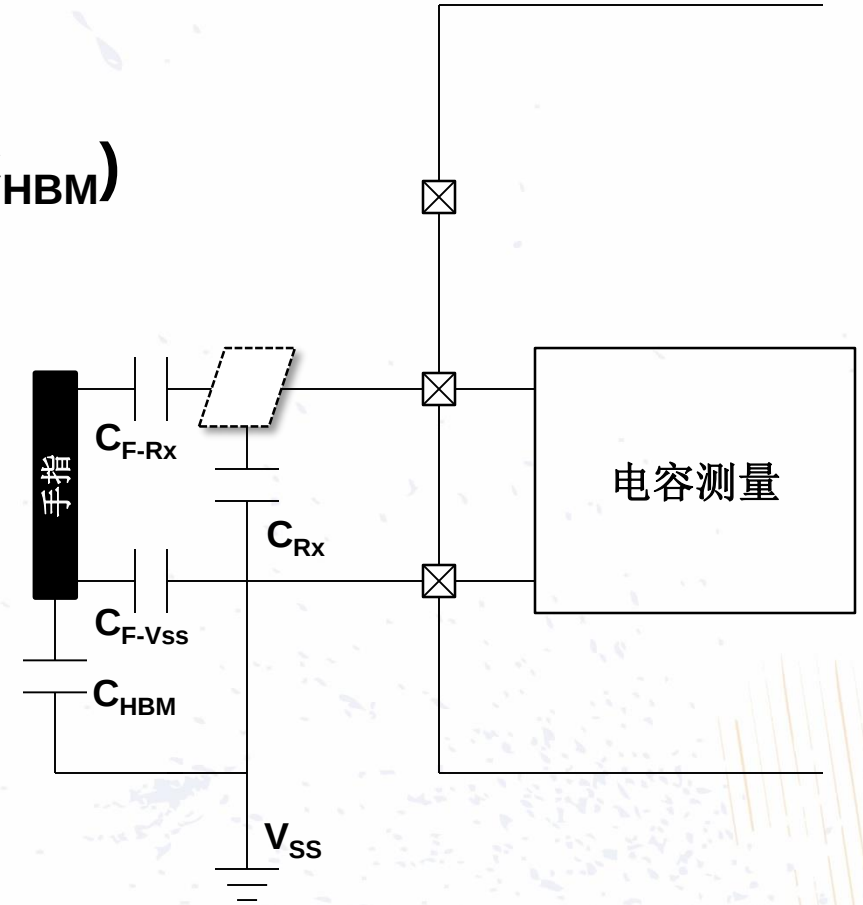
$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM})$$

$C_{F\text{-}Vss}$通常比C_{HBM}小很多。

$$\Delta C_{total} \approx C_{F\text{-}Rx} \oplus C_{HBM}$$

$C_{F\text{-}Rx}$通常比C_{HBM}小很多。

$$\Delta C_{total} \approx C_{F\text{-}Rx}$$



自电容

共用接地系统

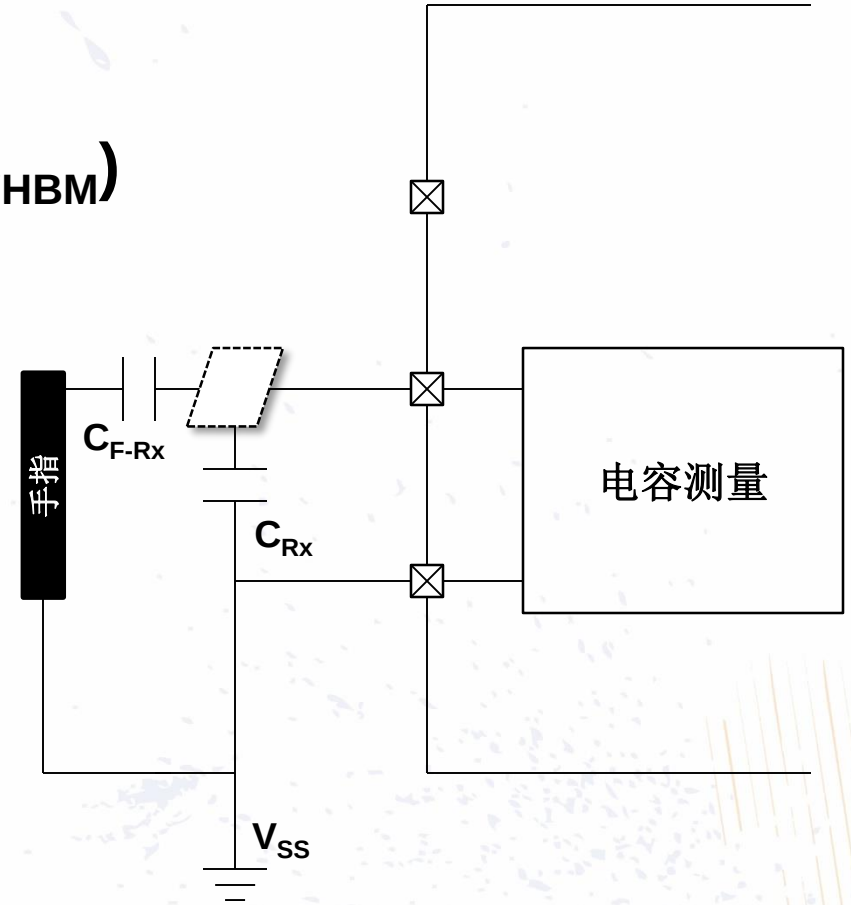
$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM})$$

$C_{F\text{-}Vss}$通常比C_{HBM}小很多。

$$\Delta C_{total} \approx C_{F\text{-}Rx} \oplus C_{HBM}$$

$C_{F\text{-}Rx}$通常比C_{HBM}小很多。

$$\Delta C_{total} \approx C_{F\text{-}Rx}$$



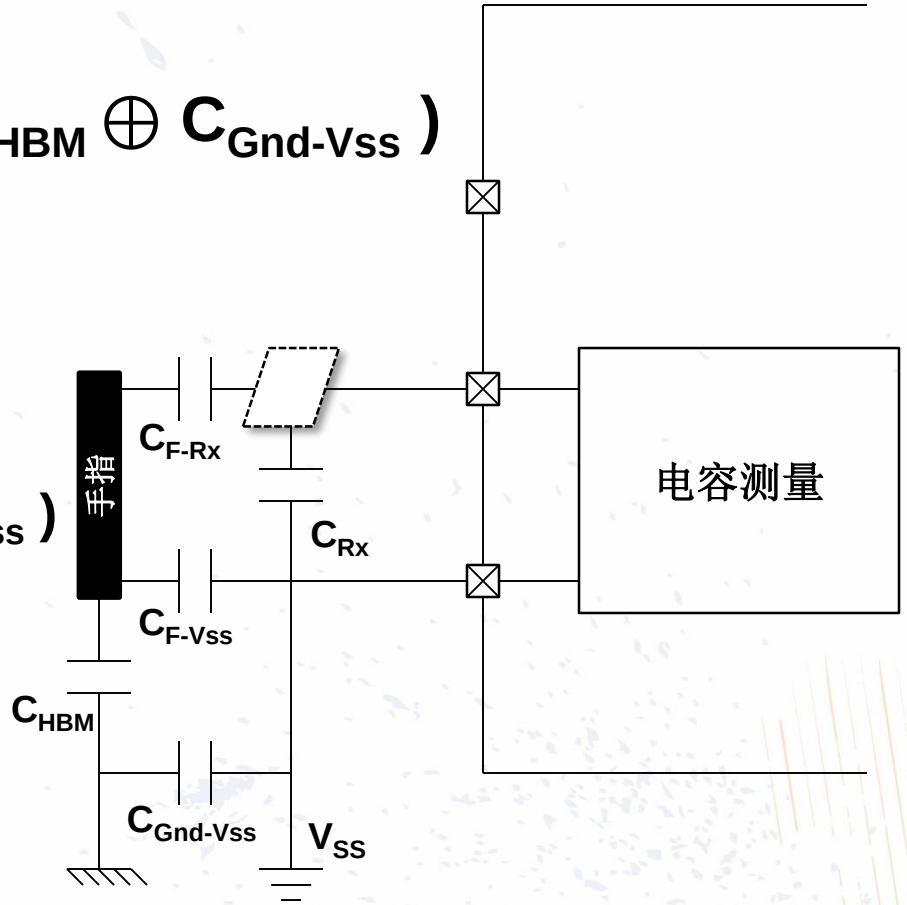
自电容

隔离接地系统

$$\Delta C_{total} = C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{HBM} \oplus C_{Gnd\text{-}Vss})$$

$C_{GND\text{-}Vss}$通常比C_{HBM}小很多。

$$\Delta C_{total} \approx C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{Gnd\text{-}Vss})$$



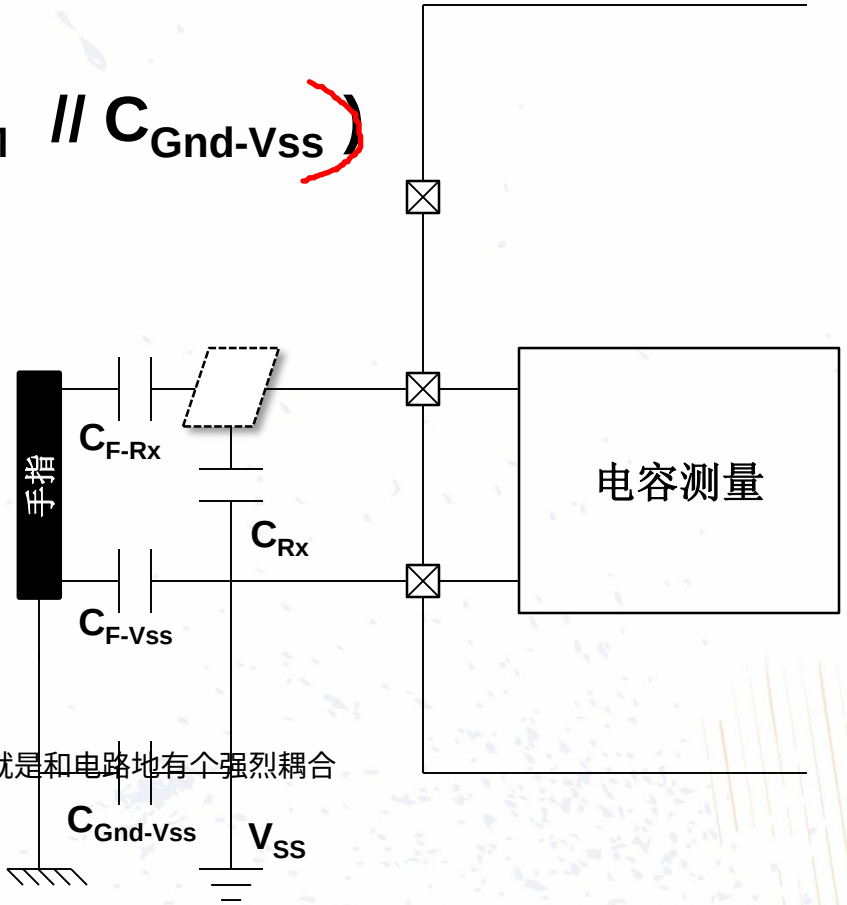
自电容

隔离接地系统

$$\Delta C_{total} = C_{F\text{-}Rx} \,//\, (C_{F\text{-}Vss} + C_{HBM} \,//\, C_{Gnd\text{-}Vss})$$

$C_{GND\text{-}Vss}$通常比C_{HBM}小很多。

$$\Delta C_{total} \approx C_{F\text{-}Rx} \oplus (C_{F\text{-}Vss} + C_{Gnd\text{-}Vss})$$

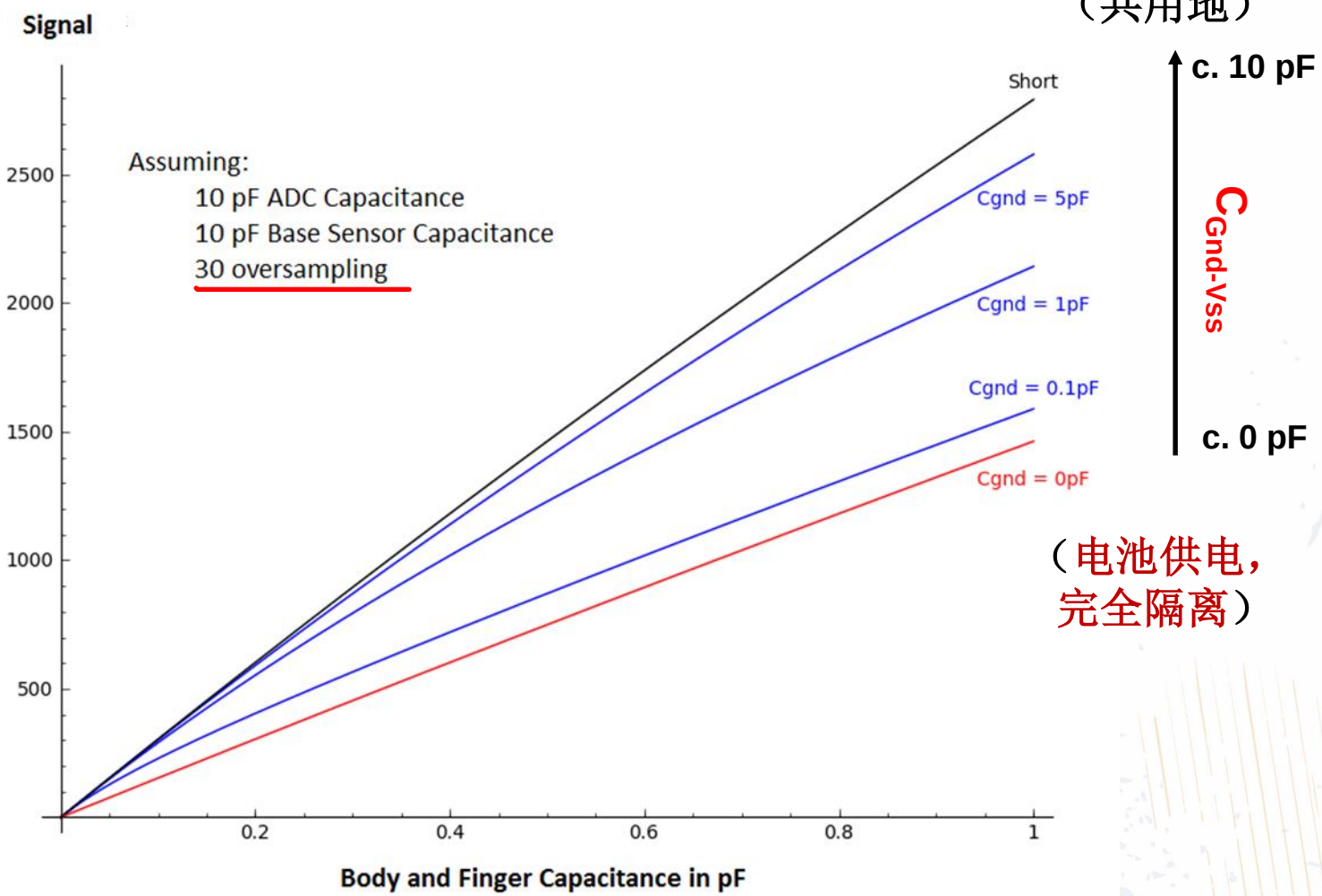


总结：
除了良好的传感器设计，还可最大限度地增大PCB上的地平面，但远离传感器。

增大面积，就是和电路地有个强烈耦合

自电容
$C_{Gnd-Vss}$的影响

（共用地）



（灵敏度）

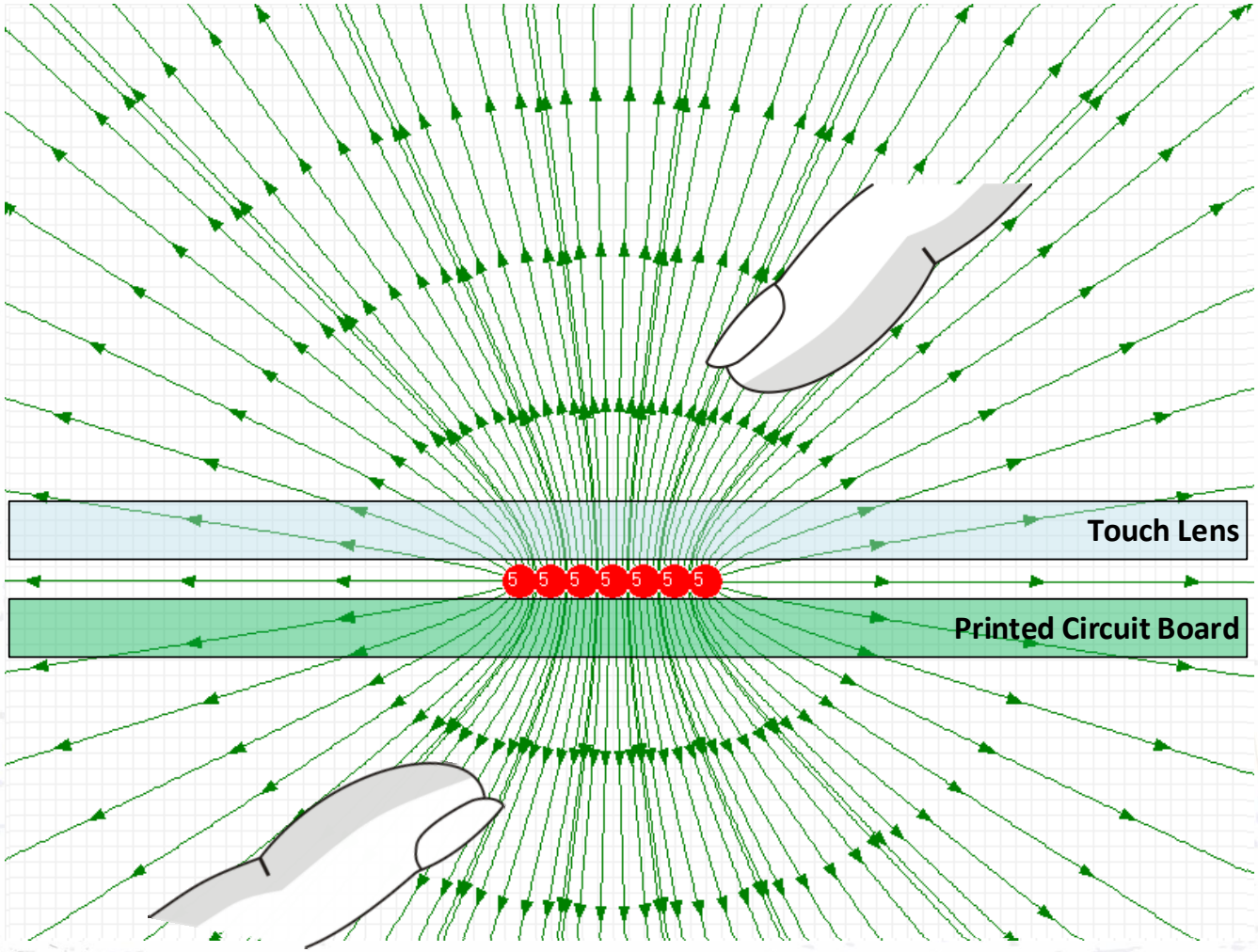
这个灵敏度怎么来的

（电池供电，完全隔离）

（用户与传感器的耦合）

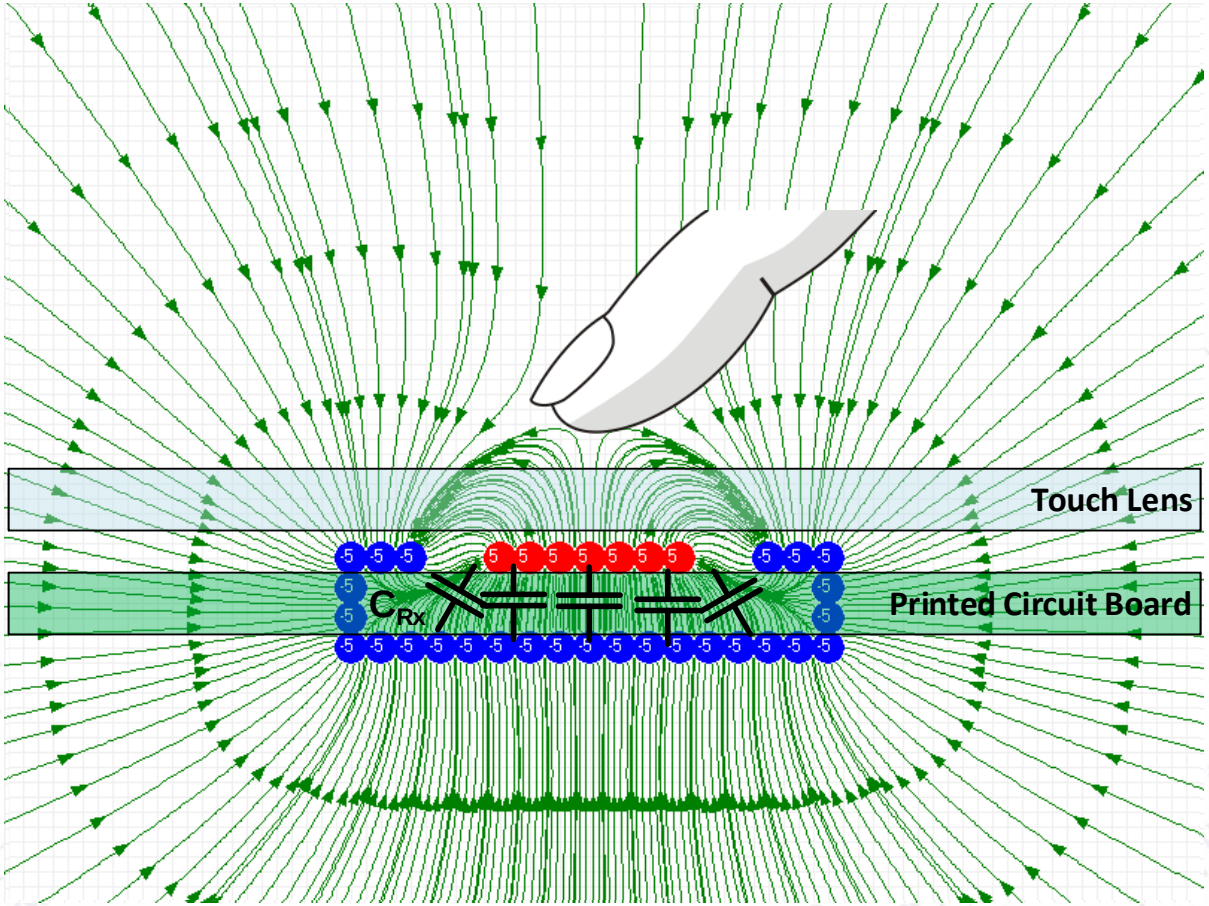
假定：身体耦合 = 手指耦合

自电容
传感器未屏蔽



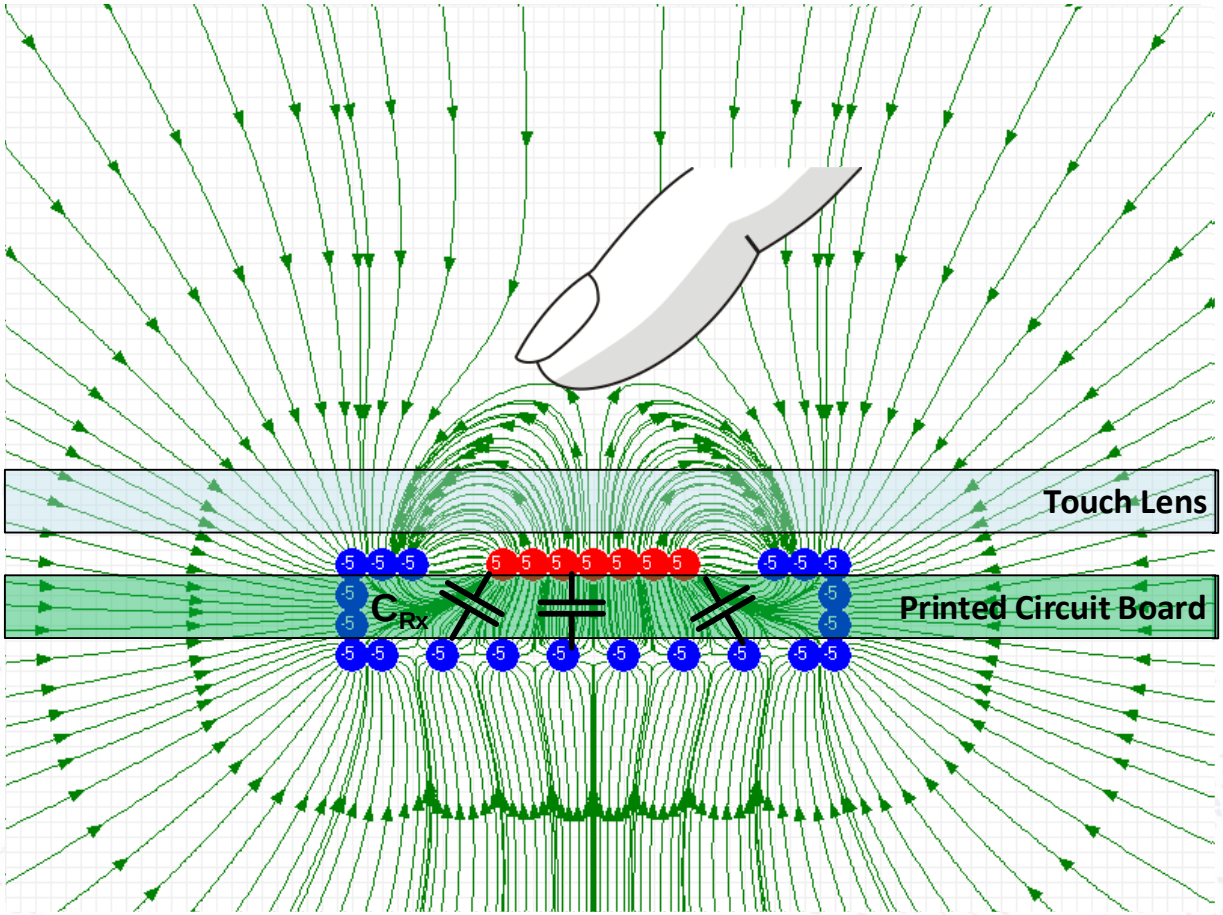
自电容

传感器 + 地



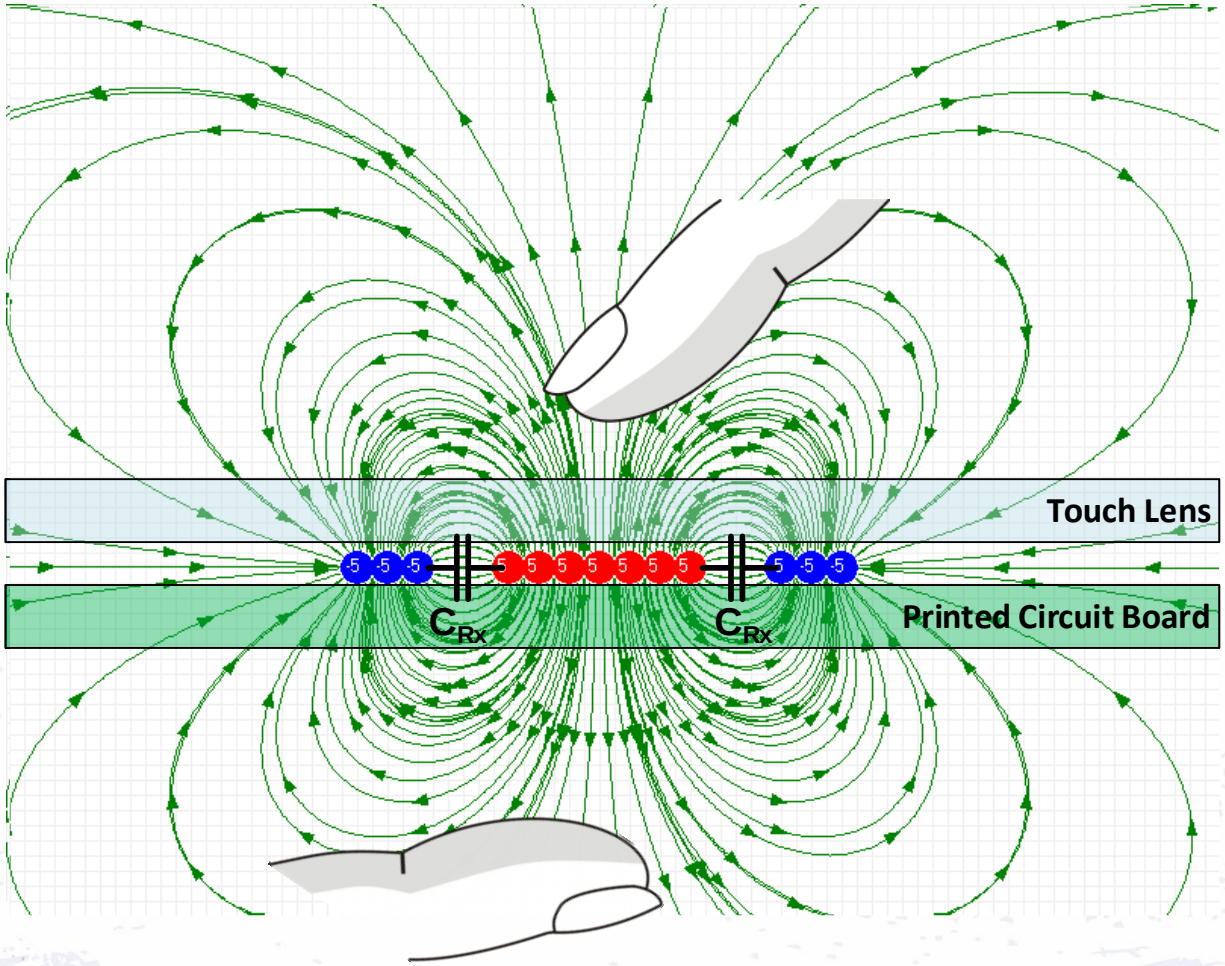
自电容

传感器 + 地



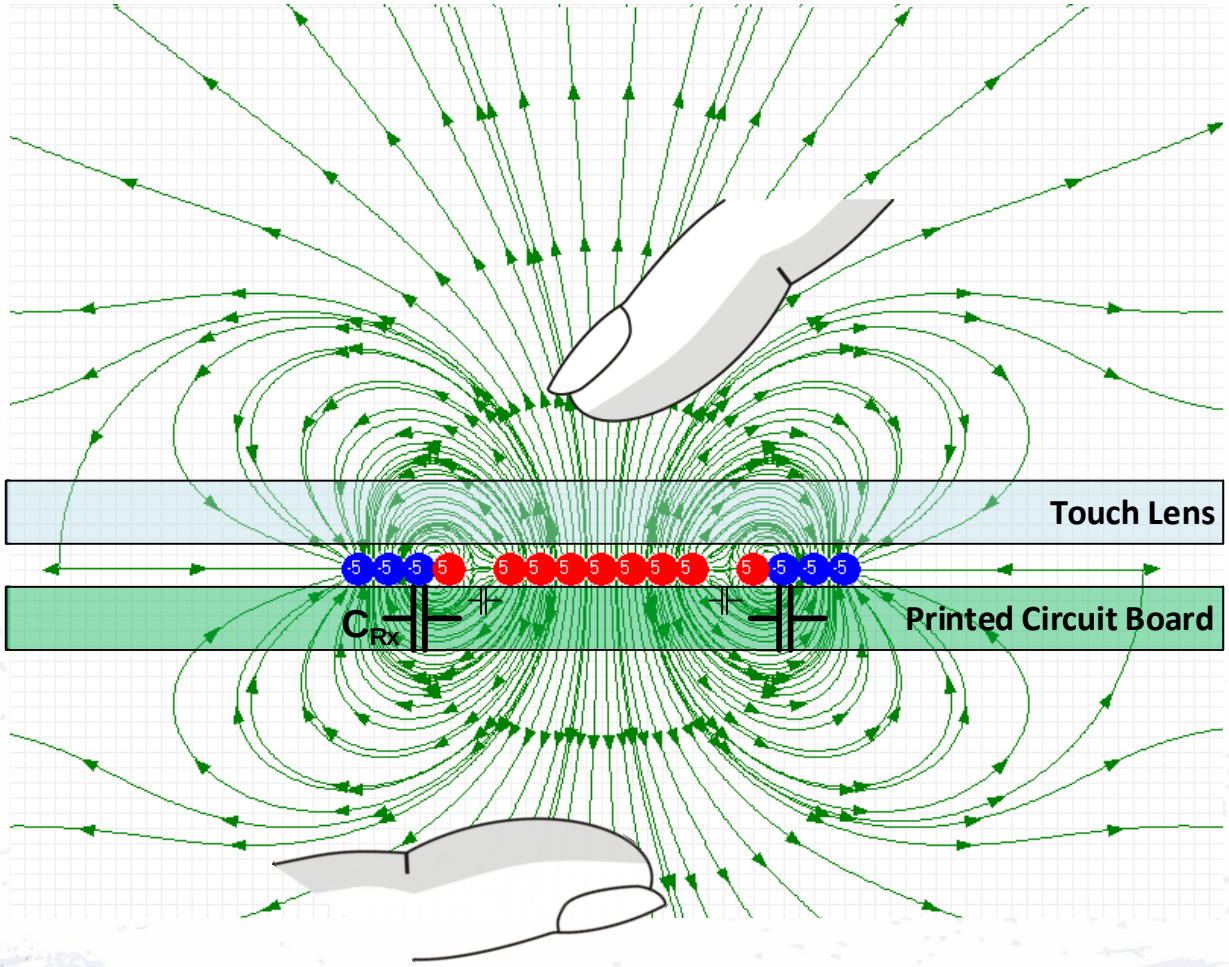
自电容

传感器 + 地



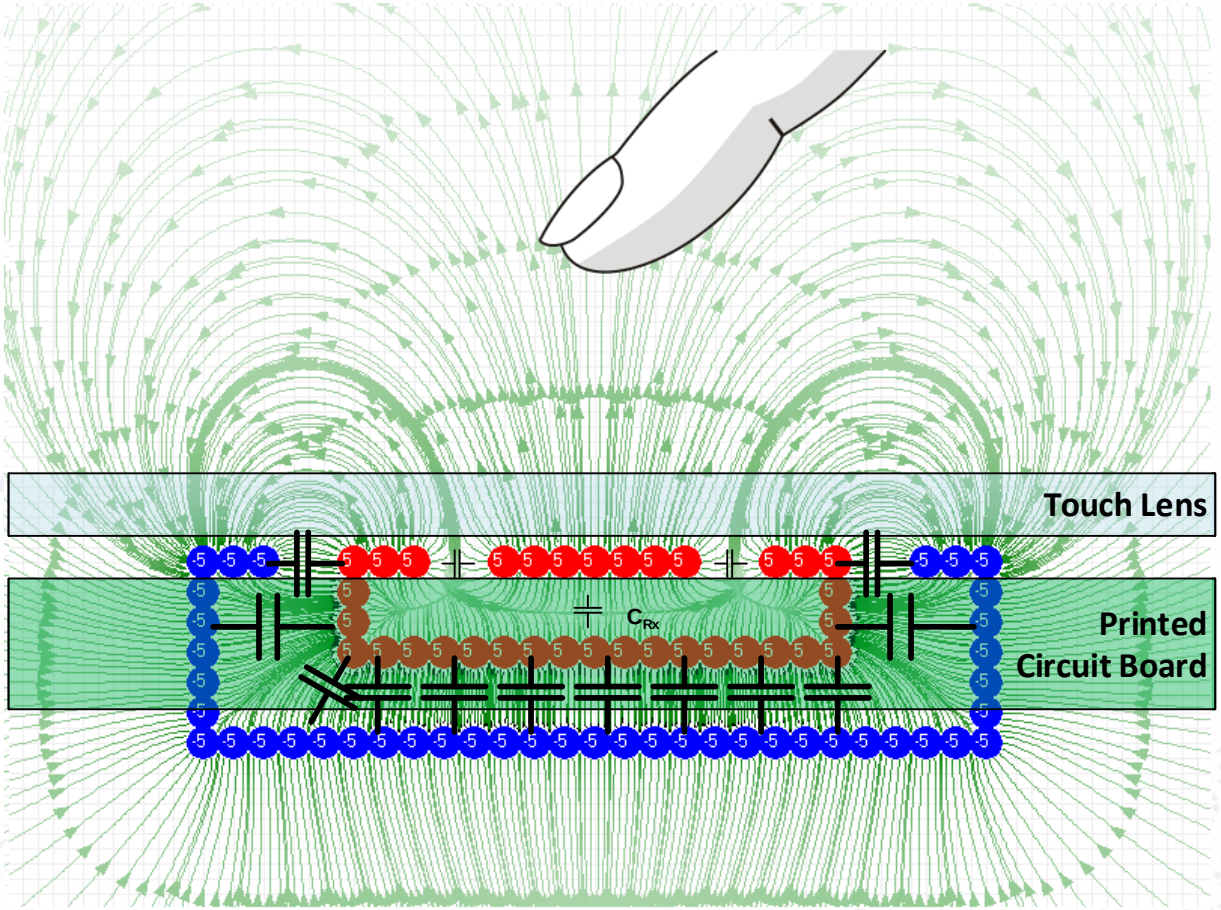
自电容

传感器 + 驱动屏蔽 + 地



自电容

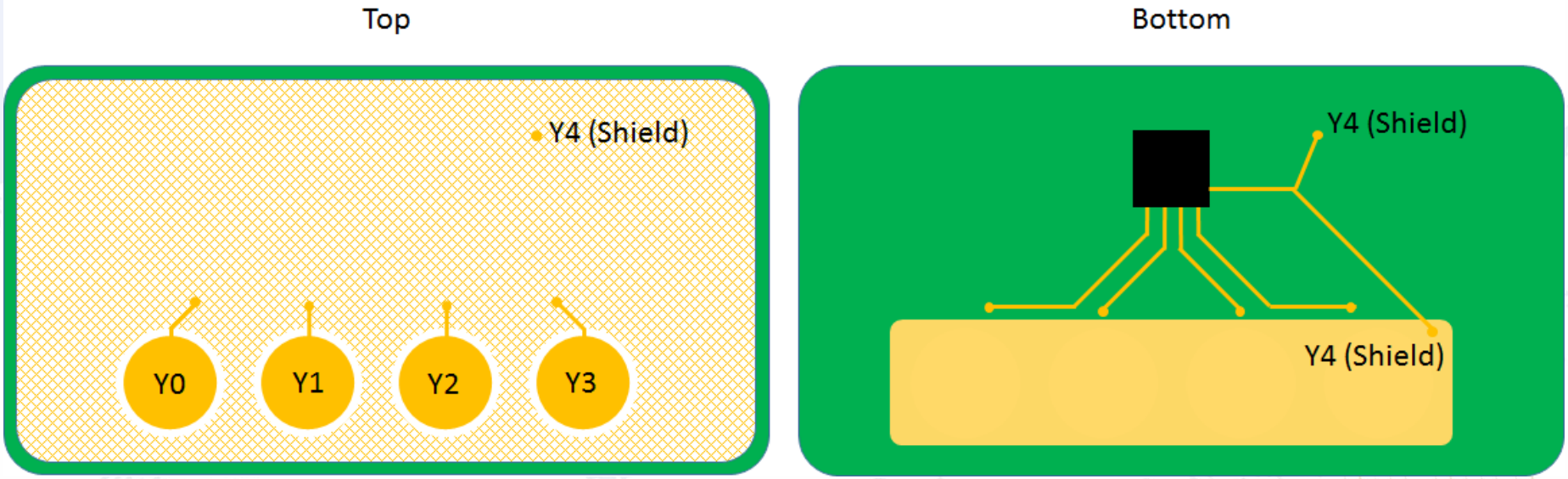
传感器 + 驱动屏蔽 + 地



四层板

驱动屏蔽示例

PCB布线

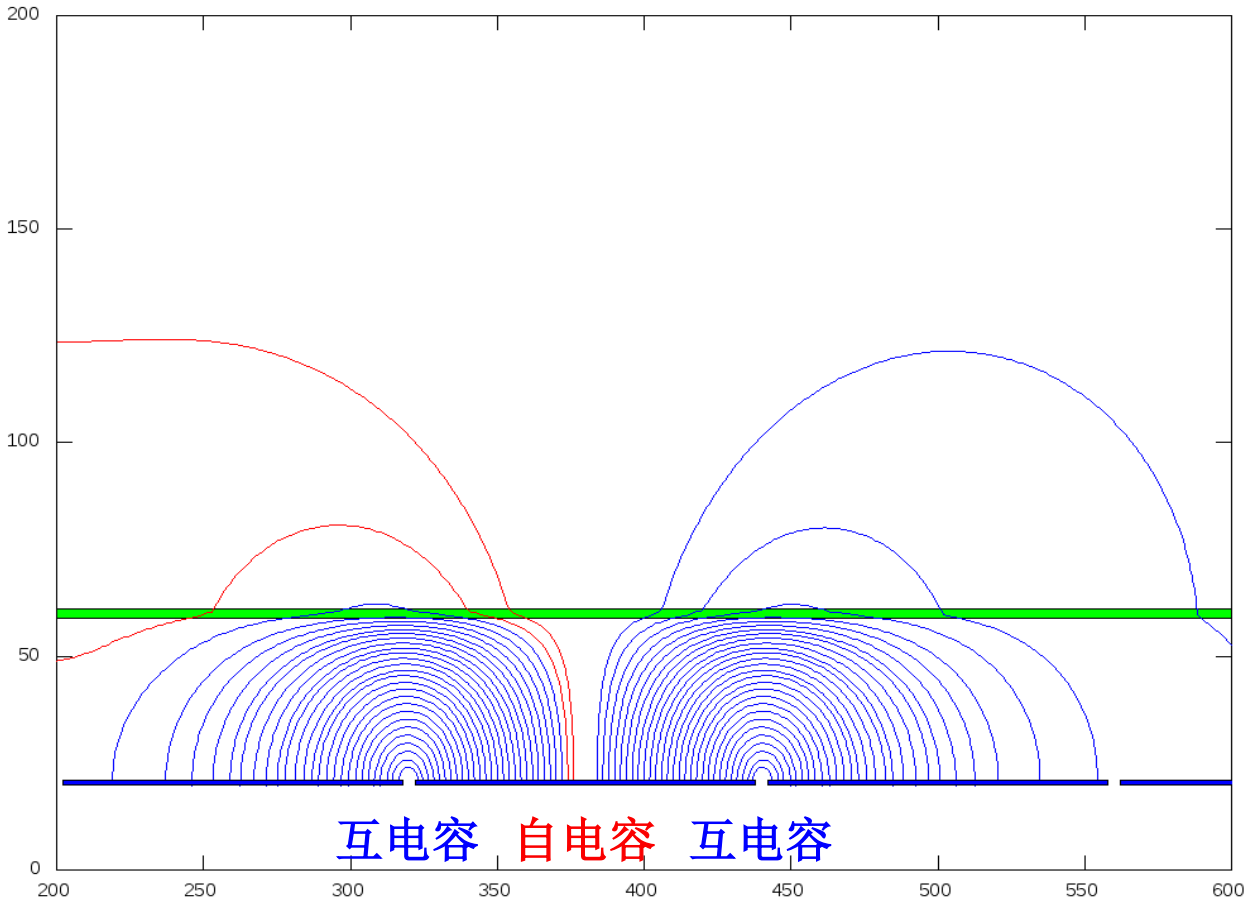


课程安排

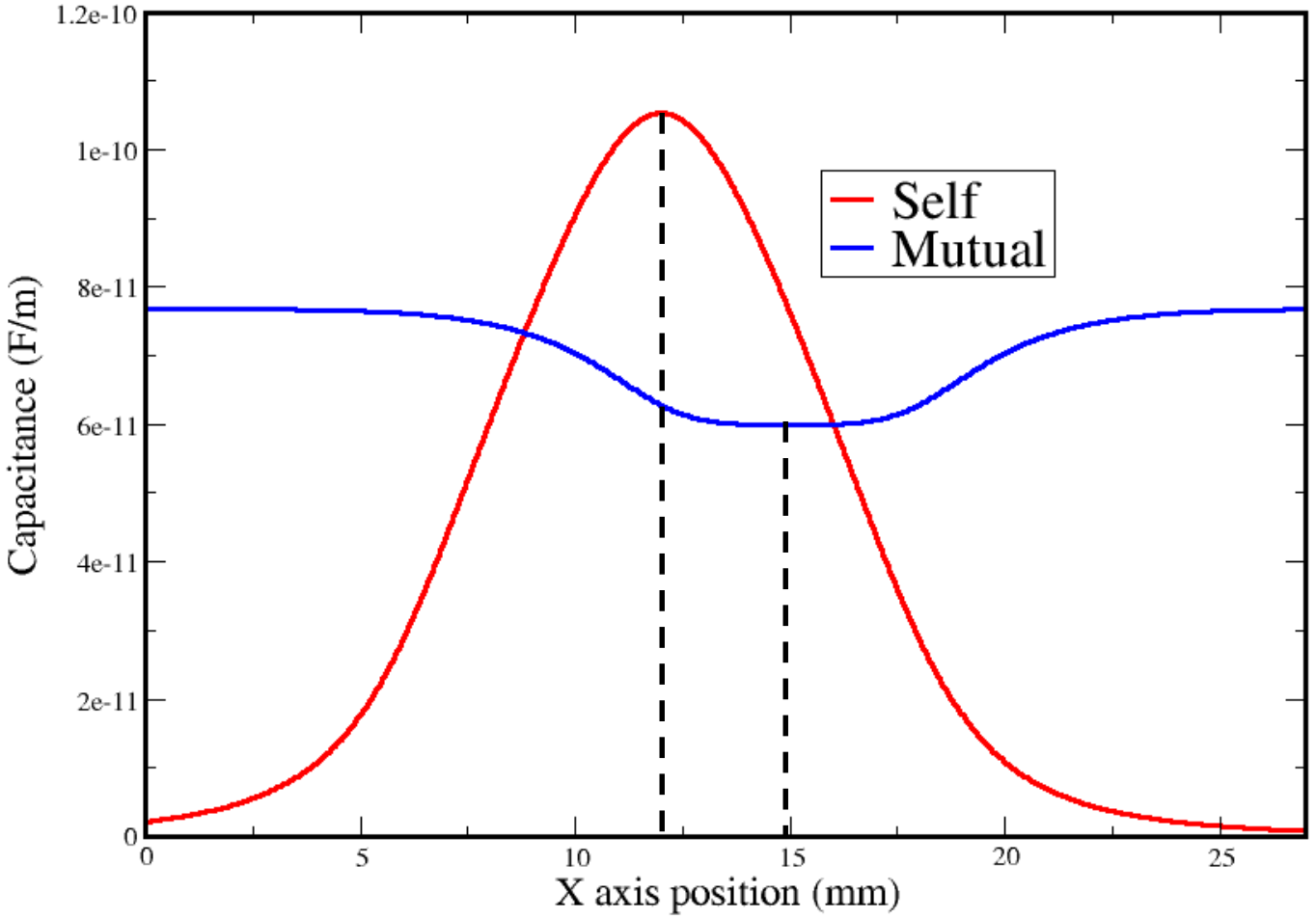
- 电容式传感电路模型基础
  - 自电容
  - 互电容

电容式传感物理学原理

耦合模拟

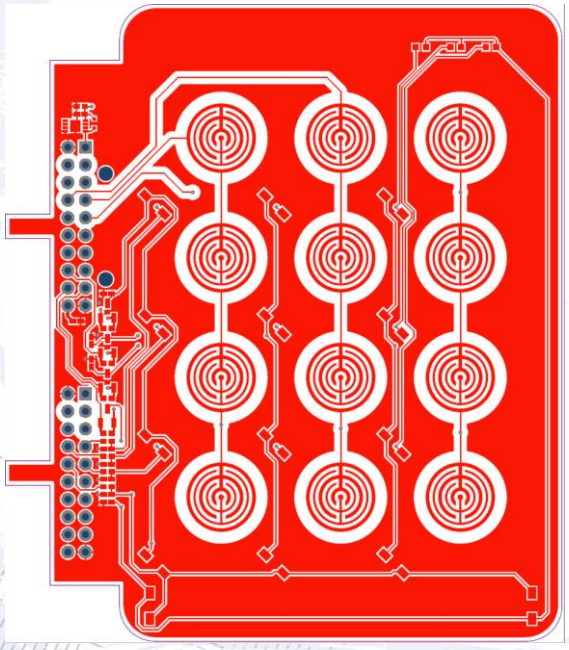


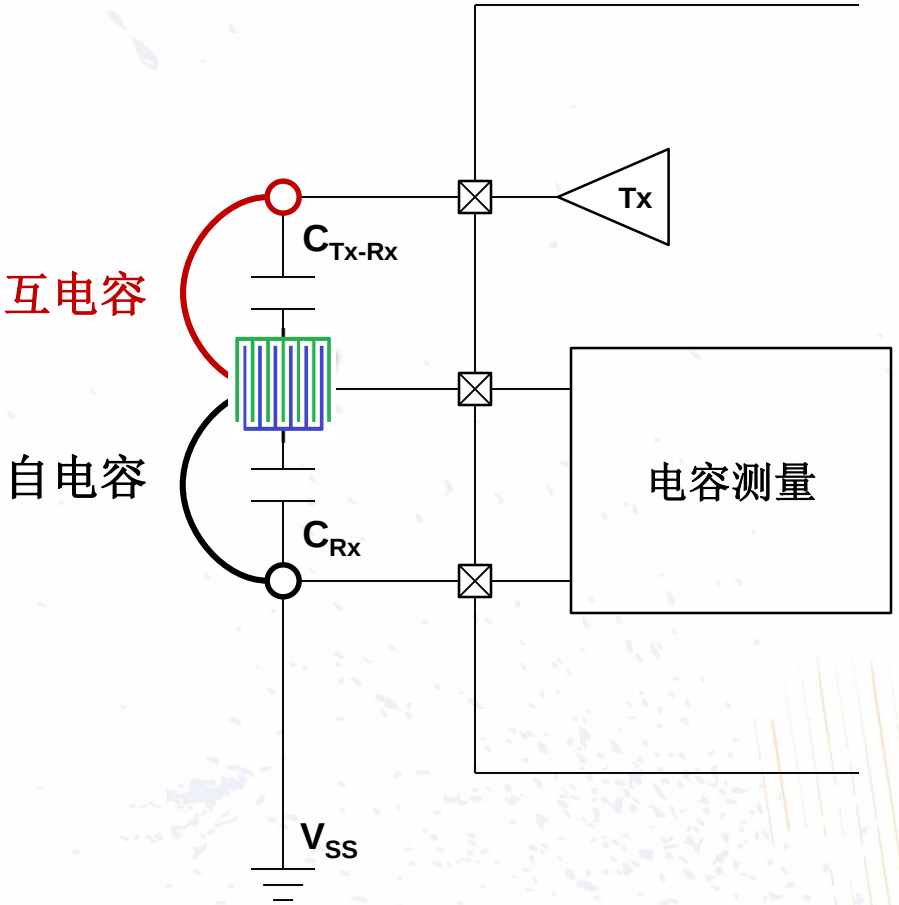
电容模拟



互电容

PCB示例





互电容

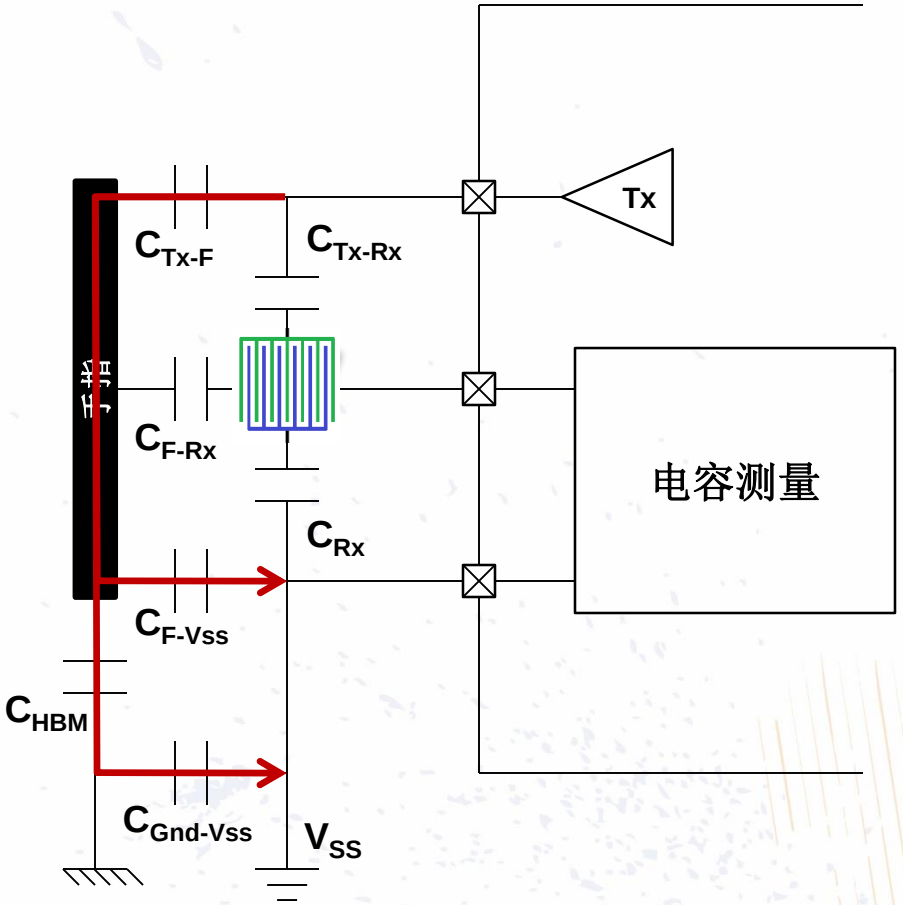
人体传导效应

电荷被导向地。

耦合量减少！

这是在互电容触摸设计中所期望的！！

将覆盖层放在传感器上，最大限度地增大人与系统地的耦合。



MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

噪声

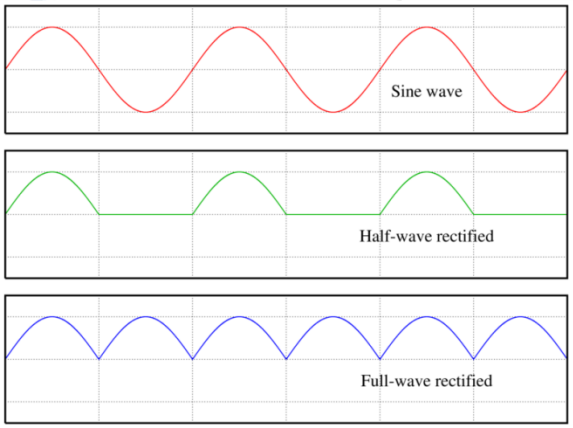
主要噪声来源

荧光灯

手机

无线电收发器

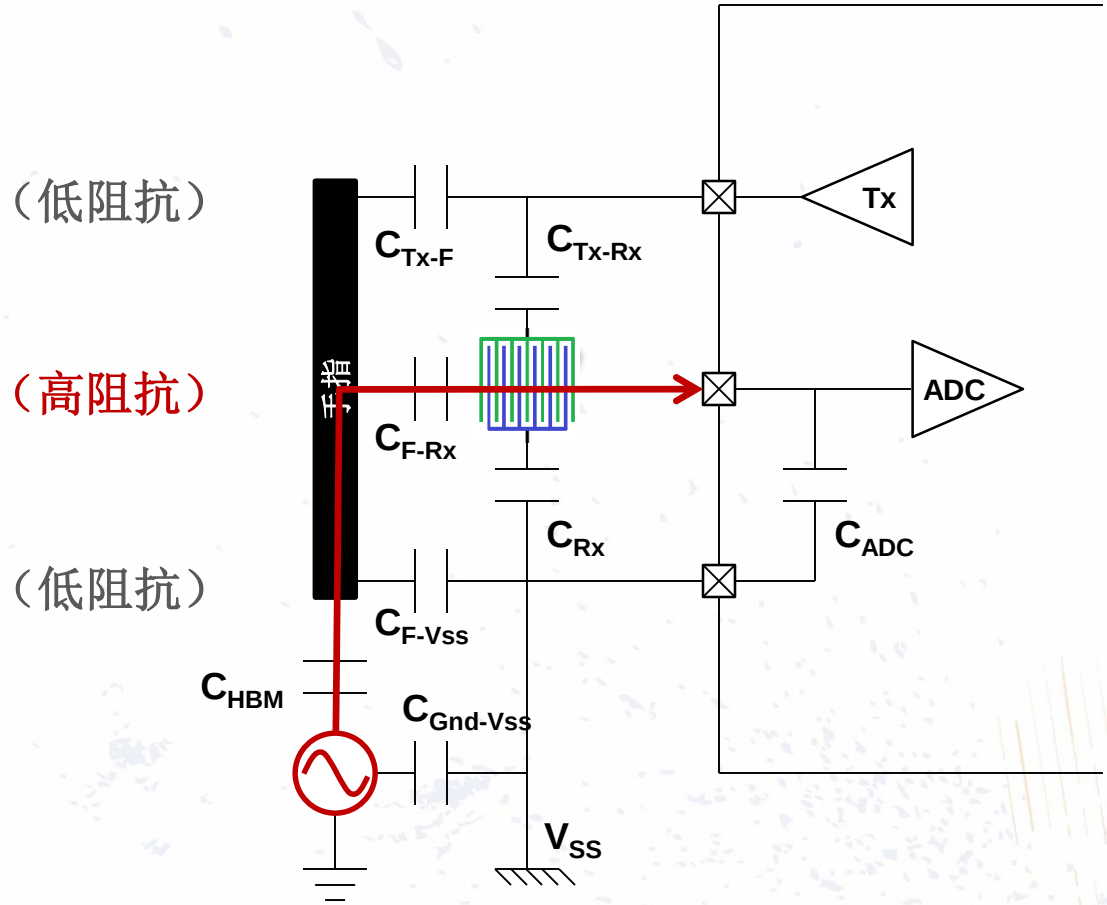
电机



有噪声的电源

50 Hz/60 Hz

噪声是如何引入的?


（低阻抗）
（高阻抗）
（低阻抗）
手指
C_{Tx-F}
C_{Tx-Rx}
C_{F-Rx}
C_{Rx}
C_{F-Vss}
C_{HBM}
$C_{Gnd-Vss}$
C_{ADC}
Tx
ADC
V_{SS}

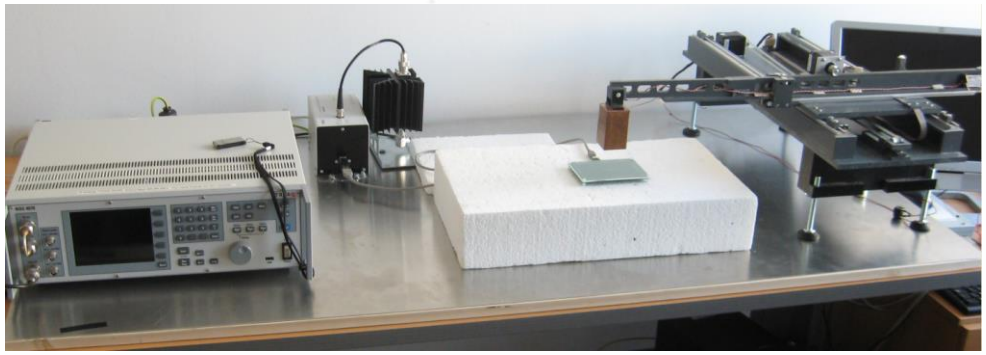
噪声
信噪比

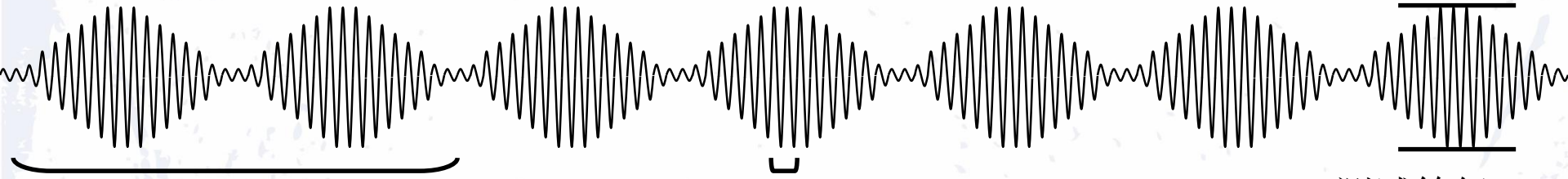
$$SNR = \frac{\text{最小信号}}{\text{最坏情况下的噪声}}$$

- 最小信号
  - 电池供电/隔离电源
  - 小手指/接近传感
  - 高跟鞋
  - 减去用户产生的串扰
- 最坏情况的噪声
  - 白噪声
  - 电路板产生的串扰
  - 传导噪声
  - 辐射噪声

噪声

传导噪声 :: IEC 61000-4-6





1 kHz，80%
幅值调制

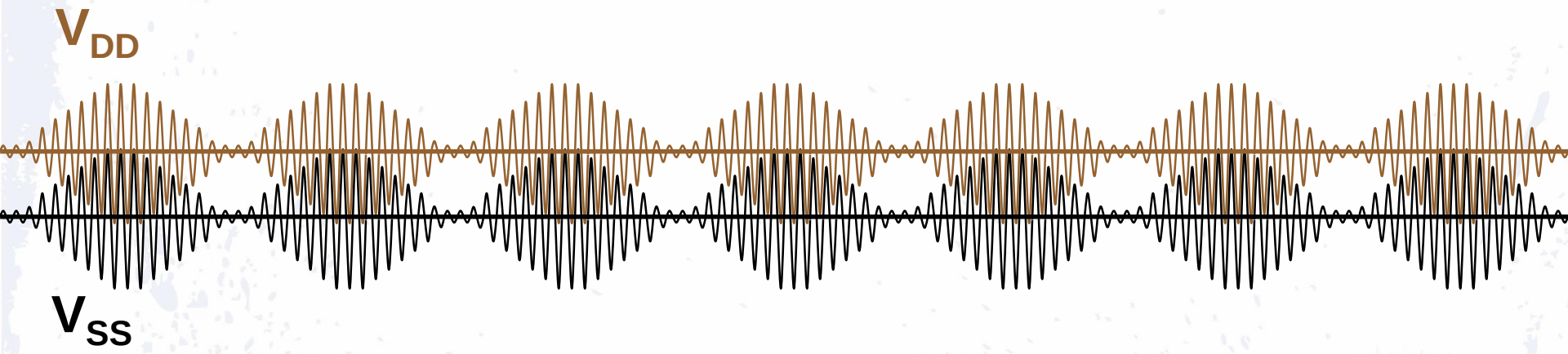
150 kHz–80 MHz
注入信号

测试等级

等级1 :: 1 V_{rms} :: 低辐射环境
等级2 :: 3 V_{rms} :: 商业环境
等级3 :: 10 V_{rms} :: 工业环境
等级X :: （Open）:: 定制

噪声

传导噪声 :: IEC 61000-4-6

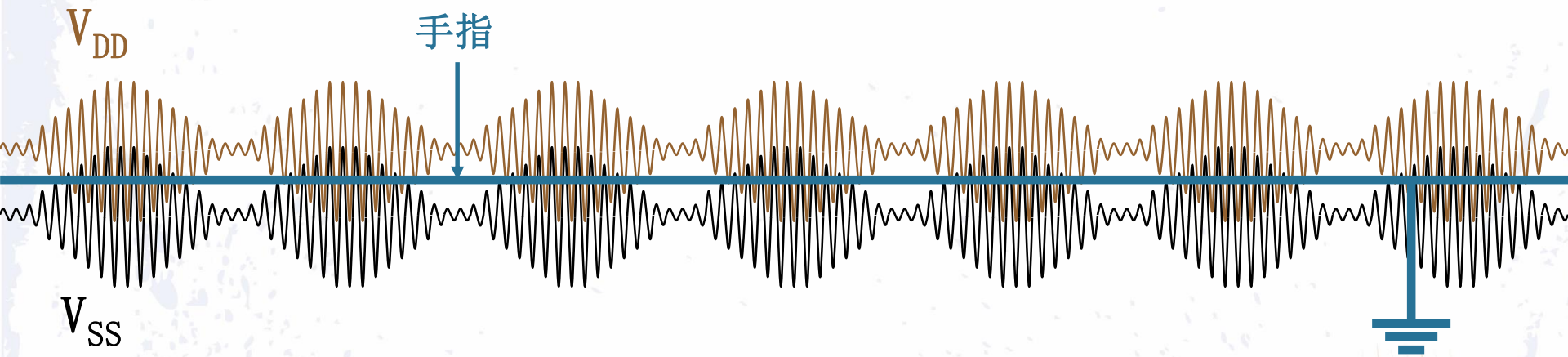


（$V_{DD} - V_{SS}$）

无变化！Vdd和Vss相对于彼此是稳定的，但相对于大地是浮动的。

噪声

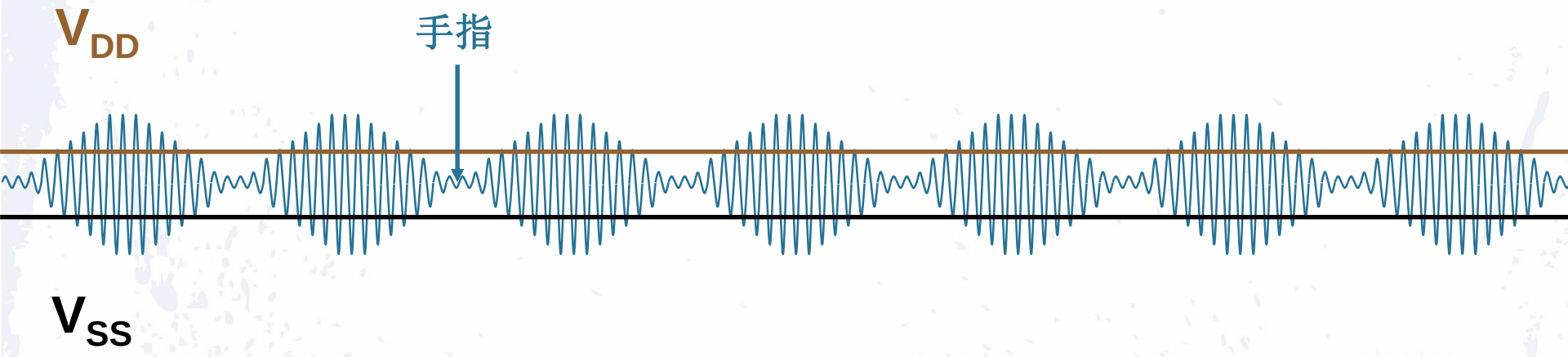
传导噪声 :: IEC 61000-4-6



触摸应用时，它相对于大地很稳定。

噪声

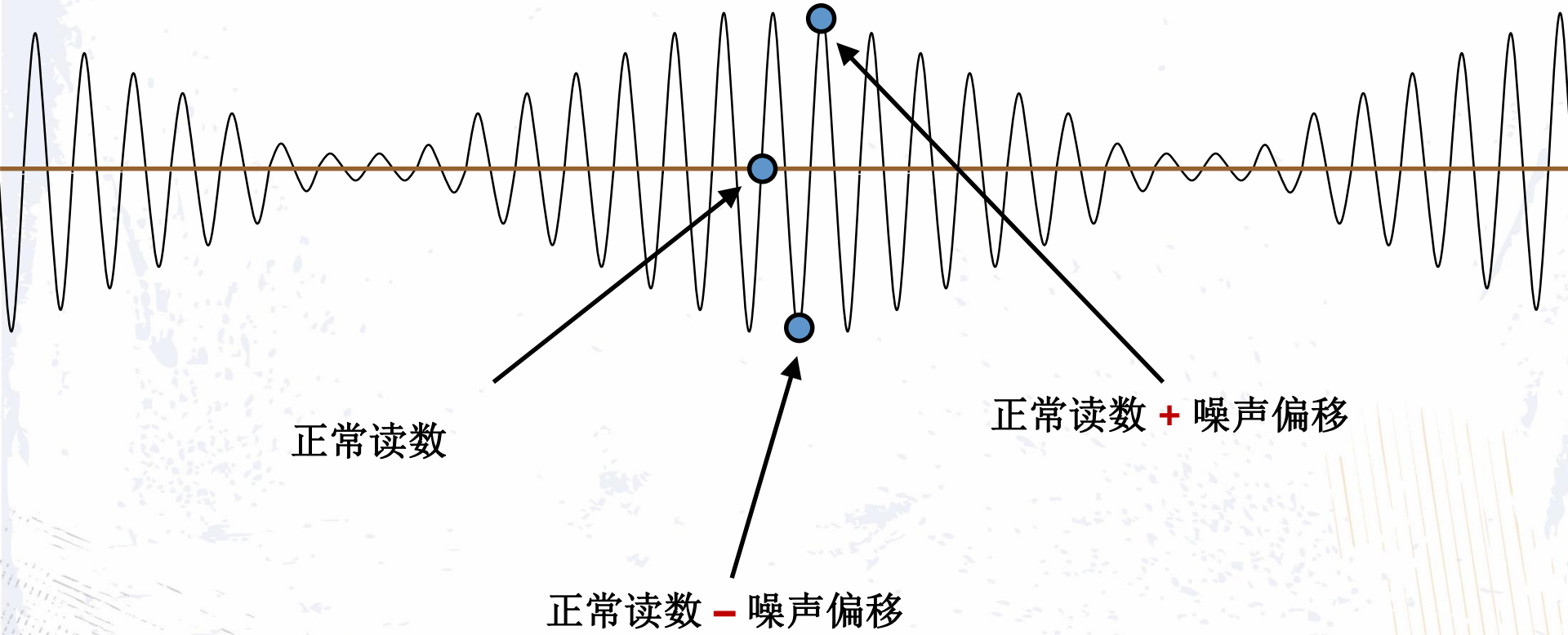
传导噪声 :: IEC 61000-4-6



因此，从系统的角度来看，当触摸应用时，就会注入噪声。

噪声

传导噪声 :: IEC 61000-4-6

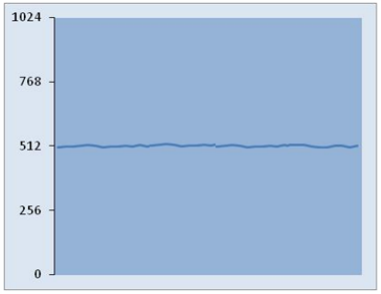


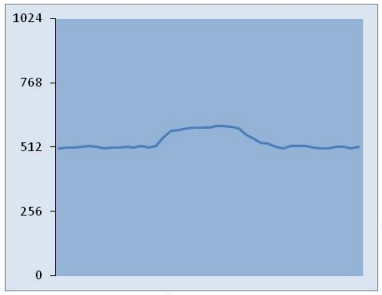
传导噪声

对触摸应用的影响：

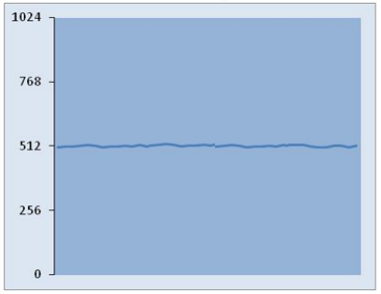
无触摸　　触摸

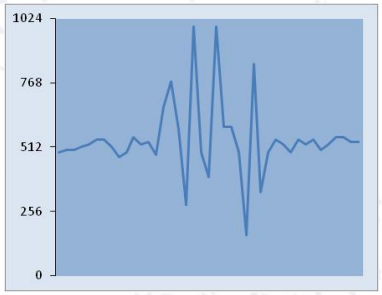
无传导噪声





有传导噪声

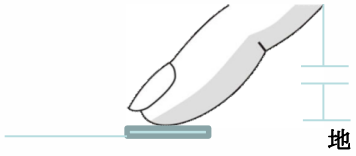




传导噪声仅影响被触摸的传感器的状态。

辐射噪声

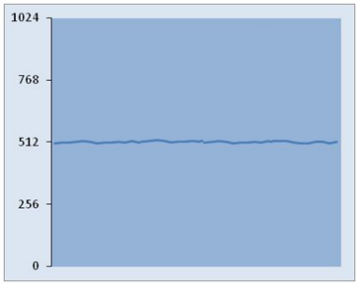
对触摸应用的影响：

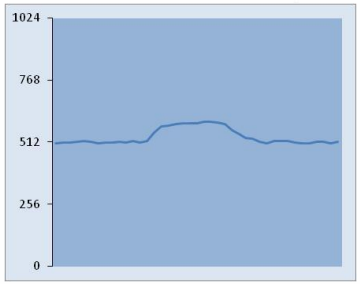


无触摸

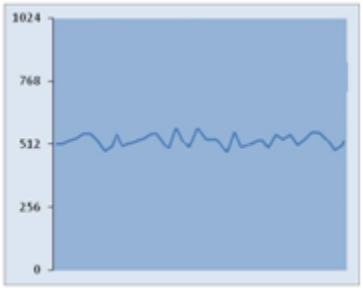
触摸

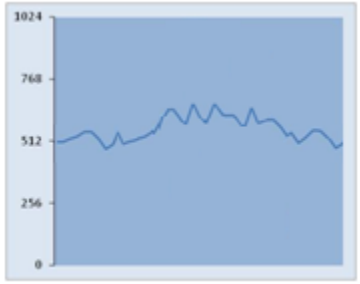
无辐射噪声





有辐射噪声





辐射噪声会影响被触摸的和未被触摸的传感器的状态。

噪声

传导噪声和辐射噪声的解决方案是相同的：

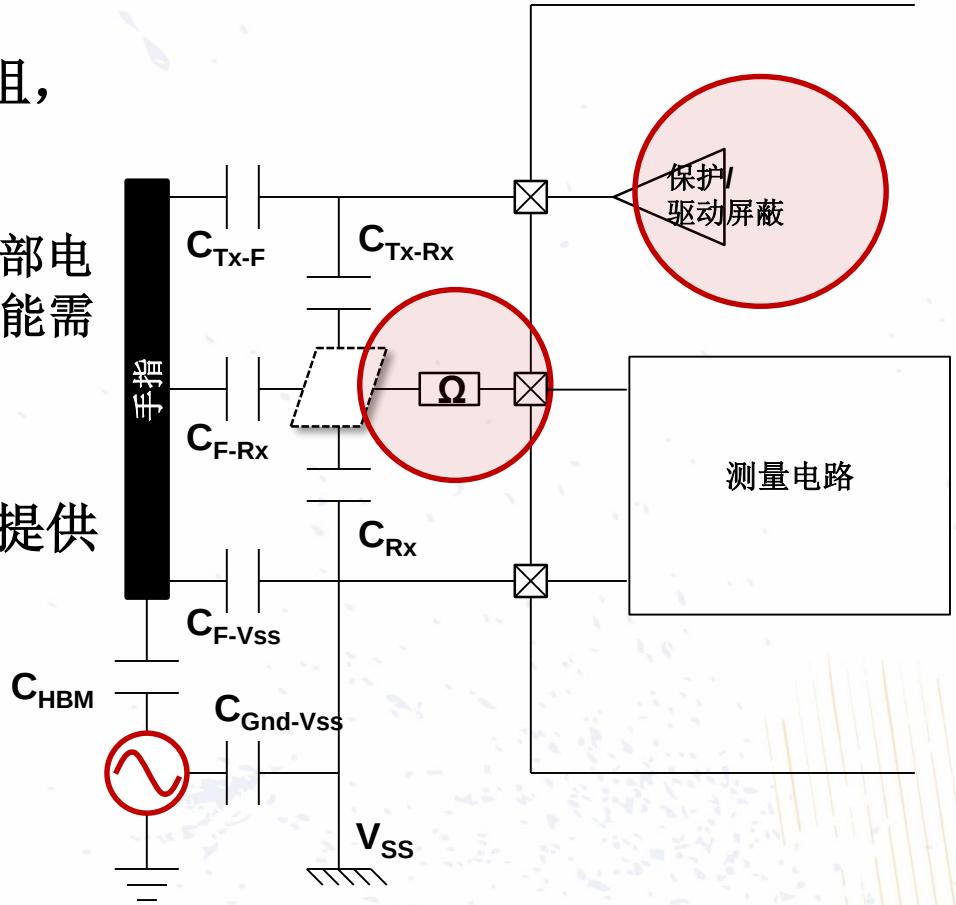
始终在传感器上放置一个串联电阻，以形成RC低通滤波器

- 最低：**10 kΩ**
  （如果采用PTC设计，请使用内部电阻，但还需添加焊盘，因为您可能需要更多。）
- 最高：**1 MΩ**

使用保护/驱动屏蔽来阻挡噪声并提供低电阻返回路径。

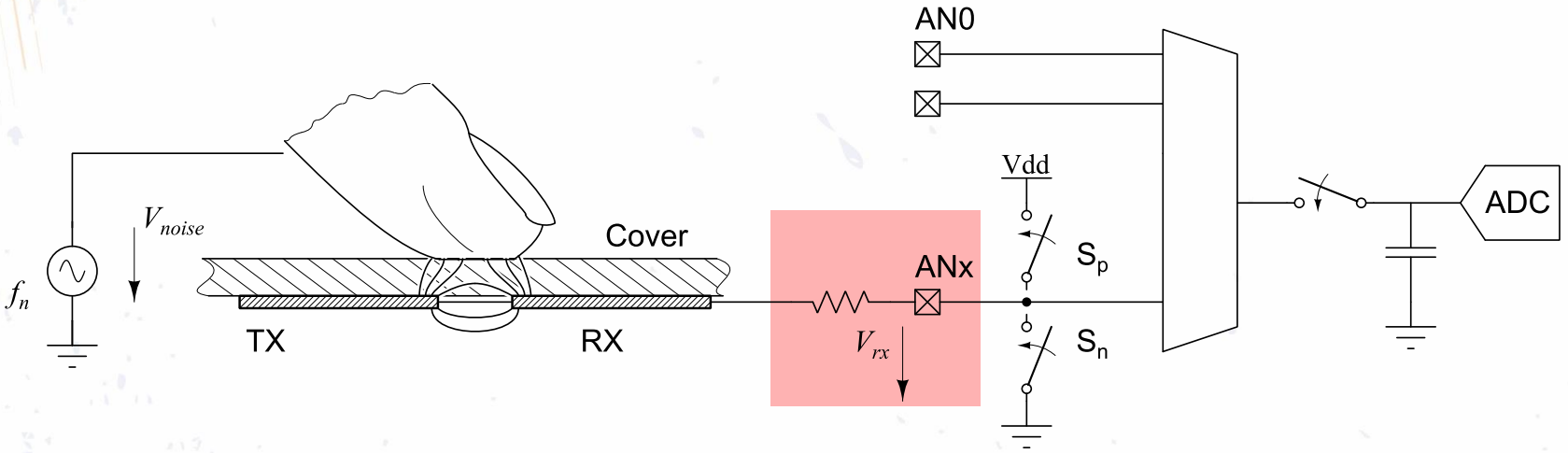
使用跳频。

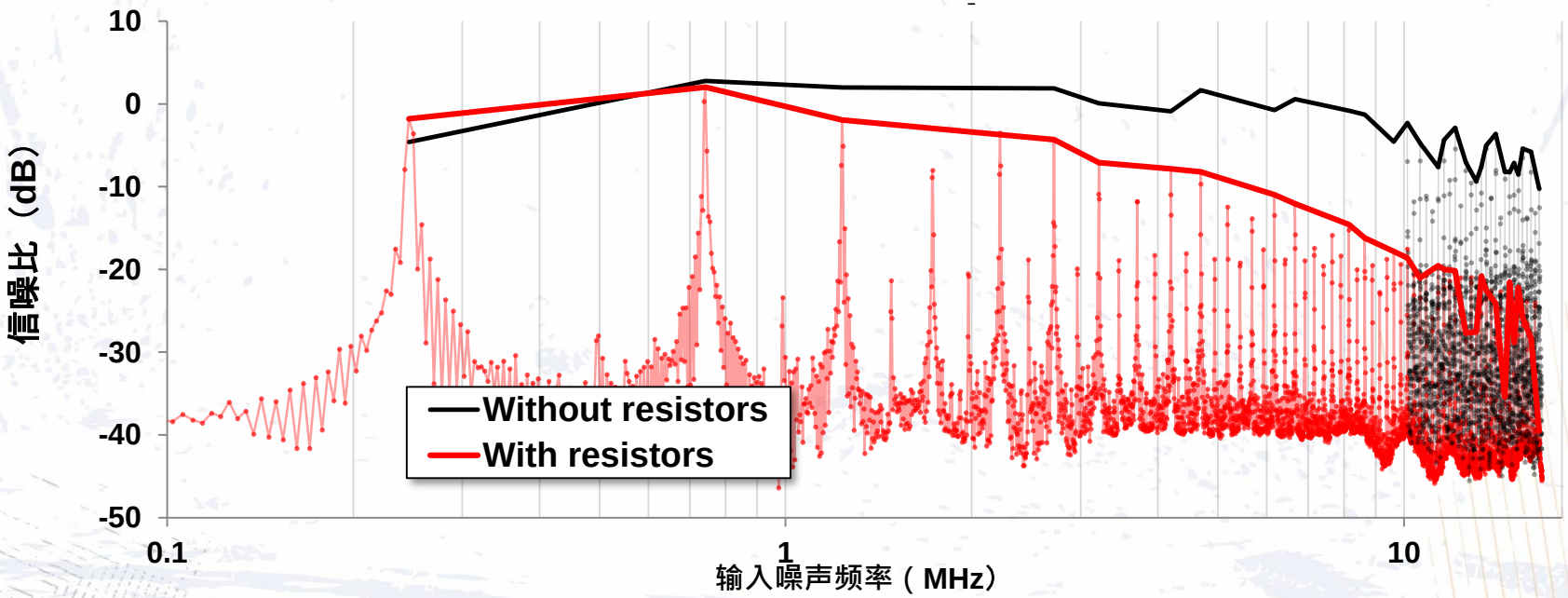
警告：
检查充电时间！



噪声

传导噪声 :: IEC 61000-4-6





抗噪声对策

传感器调试不充分会导致传感器性能不佳和抗扰度测试失败...*所以应了解应用中的噪声源！！！*

解决方案不是单一的，而是通过多种技术在抑制噪声和实现可接受的系统性能之间进行折衷。

噪声对策	滤波效果最好的噪声
串联电阻	传导噪声
滤波长度	辐射/传导噪声
自动过采样	传导噪声
跳频	辐射/传导噪声
检测阈值/滞后	辐射/传导噪声
检测积分	辐射噪声

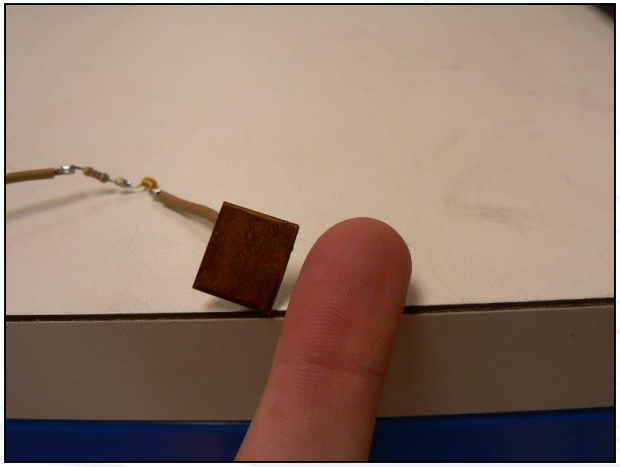
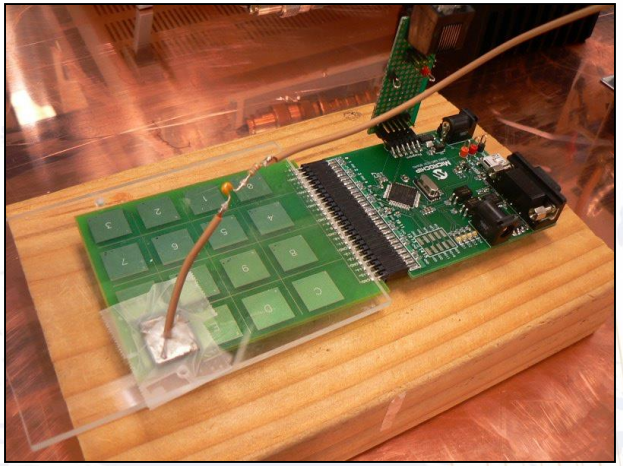
噪声
测试注意事项

- 测试可以更改系统的行为！
  - 用户是否接地？
  - 测试设备是否接地？
- 检查各频率下的“按下”行为
- 不期望的系统行为
  - 虚假触发
  - 按钮失效
  - 传感器波动
  - 灵敏度增大/减小

噪声
假手指

用于产生重复手指按压

- 请勿直接接地
- Microchip使用...
  - 1.6 kΩ电阻与220 pF电容串联
  - ~1m长度

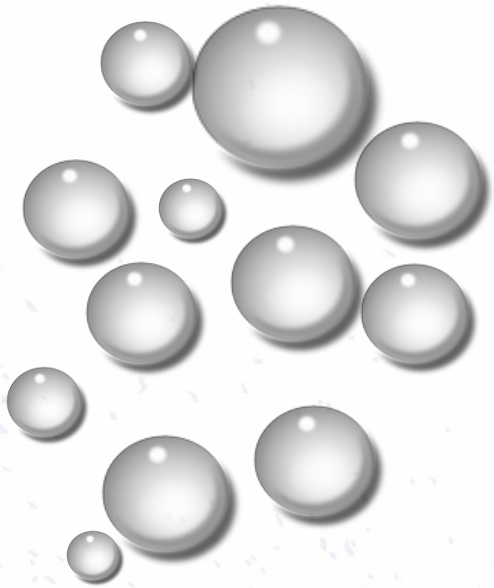




课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

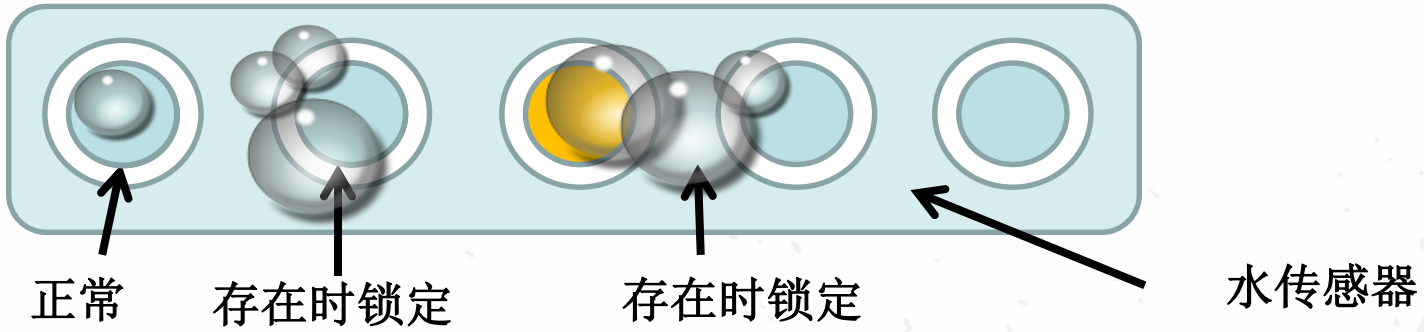
防水问题说明

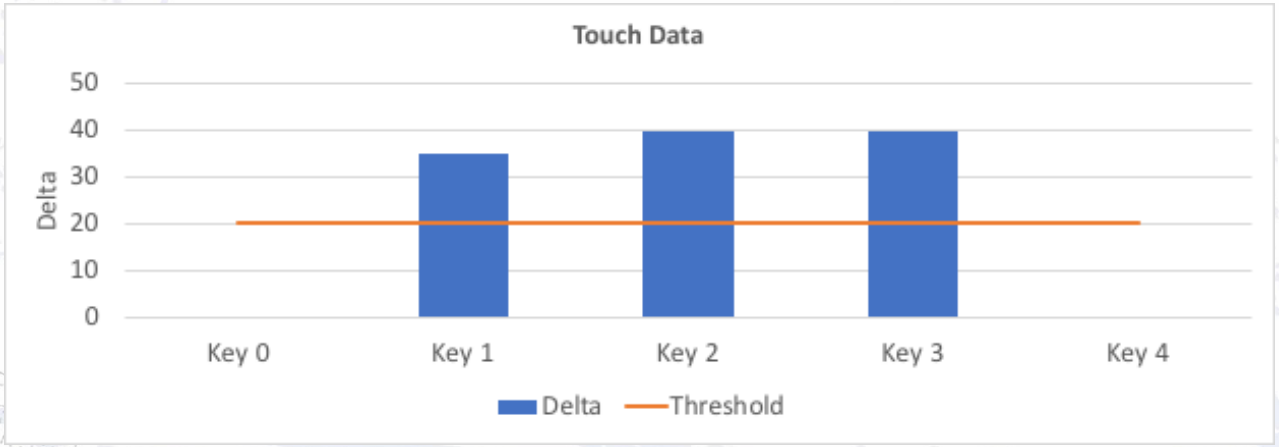
- **传递机制**
  - 传感器覆盖层上有凝露
  - PCB上有凝露
  - 传感器覆盖层有雨水或溢水
- **现象**
  - 对触摸无反应
  - 虚假触摸
- **处理水的技术**
  - 锁定（最不理想）——水传感器
  - 完整操作——驱动屏蔽



水传感器——锁定

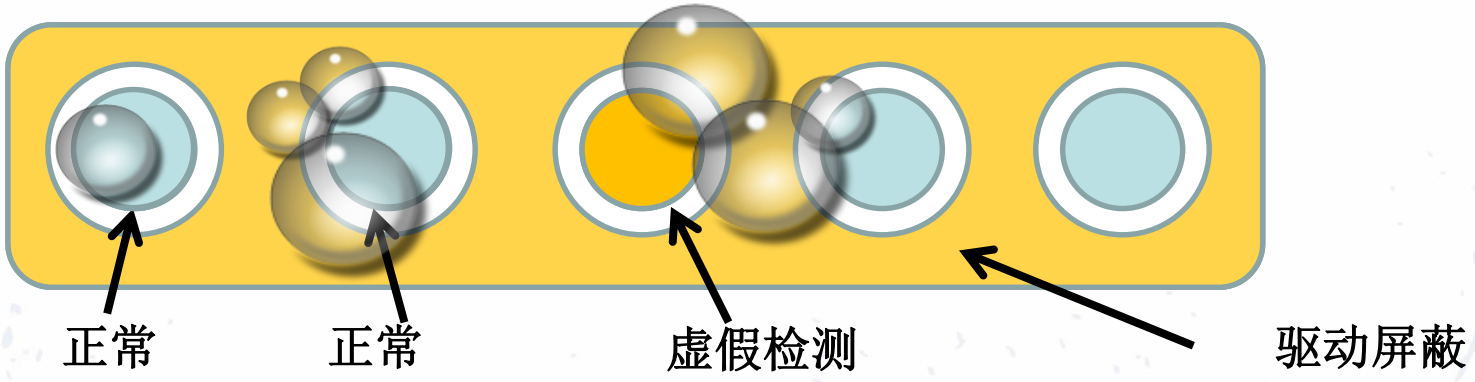
- 所有电极周围有一个额外的大电极
- 所有自电容传感器和水传感器环之间采用电容式触摸传感

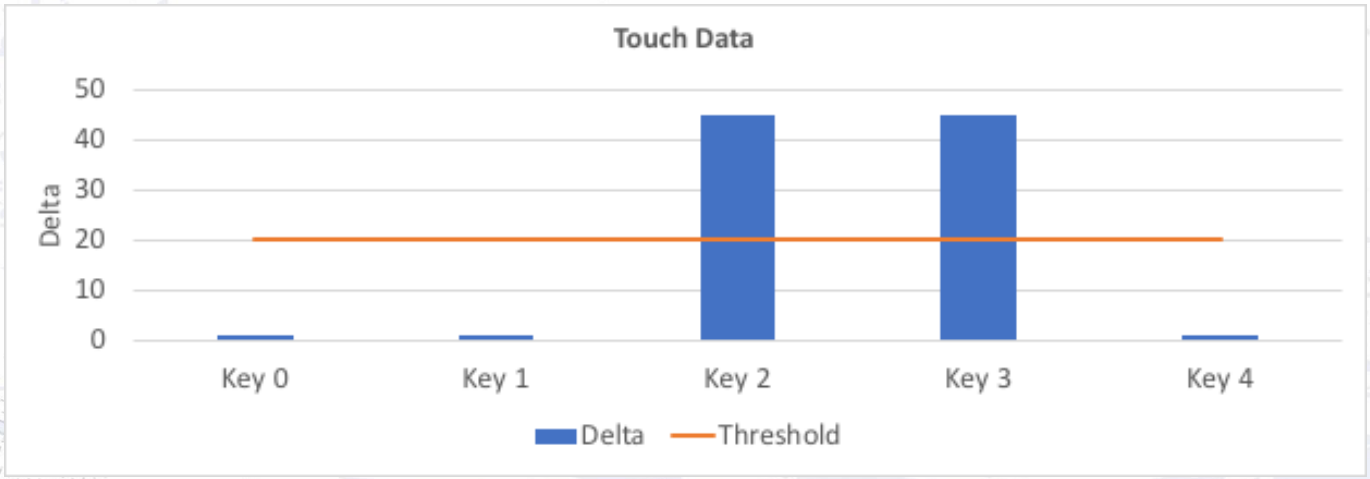




简单的驱动屏蔽

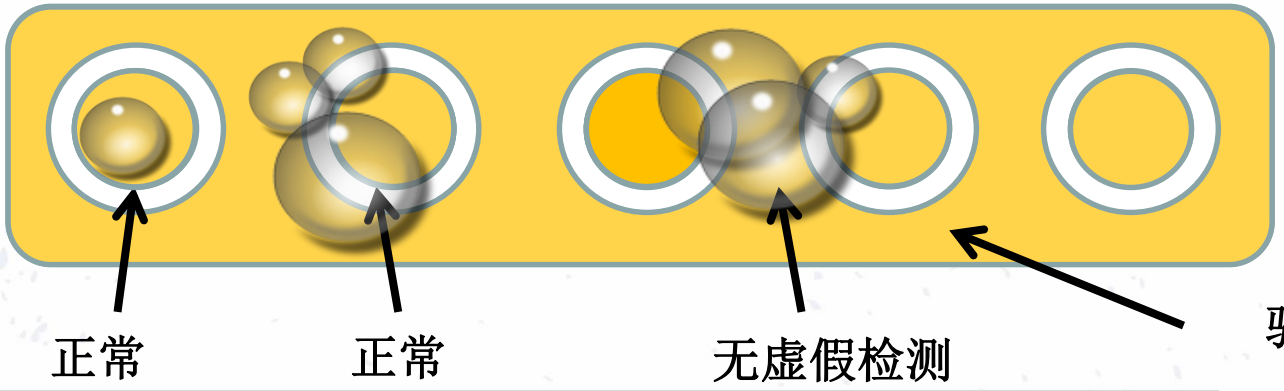
- 驱动屏蔽电极与被扫描电极相同的电位

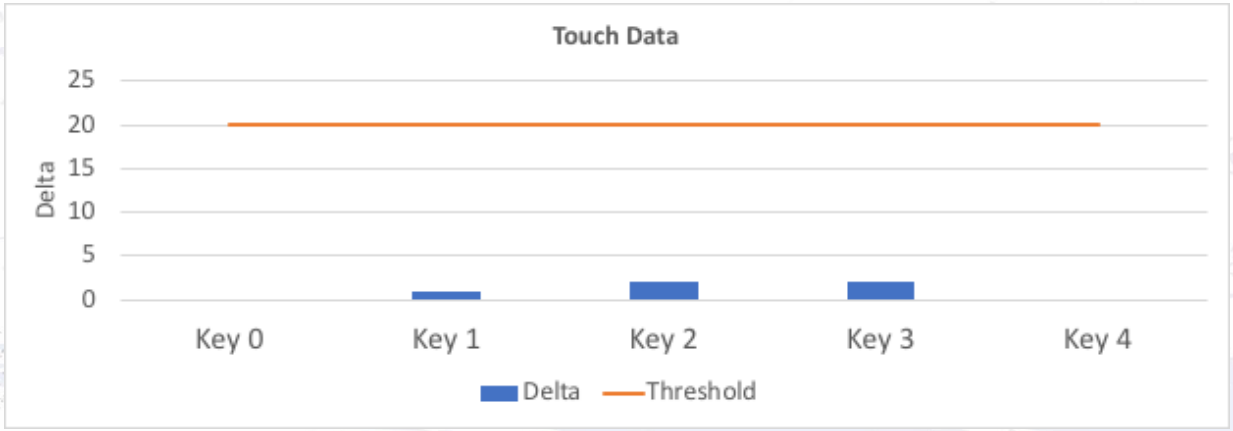




增强型高级驱动屏蔽

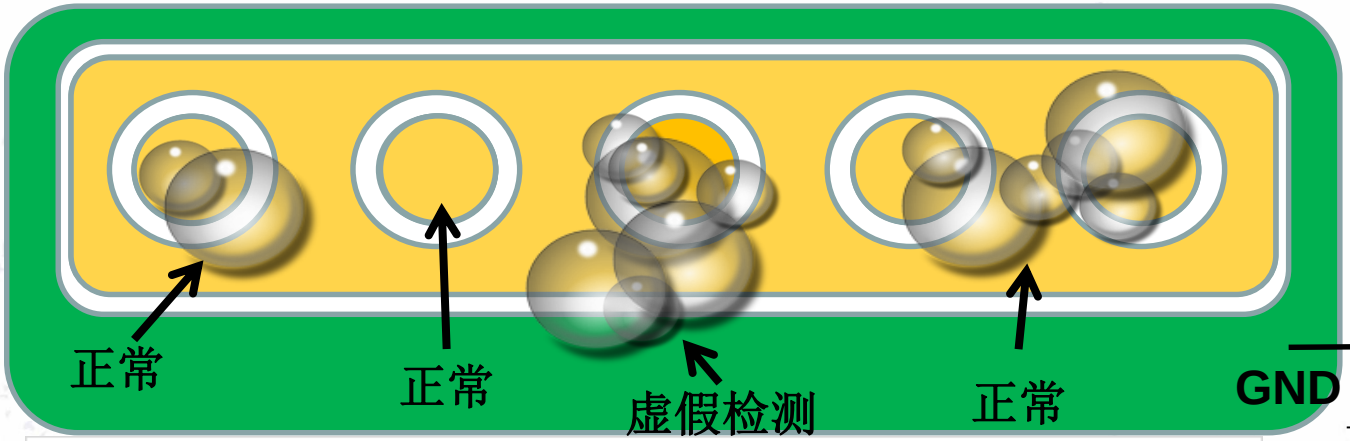
- 驱动屏蔽电极和所有其他电极与被扫描电极相同的电位

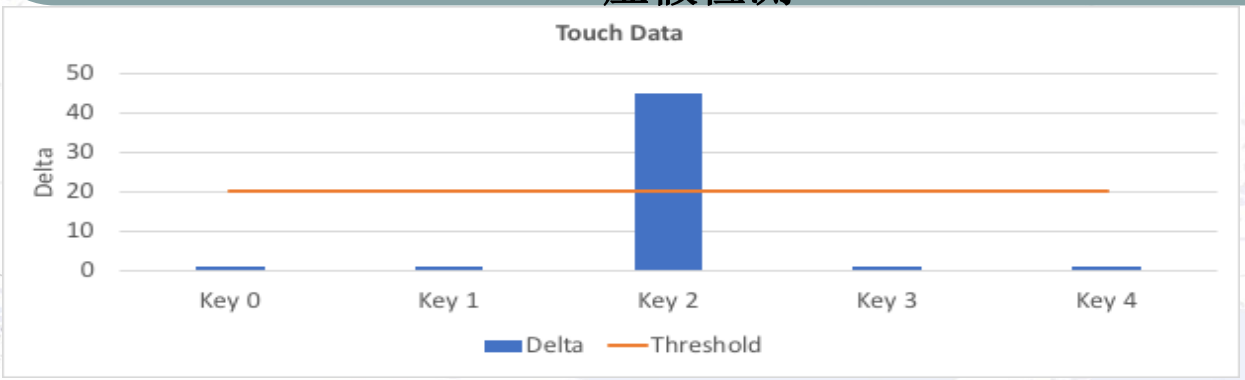




增强型高级驱动屏蔽——接地问题

- 如果屏蔽电极和所有其他电极以与被扫描电极相同的电位驱动但耦合到GND，则将导致虚假检测





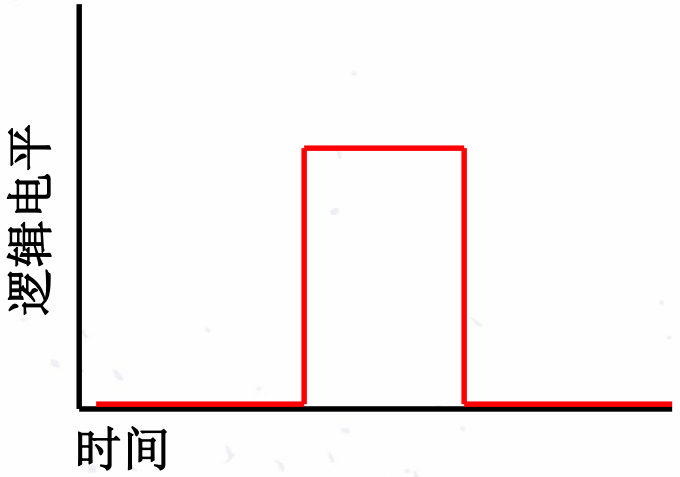
课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

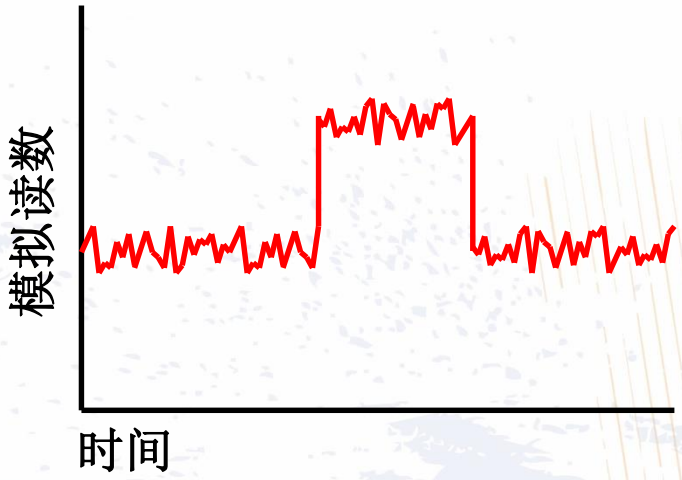
低功耗设计挑战

机械按钮与电容式传感器

- 机械按钮可通过电平变化产生中断在掉电模式（休眠）下工作。
  - PIC16LF1xxx掉电电流~80 nA @ 3V

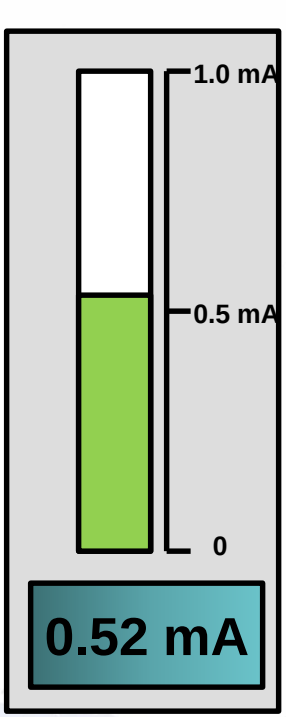


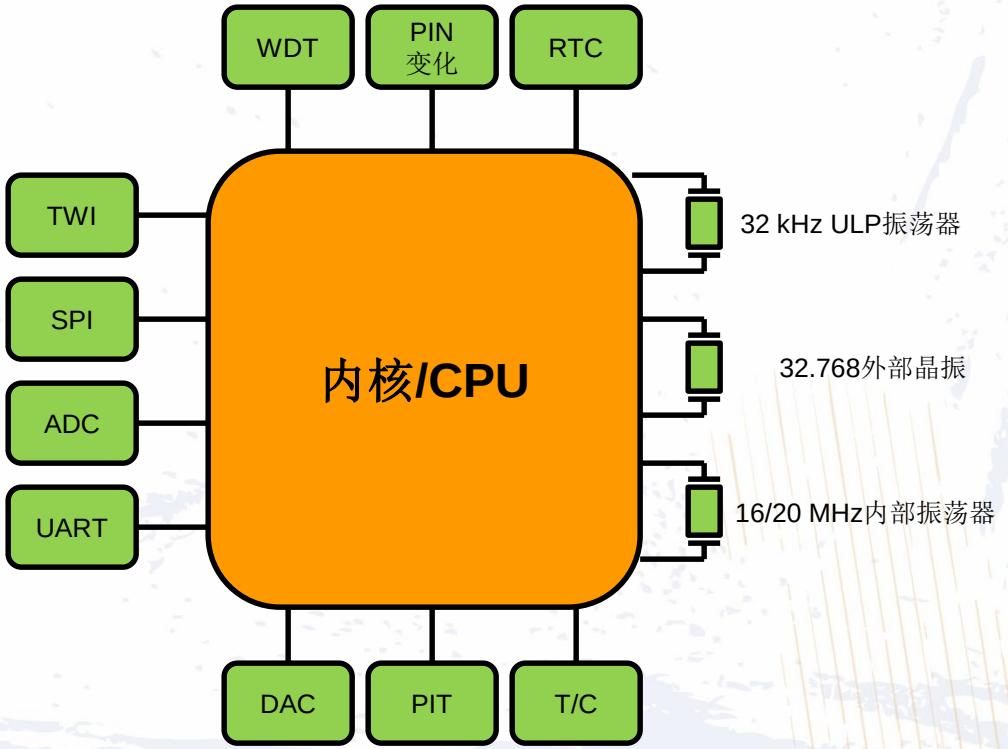
- 电容式传感需要持续扫描以及与基准值进行比较。
  - PIC16LF1xxx工作电流：~2.1 mA @ 3V，Fosc =32 MHz



低功耗设计基础

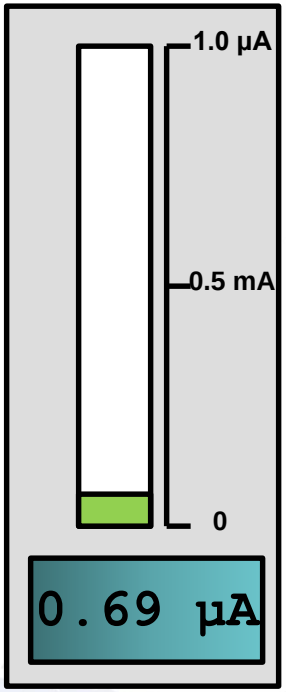
- 休眠模式用于关闭外设和时钟域以实现节能目的。
- 空闲模式：
  - CPU时钟停止。外部时钟仍然运行并提供给所有外设，以便立即唤醒。

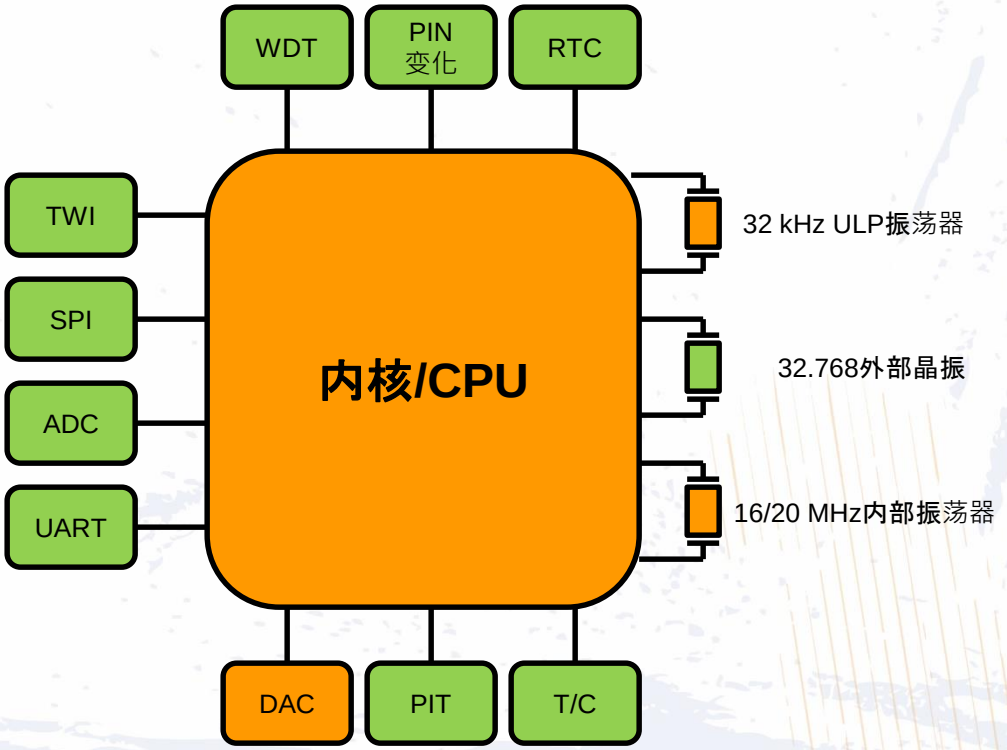




低功耗设计挑战

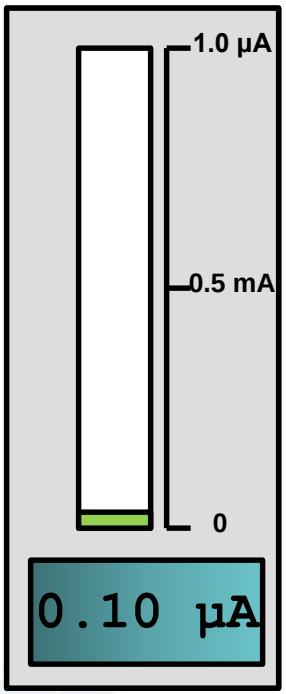
- 待机模式：
  - 用户可以配置使能还是禁止外设。
  - **ADC可以置于sleepwalking模式**

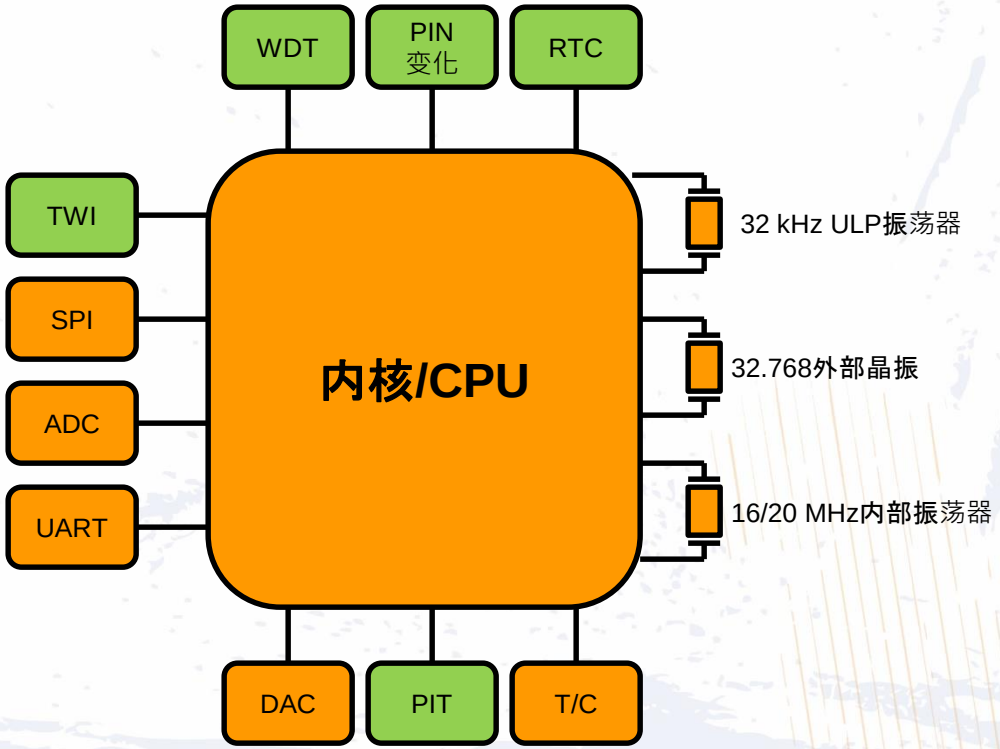




低功耗设计基础

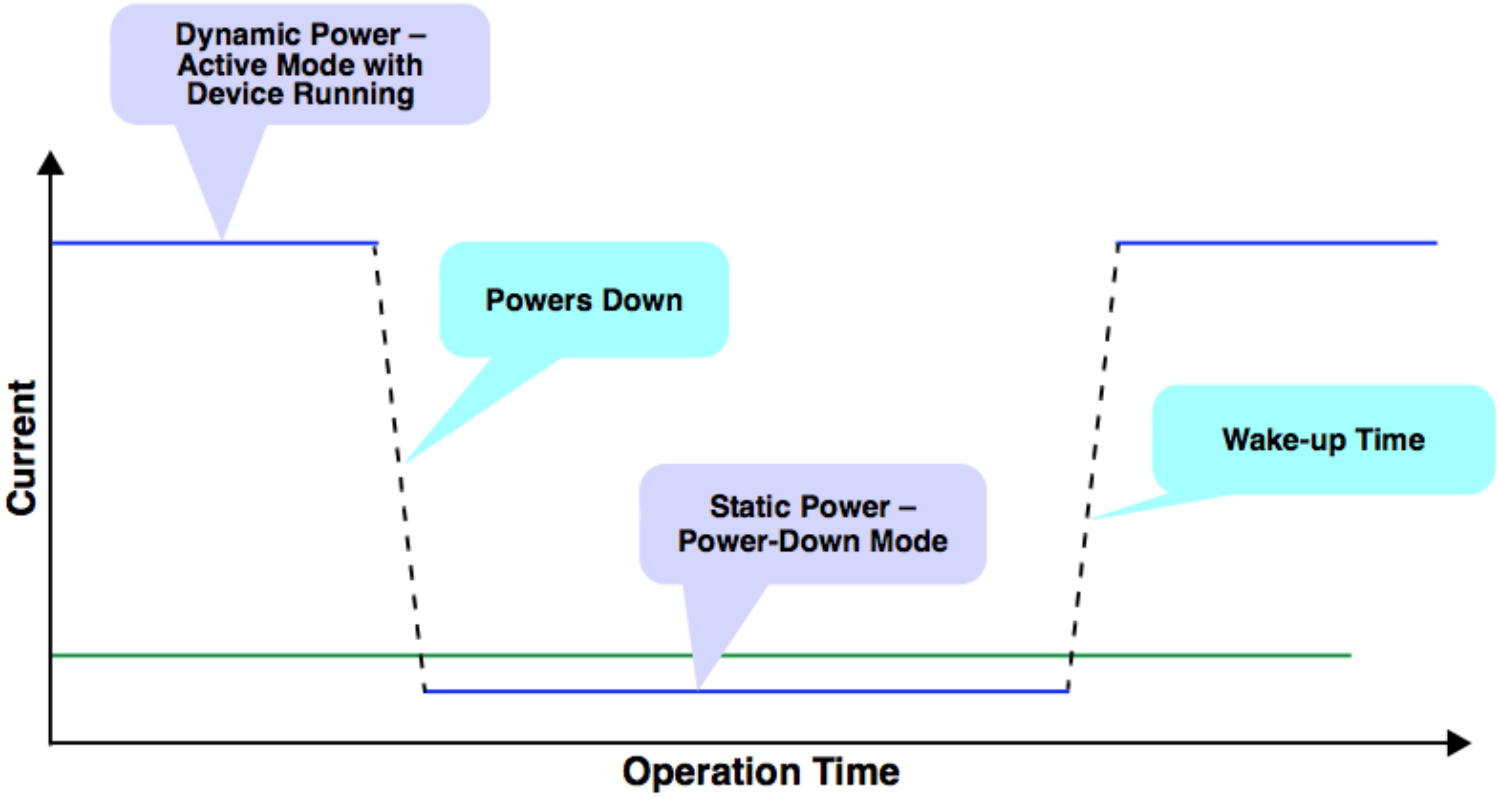
- 掉电：
  - 只有**WDT**和**RTC**处于工作状态。惟一的唤醒源是外部引脚电平变化中断和**TWI**地址匹配。





低功耗设计注意事项

计算系统平均电流



$$Average\ \ Current\ = \frac{I_{active} \times T_{active}\ +\ I_{powerdown} \times T_{powerdown}}{T_{active} + T_{powerdown}}$$

低功耗设计基础

如何降低功耗

- 应用通用技术来实现低功耗设计
  - 降低V_{DD}
  - 无浮动I/O
  - 尽可能使用较大的上拉电阻
  - 优化代码
  - 关闭所有未使用的外设
- 应用笔记*AN1416《低功耗设计指南》*

$$P_{dyn} = K \cdot V_{CC}^2 \cdot f$$

K = 常量，取决于过程
F = 开关频率

低功耗设计基础第二部分

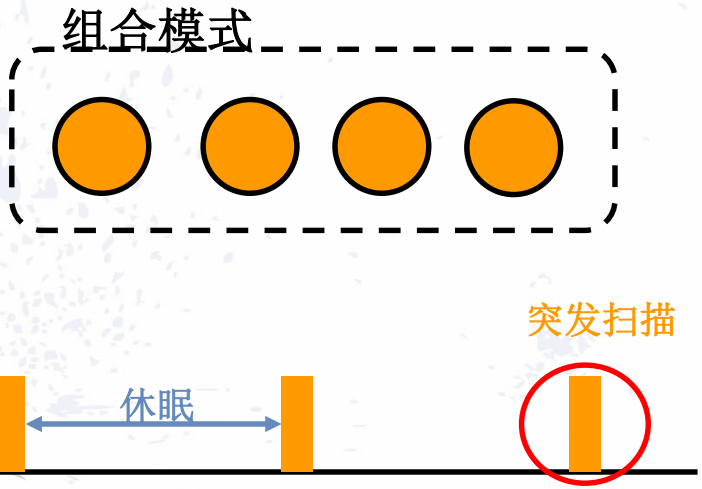
- 电容式触摸特定的注意事项：
  - 设计系统，最大程度减少后处理滤波。这意味着：

1. 使用优质电源，最好是LDO。电源噪声耦合到电容式触摸测量，需要更多滤波。
2. 使用推荐尺寸的传感器。小型传感器需要更多增益和更多滤波周期。
3. 尝试抑制和屏蔽噪声。使电容式触摸走线远离含噪声的走线，并根据需要使用屏蔽（50%交叉排线）。

低功耗固件设计

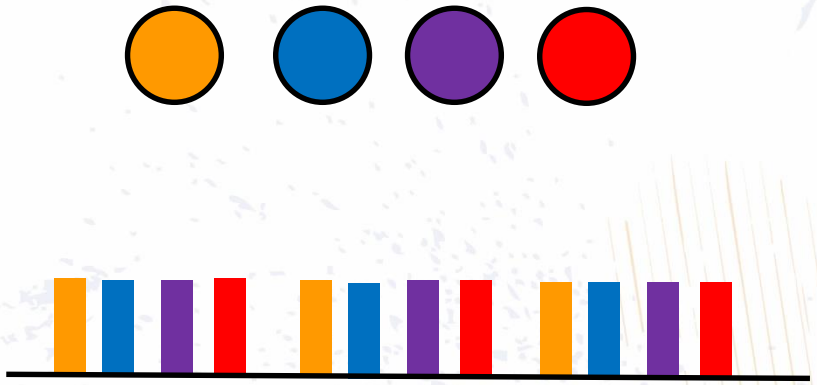
低功耗模式

- 在两次扫描之间休眠
- 仅扫描接近传感器
- 将所有传感器组合在一起



全功耗模式

- 单独扫描所有传感器
- 超时后返回低功耗模式



低功耗固件设计
软件驱动

```
void main( )
{
   Init();

   while(1)
   {
         switch(PowerState)
         {
               case
LOWPOWER:
               {...}
               case ACTIVE:
               {...}
         }
   }
}
```

LOWPOWER：

- 扫描并监视“唤醒”传感器
- 关闭所有未使用的外设后，将器件置于休眠状态
- 使用定时器中断定期唤醒器件
- 仅在唤醒后扫描“唤醒”传感器，并对比阈值检查数据
- 如果数据高于阈值，切换到**ACTIVE**。

ACTIVE：

- 扫描所有传感器
- 运行应用代码
- 特定事件（超时，键断电）后切换回**LOWPOWER**

低功耗固件设计
事件驱动

```
void main( )
{
    Init();

    while(1)
    {
       switch(PowerState)
       {
              case LOWPOWER:
              {...}
              case ACTIVE:
              {...}
       }
    }
}
```

LOWPOWER：

- 扫描并监视“唤醒”传感器
- 关闭所有未使用的外设后，将器件置于休眠状态
- 触摸外设（HCVD和PTC）将由定时器使用内置阈值比较自动触发。
- 如果数据高于阈值，则切换到**ACTIVE**。（由硬件完成）

ACTIVE：

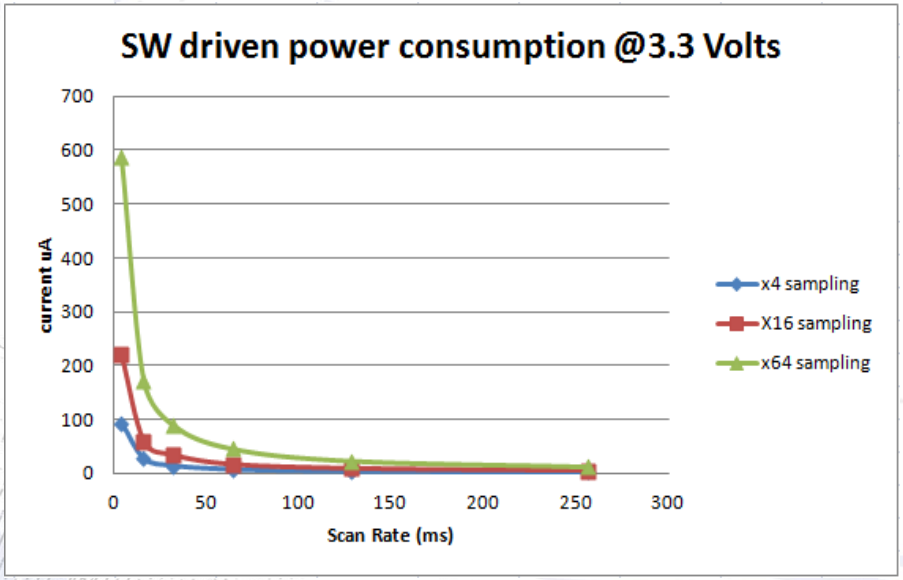
- 扫描所有传感器
- 运行应用代码
- 特定事件（超时，键断电）后切换回**LOWPOWER**

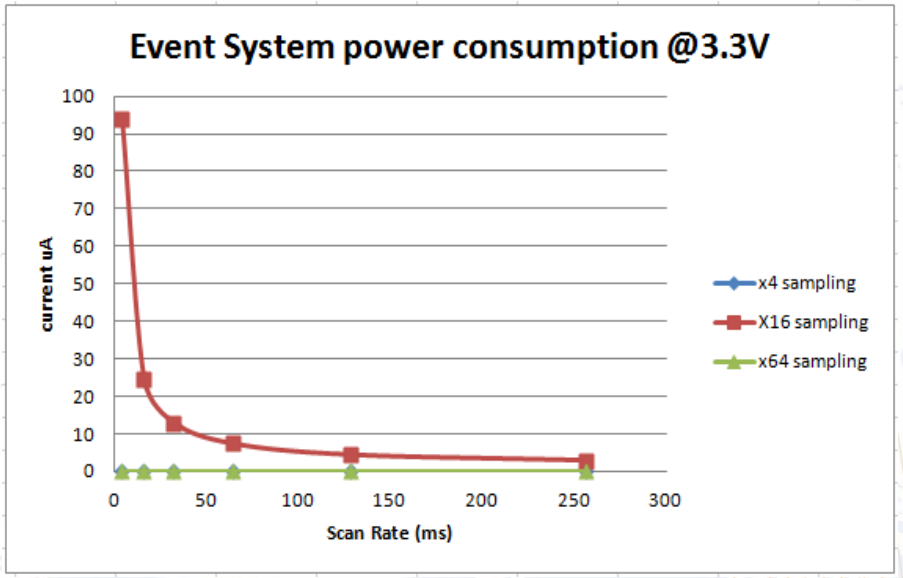
MICROCHIP MASTERs 2018

ATtiny3217低功耗

Table-1 Software Driven (uA)			
1 Key / 3.3V / CPU @ 8MHz / Prescaler = 2 / CSD = 4, Drift wakeup = 2 seconds and Driven shield = Disabled			
Scan rate (ms)	Oversampling		
	x4	x16	x64
256	2.71	4.81	12.7
128	4.16	8.74	23.6
64	7.79	16.5	46.2
32	14.7	33.7	89.9
16	28.1	60.4	173.1
4	93.1	219.7	586

Table-2 Event Driven (uA)			
1 Key / 3.3V / CPU @ 8MHz / Prescaler = 2 / CSD = 4, Drift wakeup = 2 Seconds and Driven shield = Disabled			
Scan rate (ms)	Oversampling		
	x4	x16	x64
256	1.9	3.64	11.9
128	2.46	5.56	20.6
64	3.65	7.48	44.5
32	6.02	16.2	74.5
16	10.6	30.5	167.2
4	37.8	114.9	573

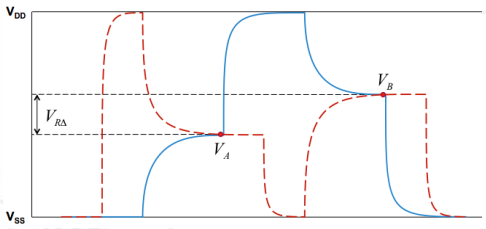


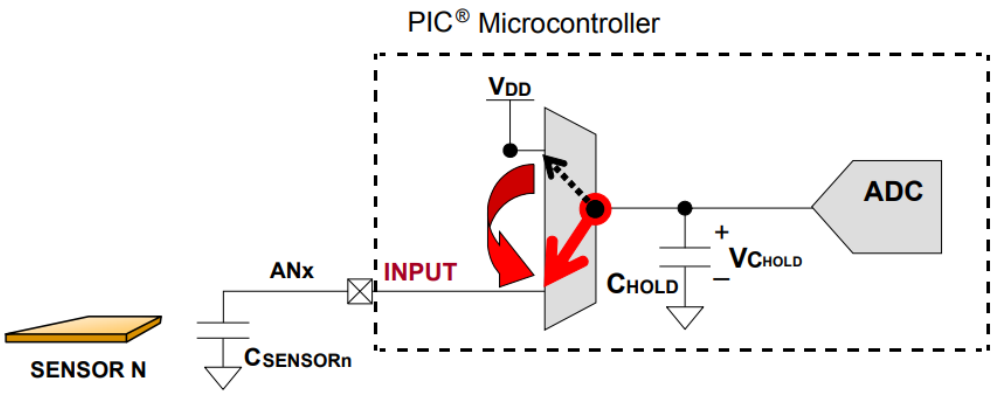


什么是CVD？

mTouch®技术：CVD

- 电容分压器（CVD）不使用外部元件。很多PIC®器件具有硬件CVD（HCVD）或带计算功能的ADC（ADC2），可自动实现触摸传感。
- 信号采集的执行方式是在传感器与测量电路之间充电和共用电荷，然后进行ADC和信号处理。

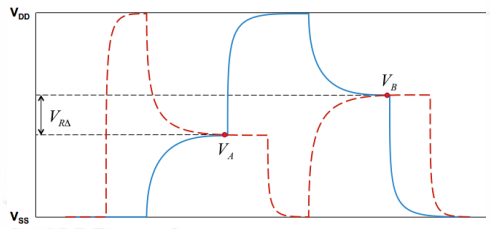


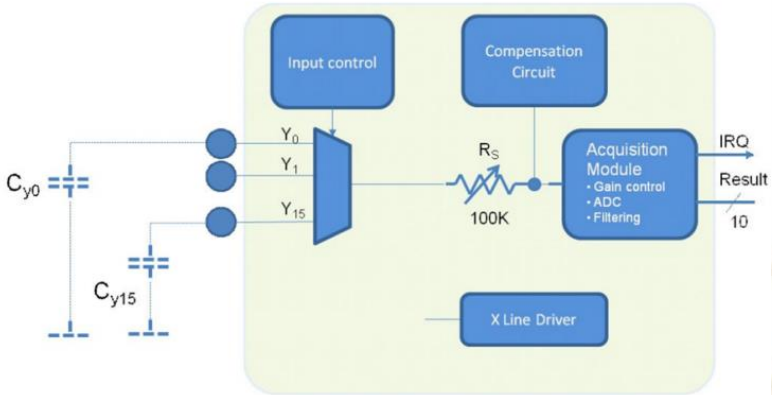


什么是PTC？

QTouch®技术：PTC

- 外设触摸控制器（PTC）不使用外部元件。很多AVR® MCU和SAM MCU以及MPU均具有PTC，可自动实现触摸传感。
- 信号采集的执行方式是在传感器与测量电路之间充电和共用电荷，然后进行ADC和信号处理。

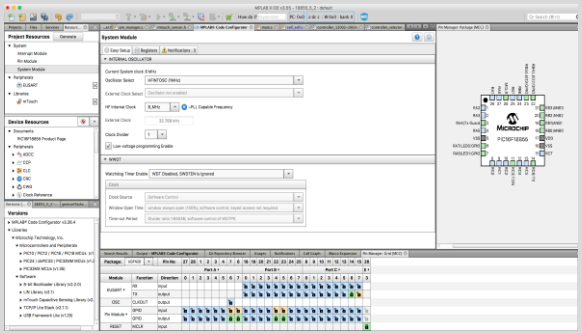




为什么选择Microchip来实现触摸应用？

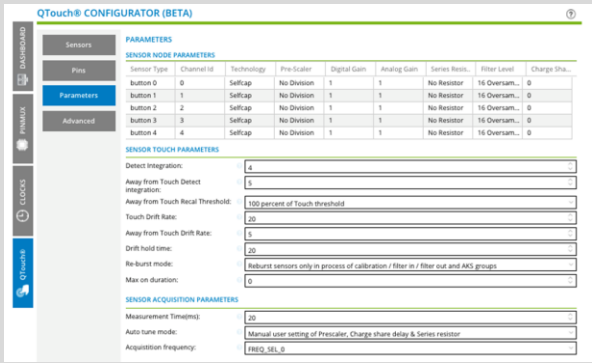
MCC + MPLAB® IDE

- PIC® MCU



START + Studio

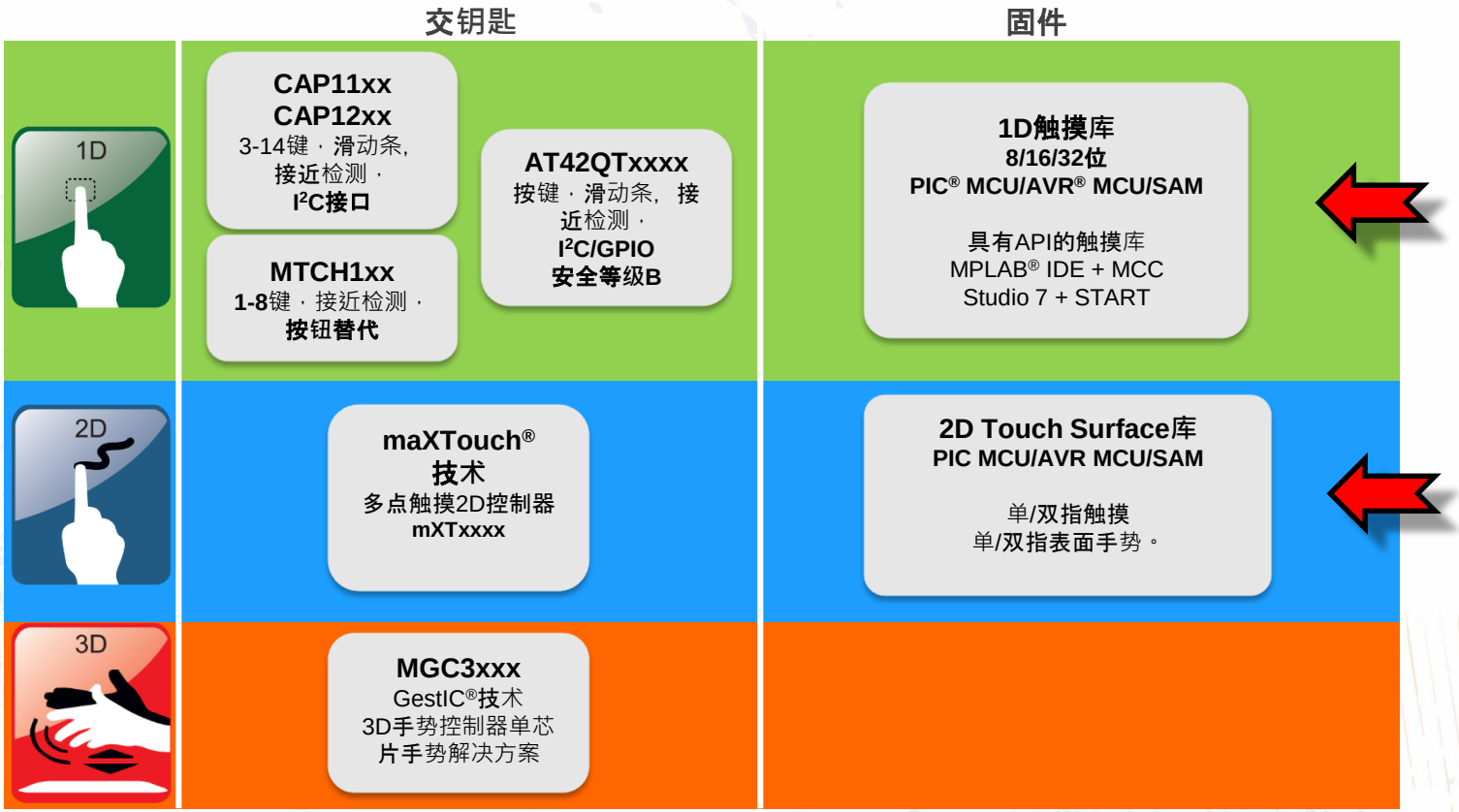
- AVR® MCU/SAM



“以前我要花费几天的时间来研究数据手册，但现在我的工程师们创造出多种解决方案，可以在几小时内开始测试。项目周转率已经提高。”

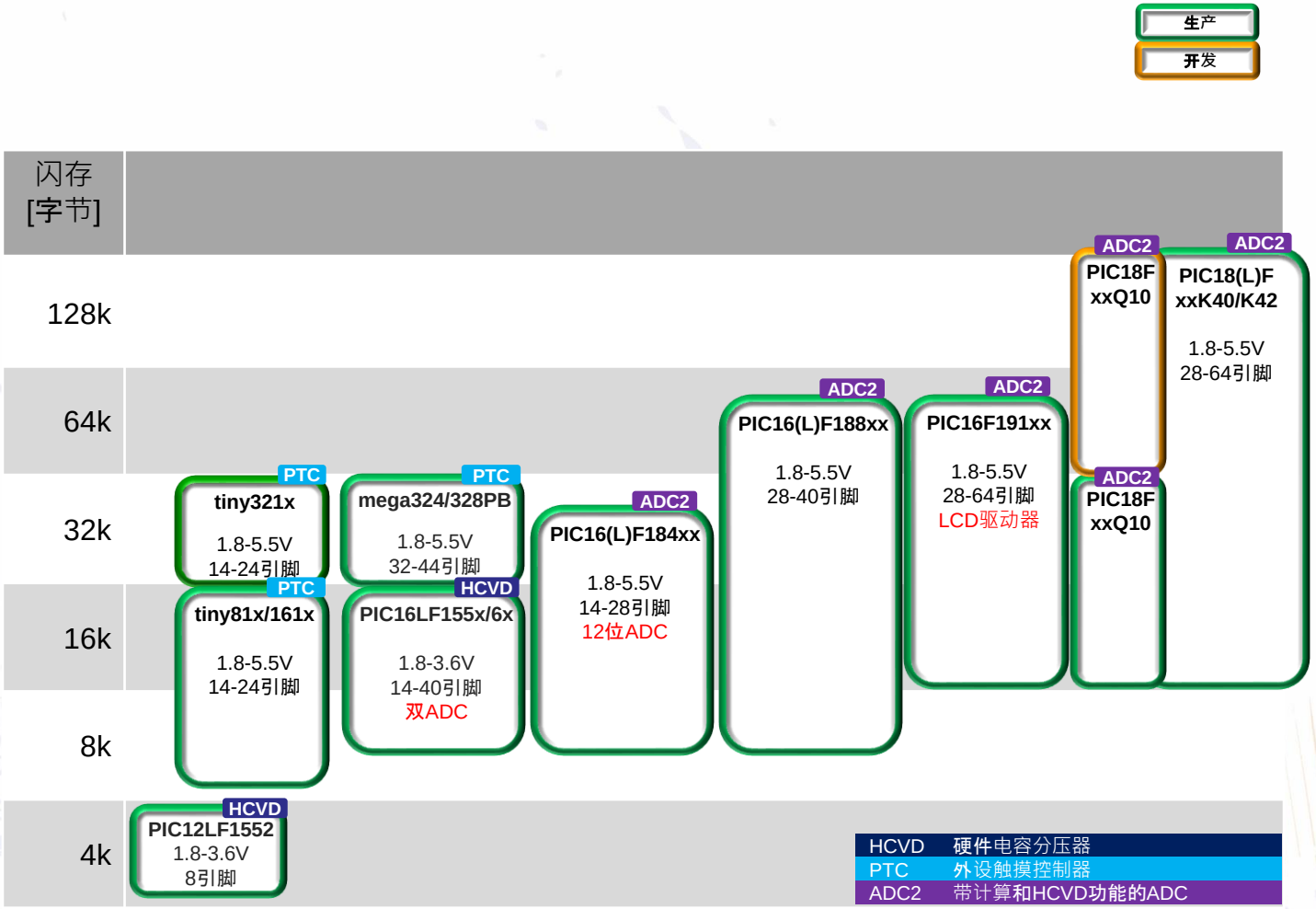
经验丰富的工程师兼研发负责人

触摸产品组合



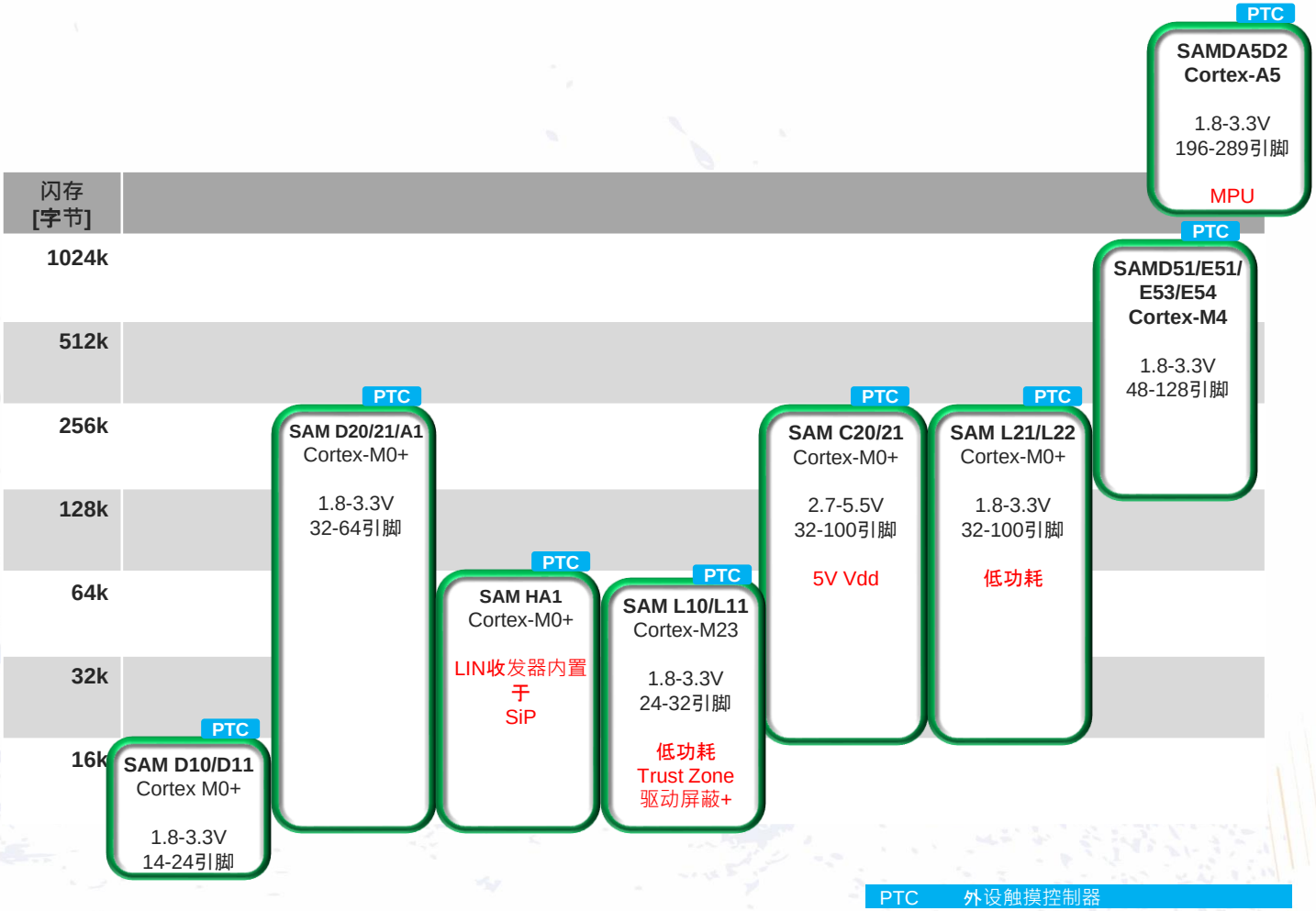
带触摸功能的8位MCU

硬件触摸控制器



带触摸功能的32位MCU/MPU

硬件触摸控制器



8位和32位触摸评估工具包

- 8位PIC® MCU
- 8位ATTINY/MEGA AVR
- 32位ATSAM ARM
- USB供电和接口
- 触摸传感器扩展板
- 嵌入式调试器
- 通过MCC和START模块化库中的应用示例支持

MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- MCC和mTouch®库——实验1
- START和QTouch®模块化库——实验2
- Data Visualizer入门——实验3
- 使用Data Visualizer和触摸库参数来调节触摸系统——实验4
- 实现触摸库的耐水和抗噪声功能——附加实验5
- mTouch®或QTouch®库——使用哪一个？
- 总结

MCC和mTouch®库

MCC简介

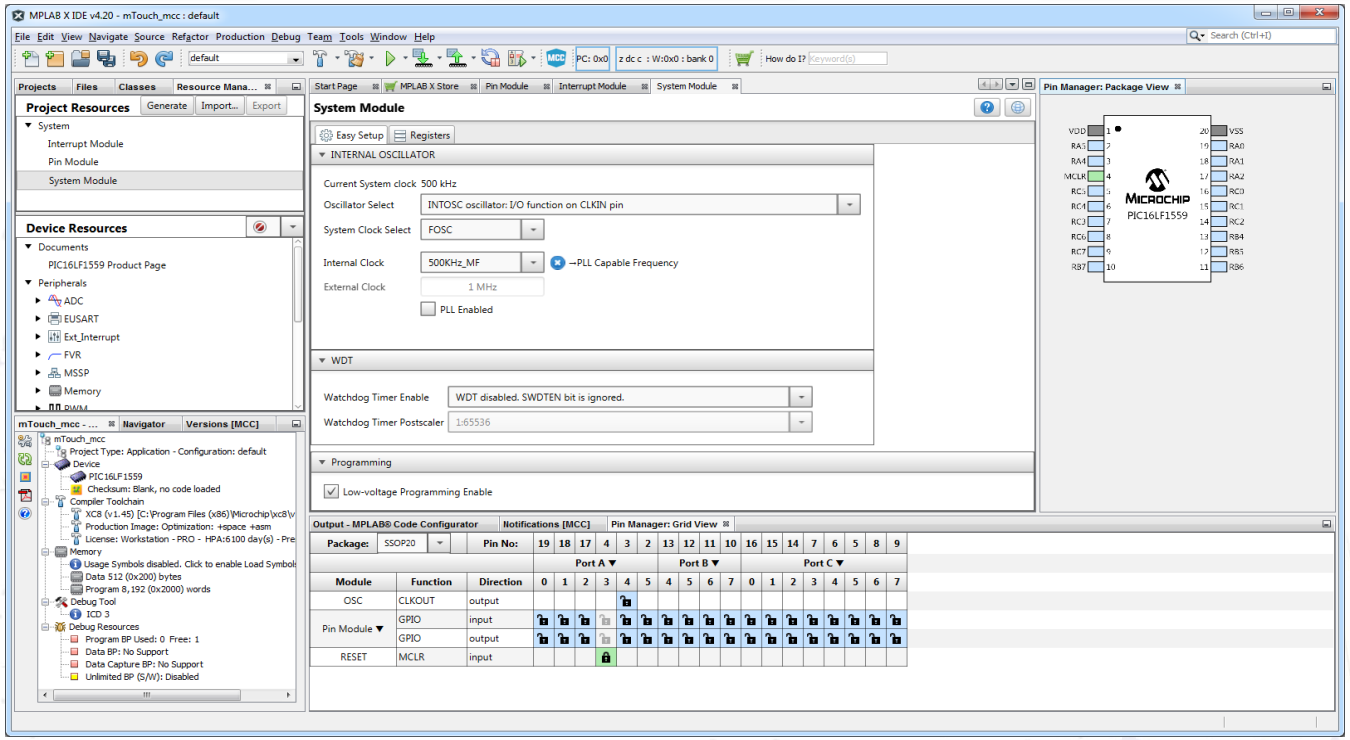
MICROCHIP MASTERs 2018

MCC mTouch®解决方案 MPLAB®代码配置器

资源区

设计区

引脚管理器区



MCC mTouch®解决方案
mTouch®电容式传感库

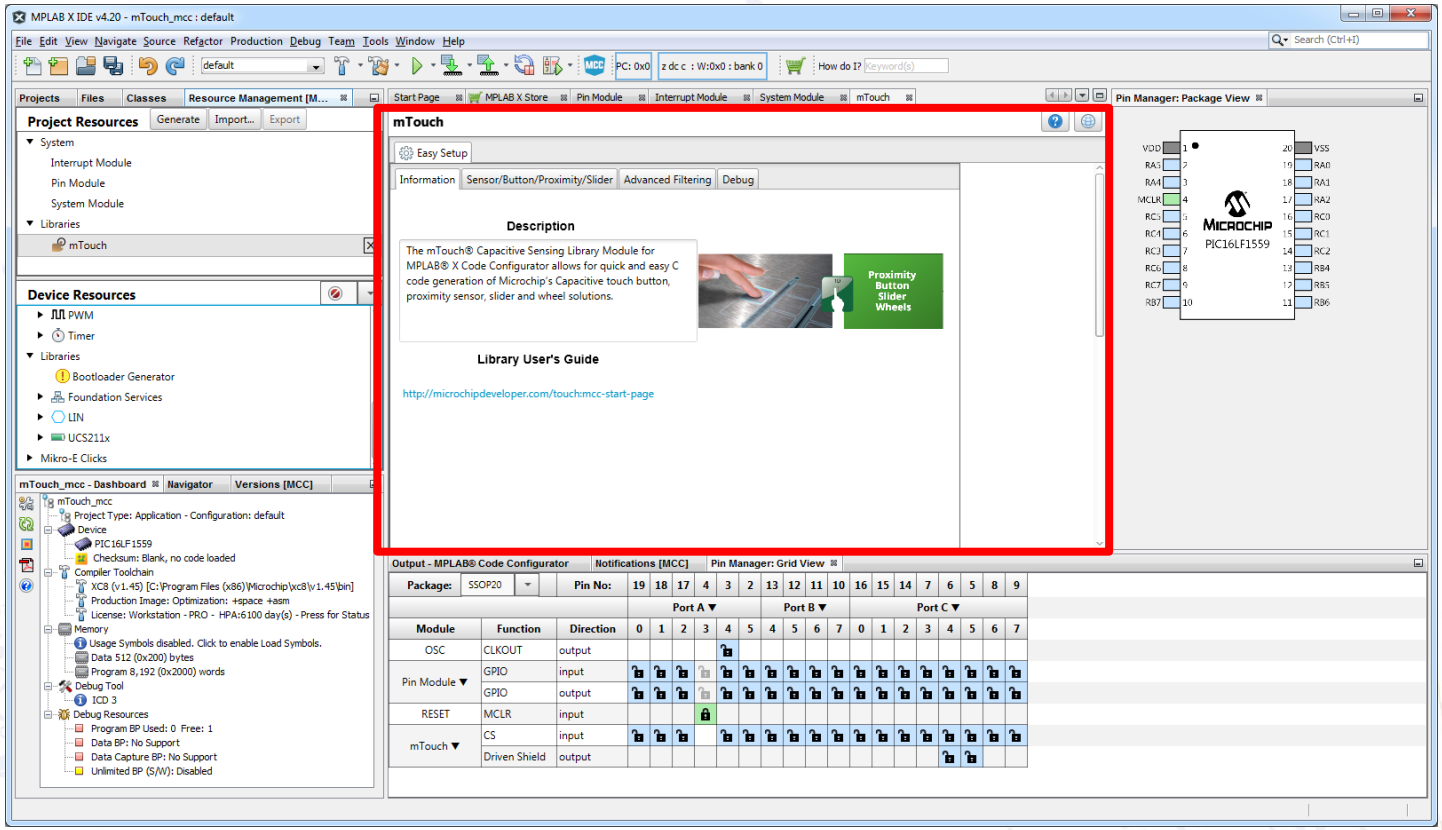
- MCC mTouch®电容式传感库支持10位SAR ADC和12位SAR ADC² 8位MCU。
- 通过MPLAB® X IDE代码配置器的图形用户界面（GUI）生成。
- 适用于所有MCU的Web工具
  - MPLAB® Xpress
- 触摸API有助于缩短开发时间
- 凭借触摸库和IDE，您可以将数十年的触摸专业知识应用到设计中

注：有关详细信息，请参见附录

MPLAB® X IDE代码配置器的mTouch®电容式传感库模块用户指南

MICROCHIP
MASTERs 2018

MCC mTouch®解决方案库

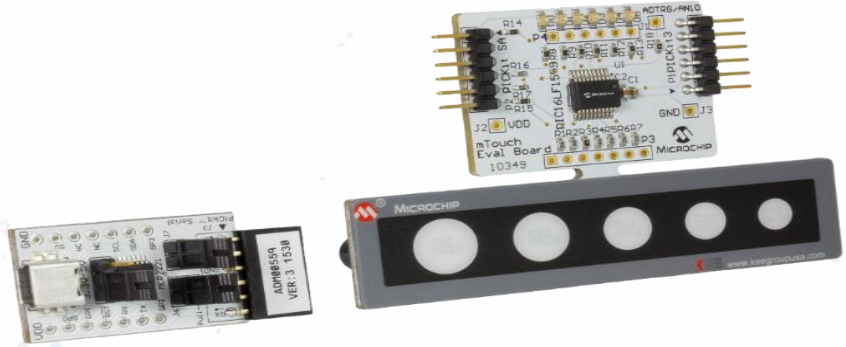


实验1：使用MPLAB®代码配置器和mTouch®库为PIC16LF1559创建mTouch®项目

实验1目标

- 使用MCC创建新项目
- 选择传感器和配置引脚
- 生成项目
- 添加代码以通过LED指示触摸

mTouch®评估工具包

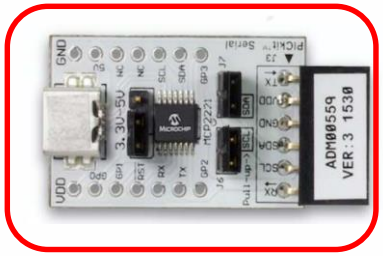


低成本mTouch®评估工具包
（部件编号DM160227）

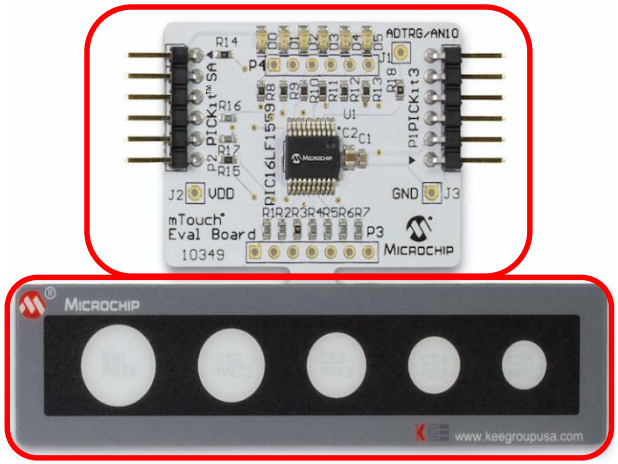
PICkit™ 3编程器
（部件编号PG164130）

mTouch®评估工具包

基于PIC16LF1559的
mTouch®板



MCP2221 USB-I2C和UART
转接模块



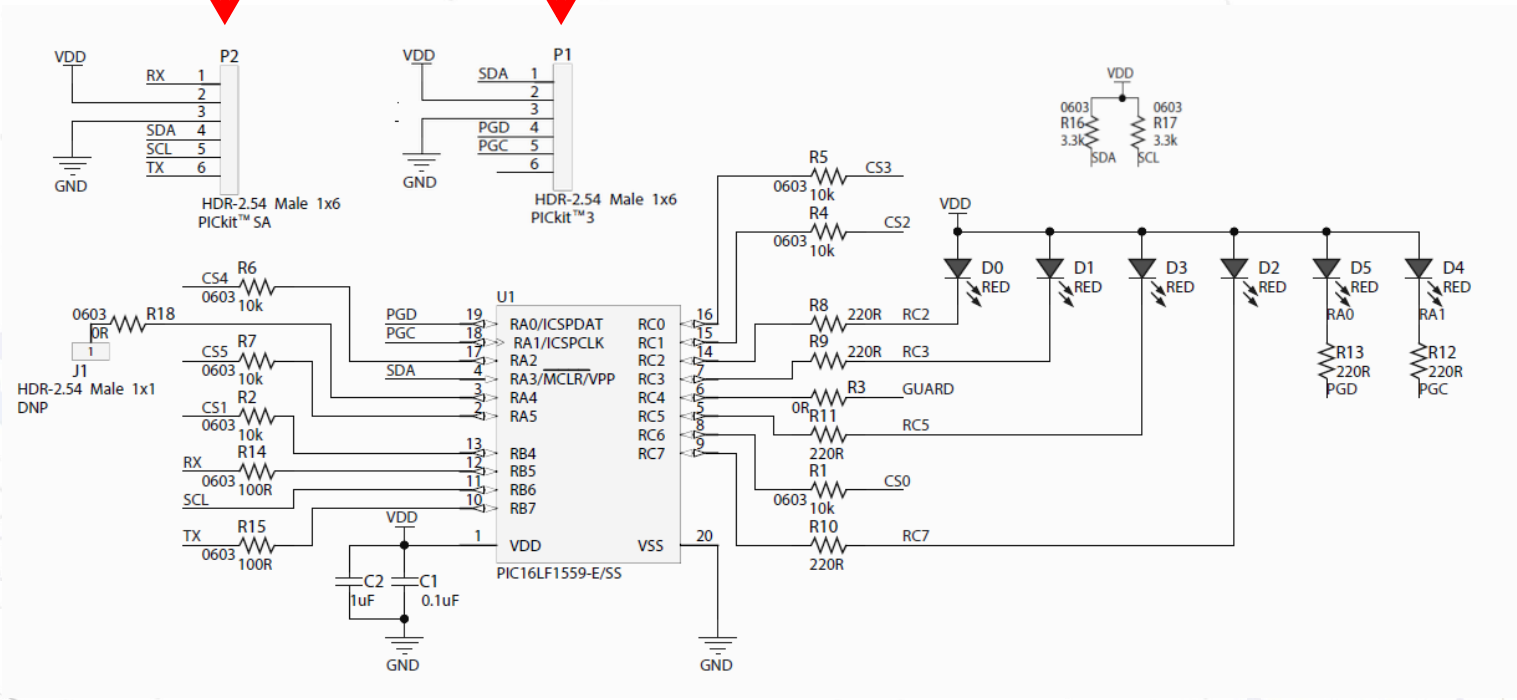
传感器

低成本mTouch®评估工具包用户指南

mTouch®评估工具包接口

MCP2221转接模块

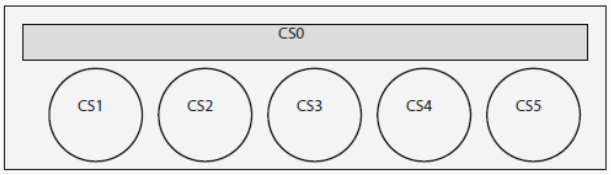
PICkit™ 3编程器

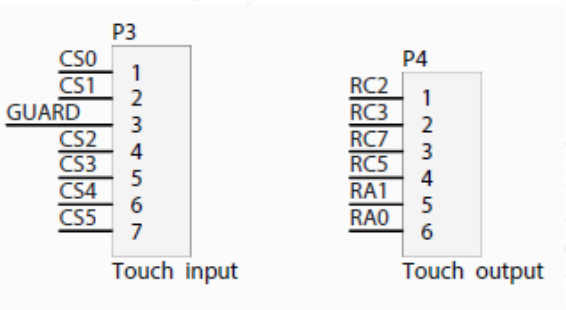


mTouch®评估工具包连接

触摸输入	引脚名称	功能
CS0	RC6	AN14
CS1	RB4	AN26
CS2	RC1	AN23
CS3	RC0	AN13
CS4	RA2	AN2
CS5	RA5	AN20
保护	RC4	AN11

输出	引脚名称	功能
LED D0	RC2	
LED D1	RC3	
LED D2	RC7	
LED D3	RC5	
LED D4	RA1	PGC
LED D5	RA0	PGD





实验1——在MPLAB® X IDE中使用mTouch®库

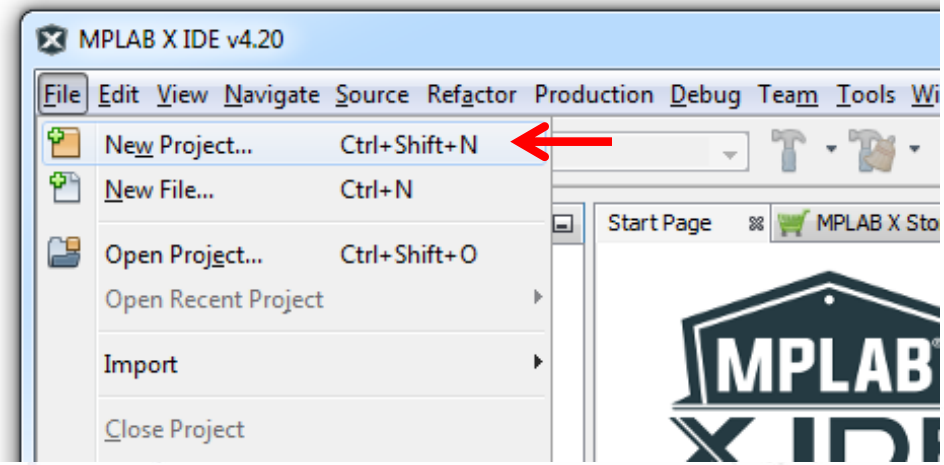
1 打开MPLAB® X IDE

- 关闭之前打开的所有项目

实验1——在MPLAB® X IDE中使用mTouch®库

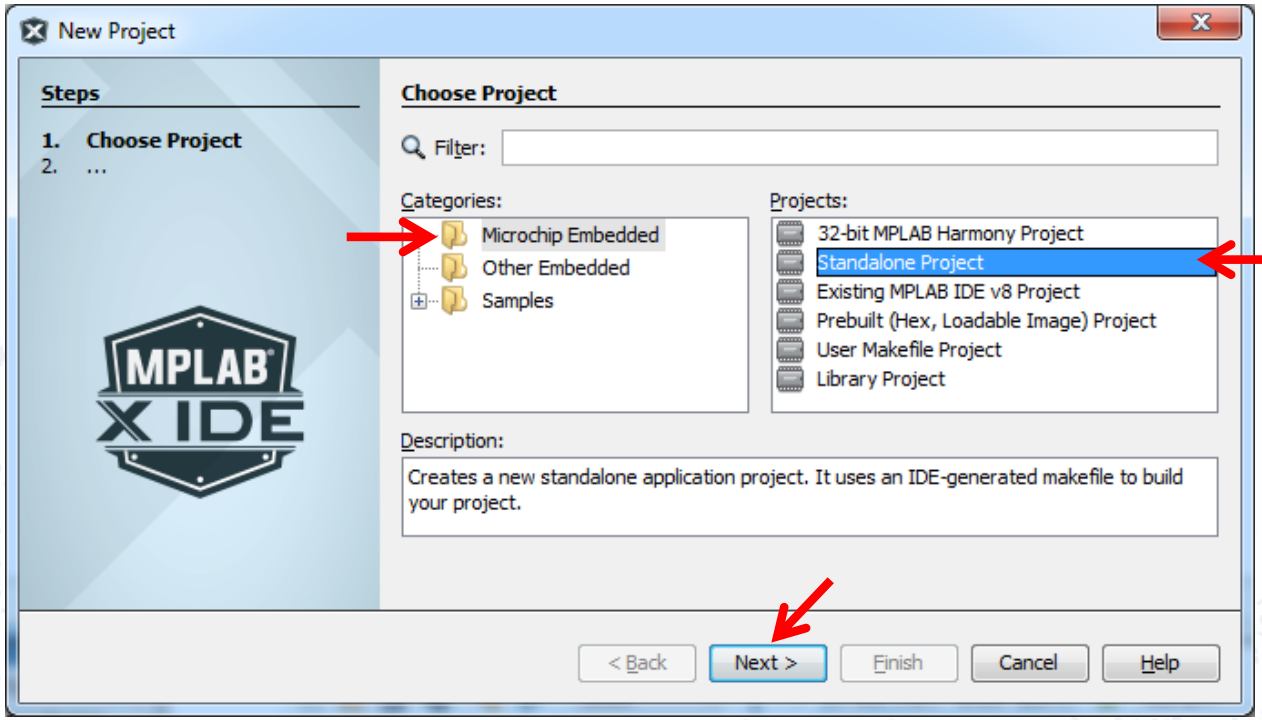
2 从头开始创建一个新项目

- 单击：File -> New Project …（文件 -> 新项目...）



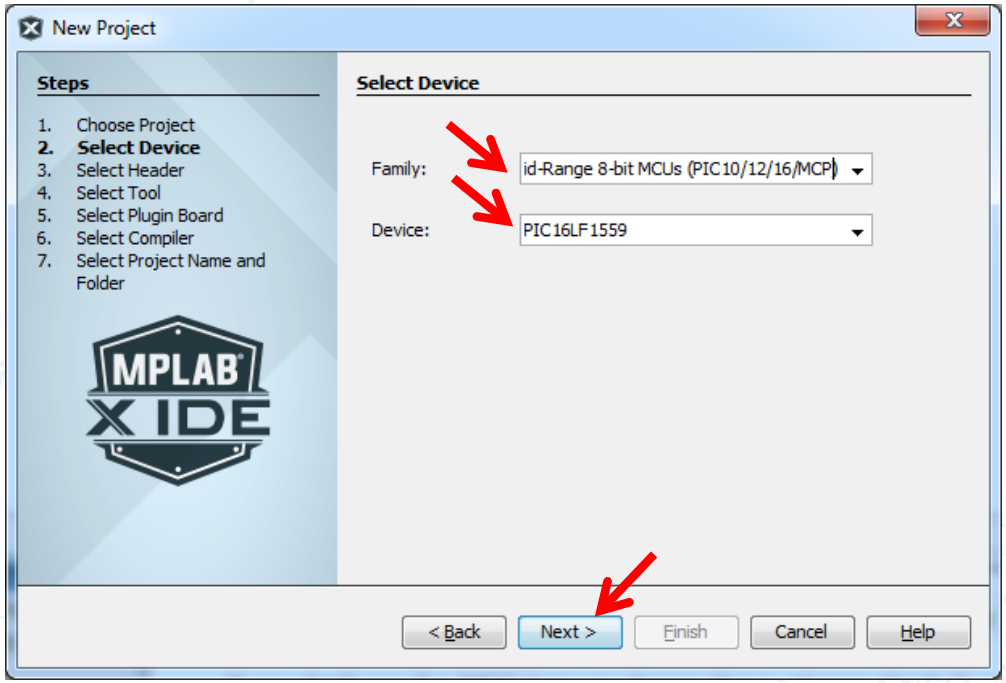
实验1——在MPLAB® X IDE中使用mTouch®库

- 选择：Microchip Embedded -> Standalone Project -> Next >（Microchip已安装工具 -> 独立项目 -> 下一步 >）



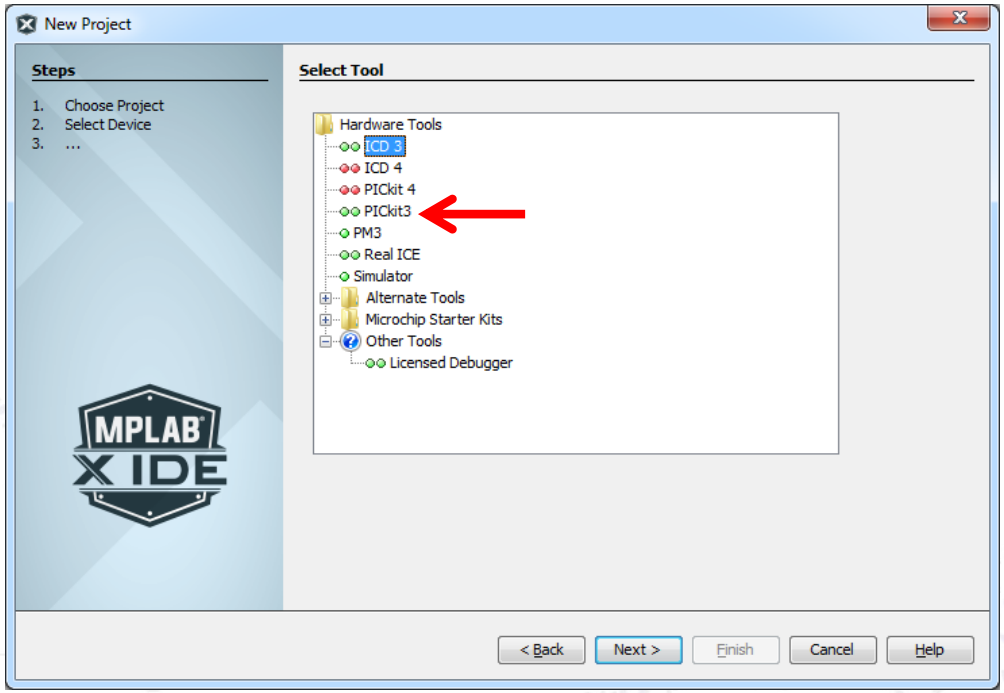
实验1——在MPLAB® X IDE中使用mTouch®库

- Select Device（选择器件）： Mid-Range 8位MCU/
- PIC16LF1559，然后单击Next >（下一步 >）



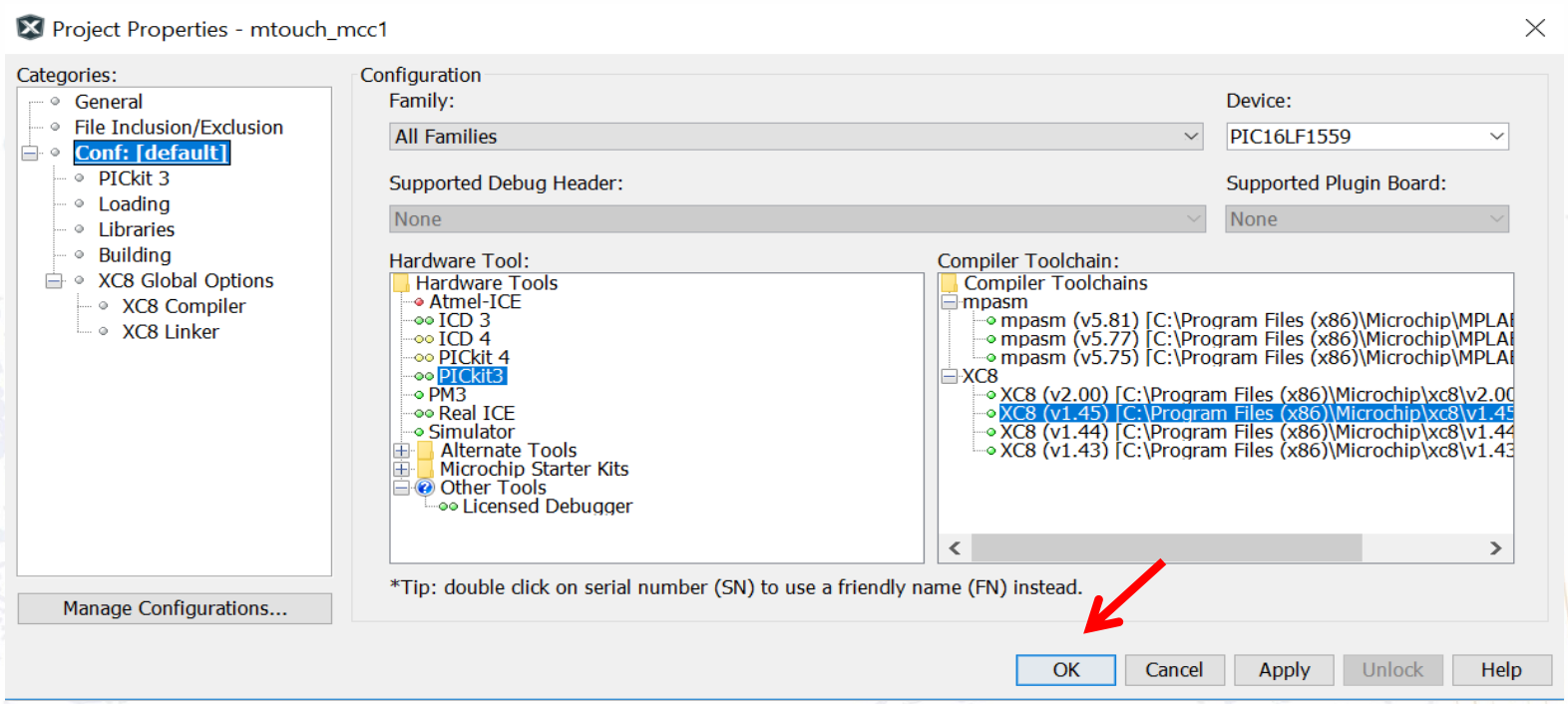
实验1——在MPLAB® X IDE中使用mTouch®库

- 选择硬件工具：模拟器或**PICkit™ 3**，然后单击Next >



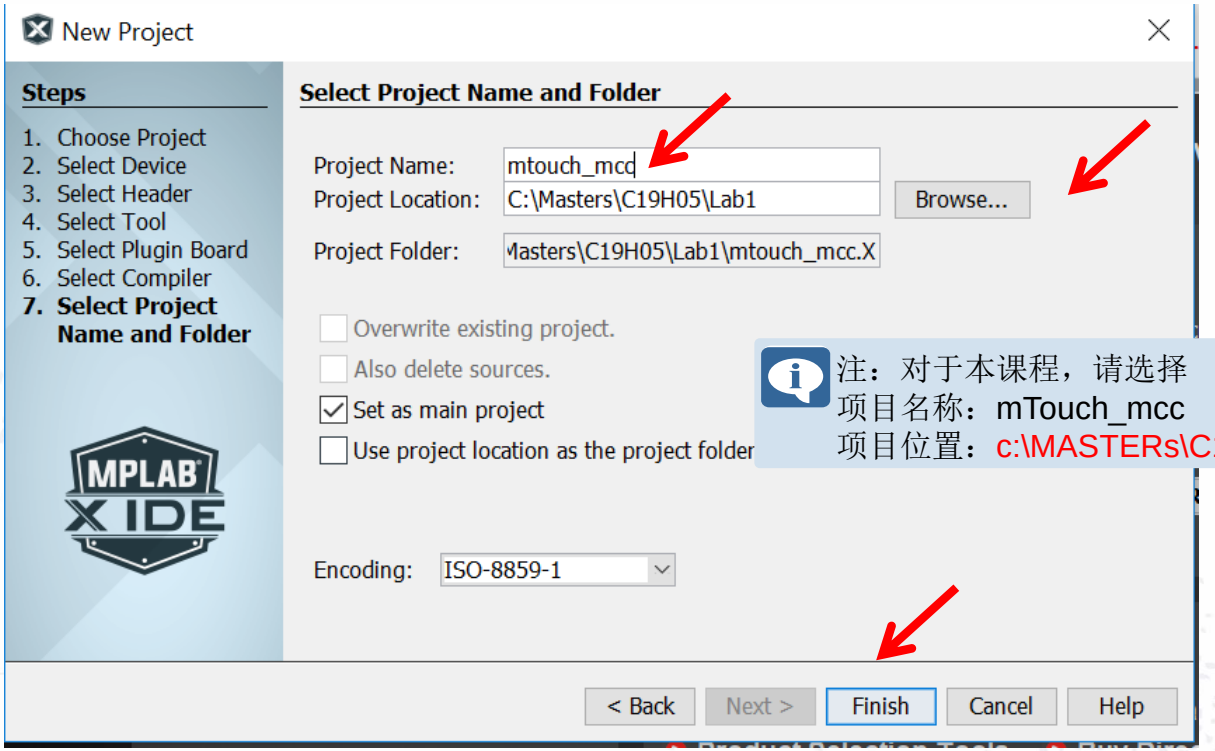
实验1——在MPLAB® X IDE中使用mTouch®库

- Select Compiler（选择编译器）：XC8 (v1.45)，然后单击Next >



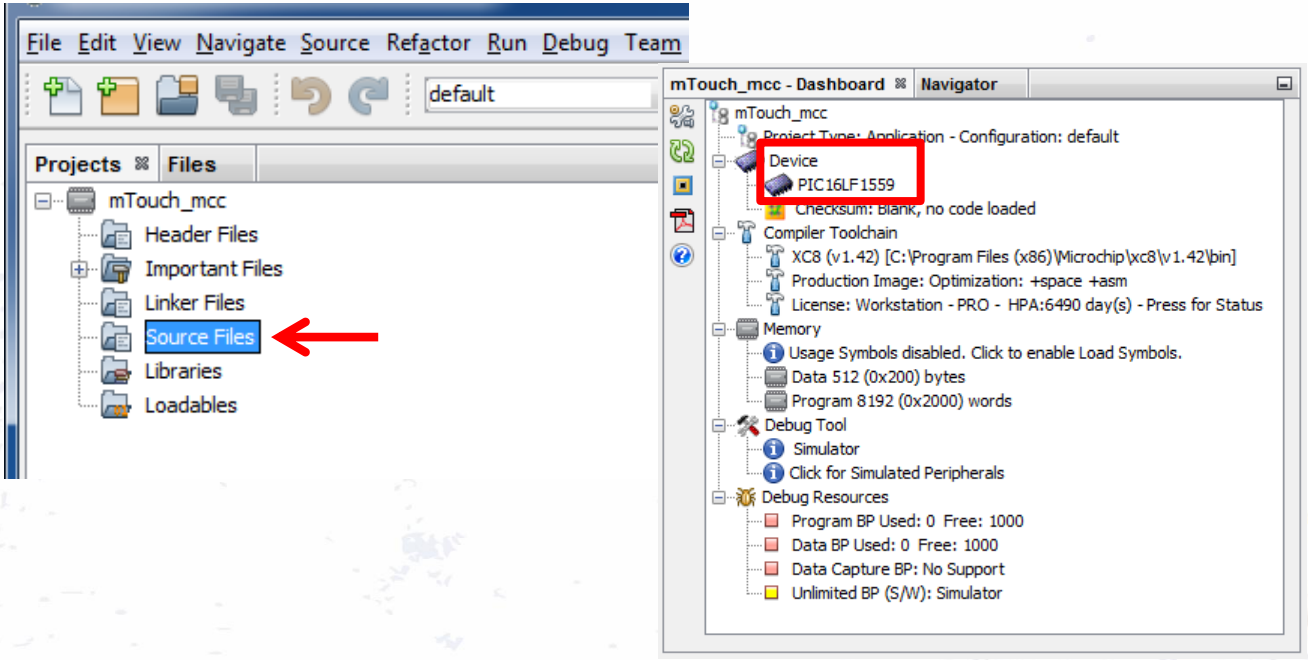
实验1——在MPLAB® X IDE中使用mTouch®库

- 选择项目名称和项目位置，然后单击：完成



实验1——在MPLAB® X IDE中使用mTouch®库

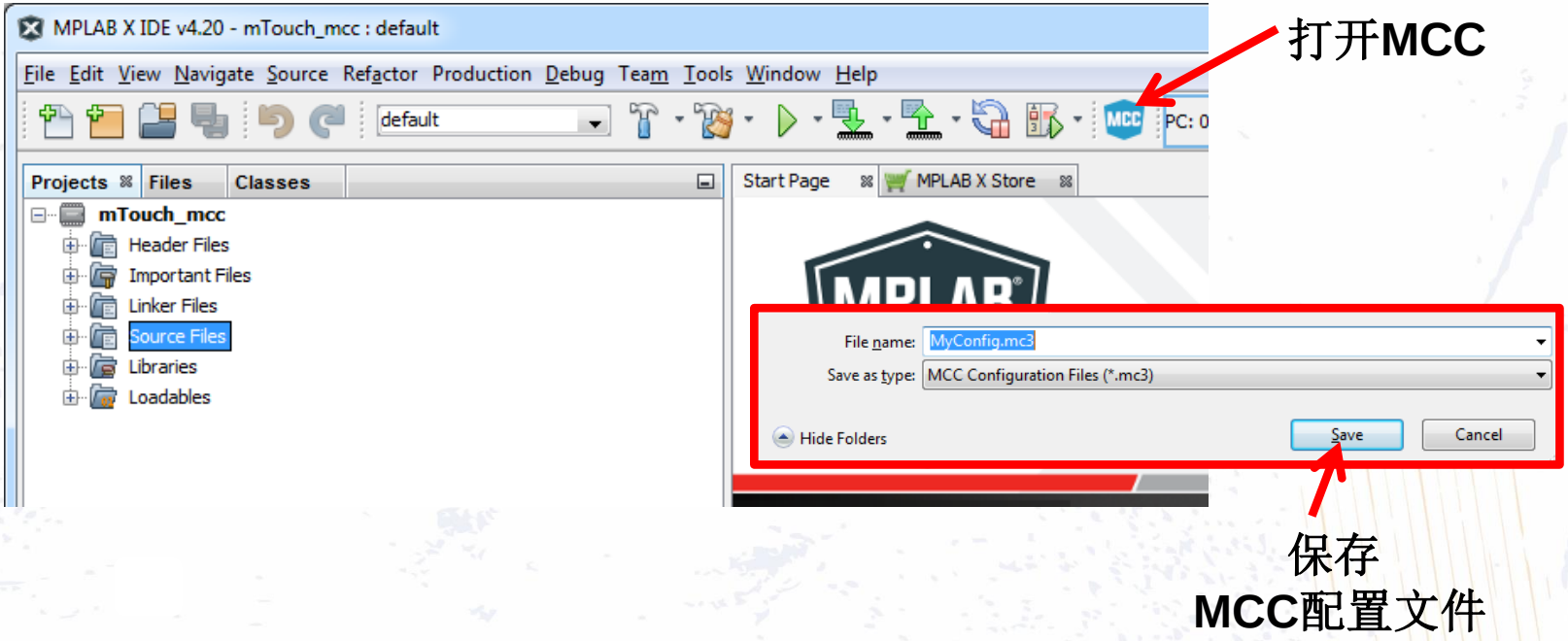
- 验证MPLAB® X IDE是否已创建新项目
- 验证Source Files（源文件）文件夹是否为空



实验1——在MPLAB® X IDE中使用mTouch®库

3 打开MPLAB®代码配置器工具

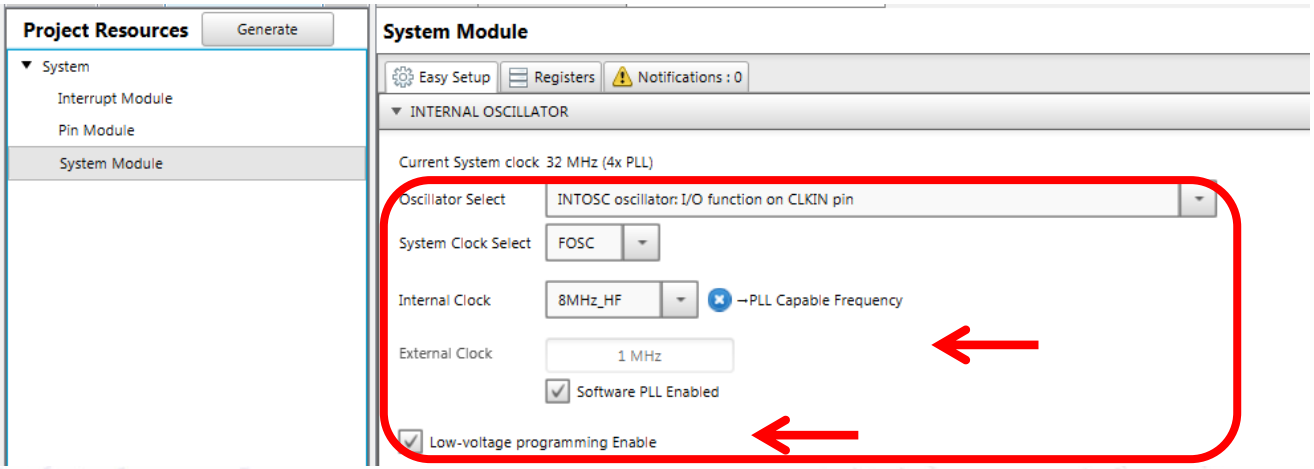
- 按要求保存MCC配置文件



实验1——在MPLAB® X IDE中使用mTouch®库

4 配置系统时钟

- 为内部时钟选择8 MHz_HF，并使能PLL。这将产生32 MHz的系统时钟。

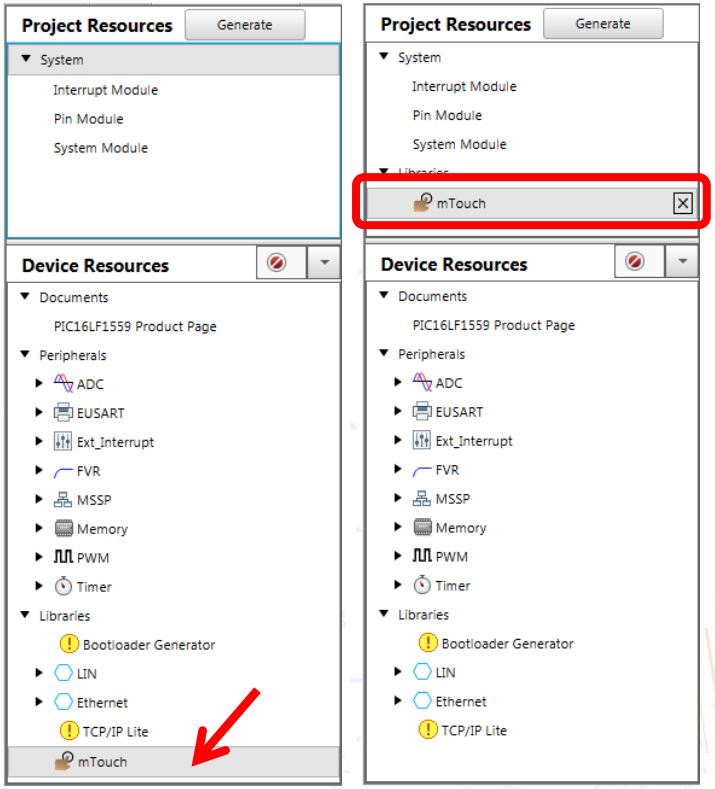


- 为确保mTouch®按钮/接近检测的性能，要求系统时钟至少为8 MHz。如果选择低于8 MHz的系统时钟，则mTouch®模块将在通知窗口中生成一条警告。

实验1——在MPLAB® X IDE中使用mTouch®库

5 加载mTouch模块

- 双击Device Resources（器件资源）窗口中Libraries（库）列表中的mTouch图标。
- 加载模块后，mTouch图标将出现在Project Resources（项目资源）中。



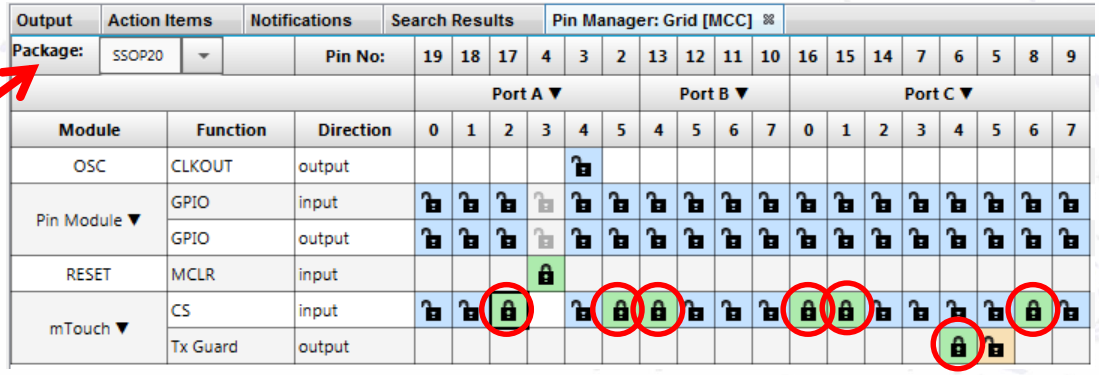
实验1——在MPLAB® X IDE中使用mTouch®库

6 选择mTouch传感器

- 加载mTouch模块后，所有可用的mTouch传感器引脚将显示在Pin Manager（引脚管理器）中。您需要根据下面的传感器引脚排列信息选择物理传感器和保护。

触摸	引脚名称	功能	方向
CS0	RC6	接近传感器	输入
CS1	RB4	Button0	输入
CS2	RC1	Button1	输入
CS3	RC0	Button2	输入
CS4	RA2	Button3	输入
CS5	RA5	Button4	输入
保护	RC4	保护	输出

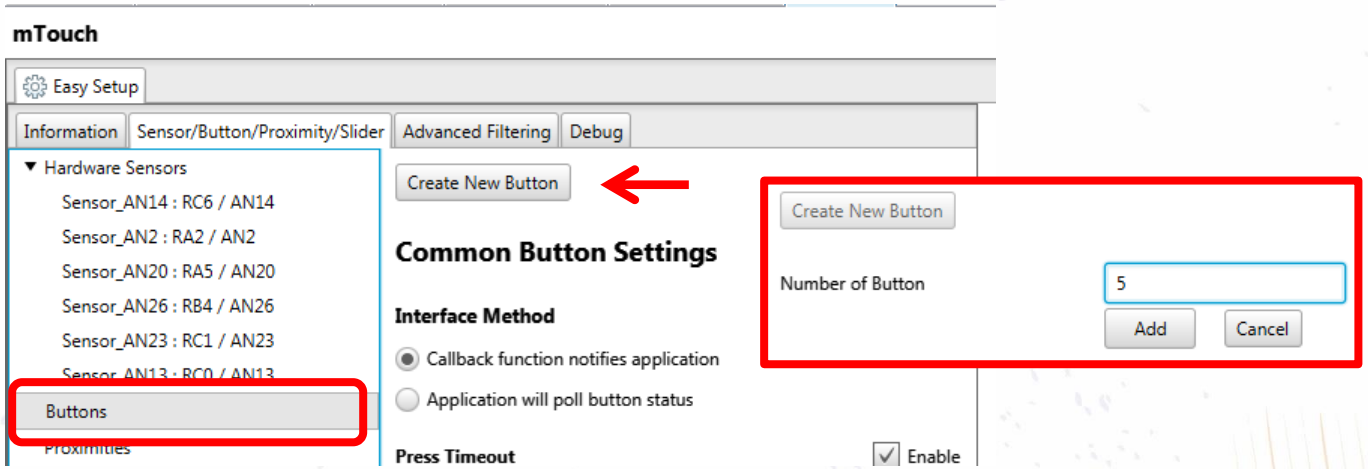
注：
选择封装：
SSOP20



实验1——在MPLAB® X IDE中使用mTouch®库

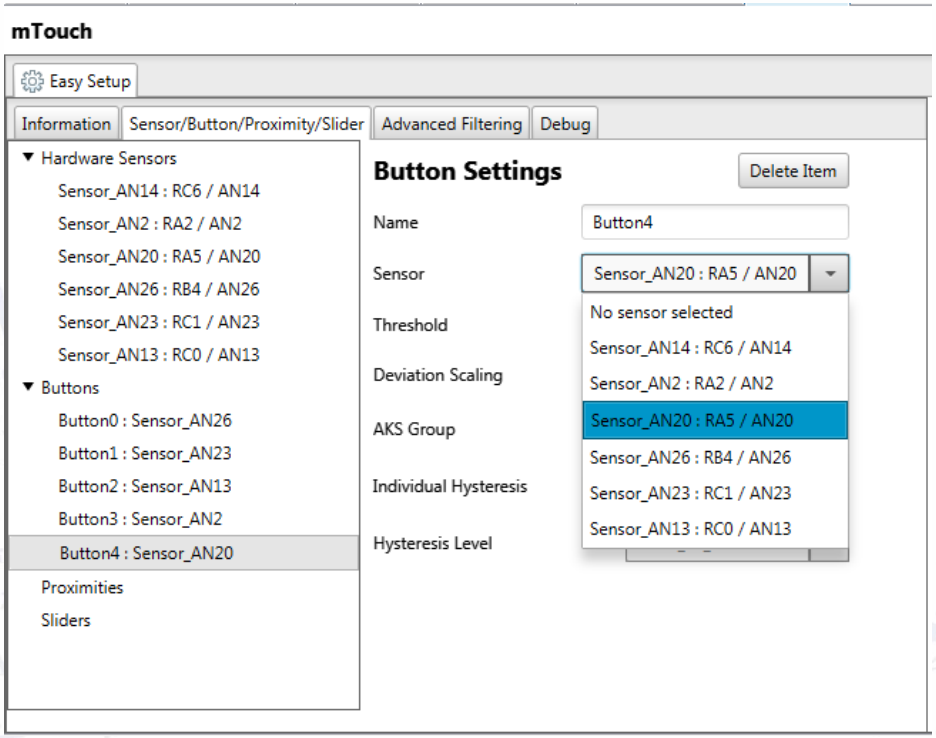
7 **添加mTouch按钮并链接到传感器。**

- 转到Sensor/Button/Proximity/Slider（传感器/按钮/接近检测/滑动条）选项卡，然后单击**Create New Button**（创建新按钮），添加5个按钮。



实验1——在MPLAB® X IDE中使用mTouch®库

- 然后，单击每个按钮名称，转到Button Settings（按钮设置）视图，并选择上一页表格中显示的相应硬件传感器

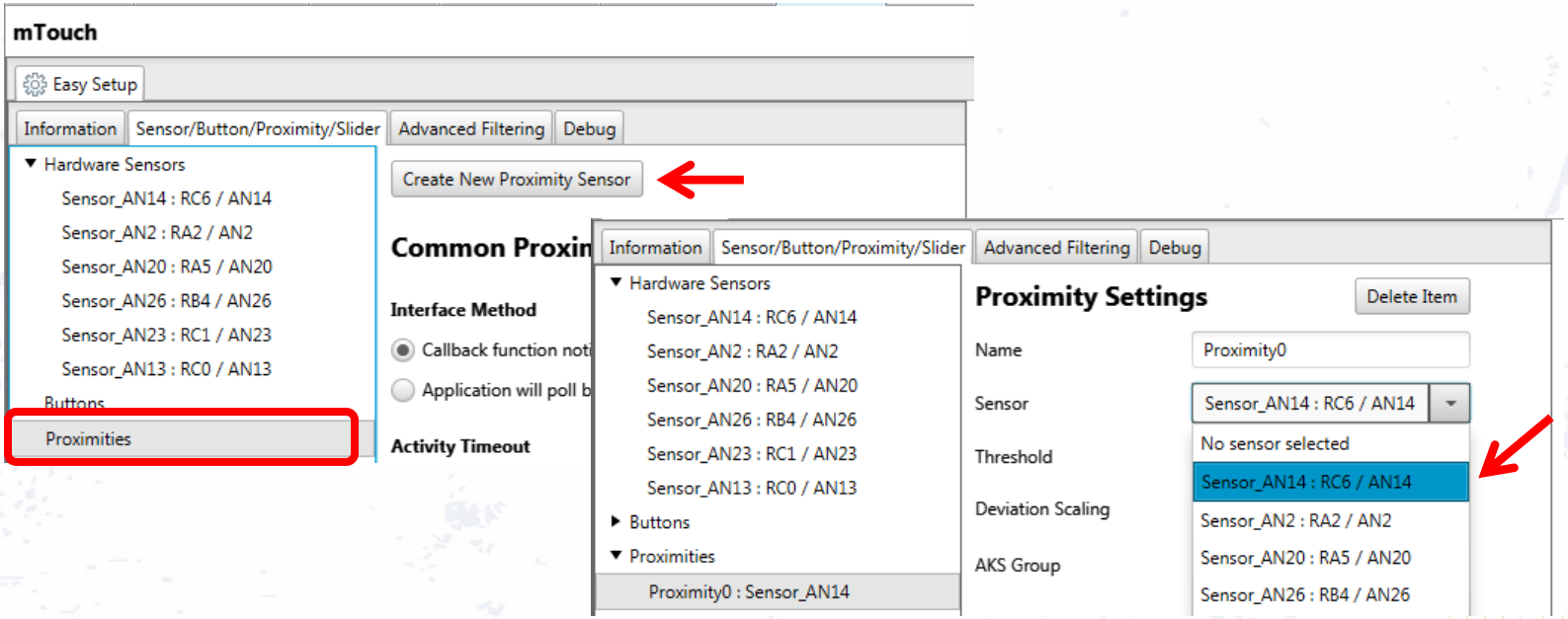


触摸	引脚名称	功能	方向
CS0	RC6	接近传感器	输入
CS1	RB4	Button0	输入
CS2	RC1	Button1	输入
CS3	RC0	Button2	输入
CS4	RA2	Button3	输入
CS5	RA5	Button4	输入
保护	RC4	保护	输出

实验1——在MPLAB® X IDE中使用mTouch®库

8 **添加mTouch接近检测并链接到传感器。**

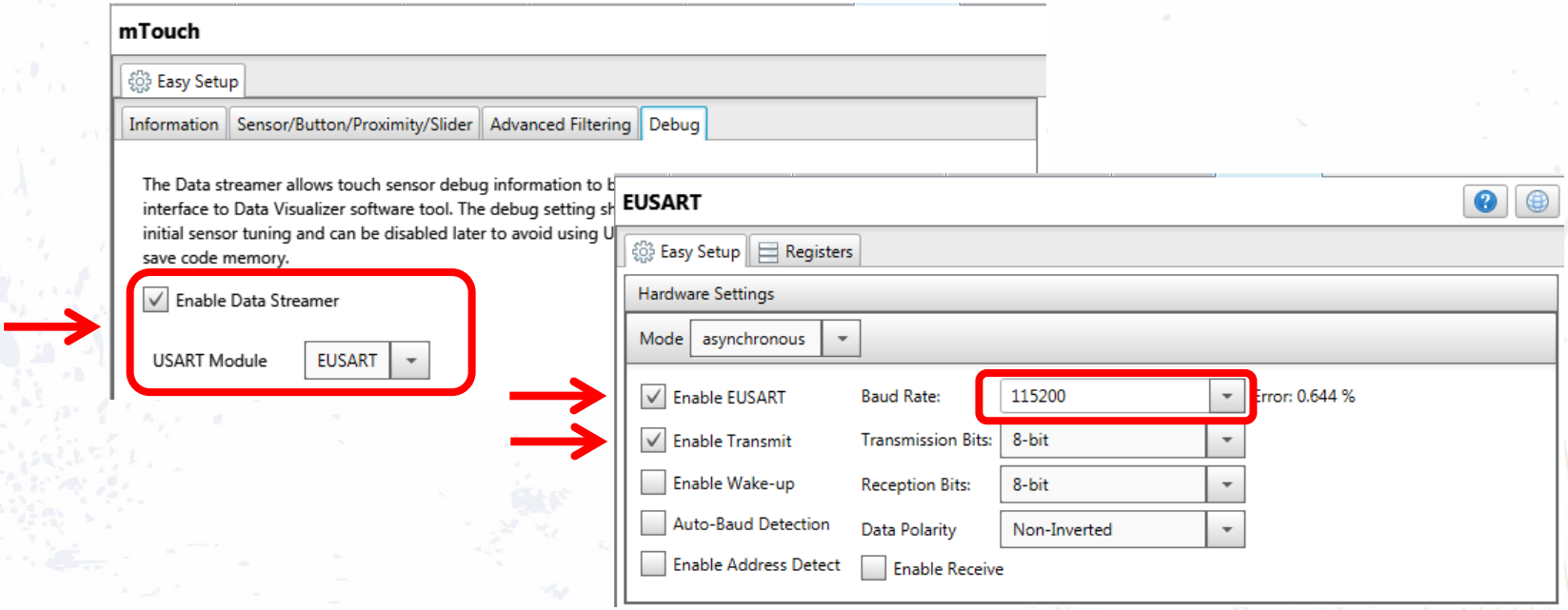
- 转到Proximities（接近检测）配置视图，单击**Create New Proximity Sensor**（创建新的接近传感器）并选择Sensor_AN14以链接到Proximity0。



实验1——在MPLAB® X IDE中使用mTouch®库

9 **在EUSART1上使能Data Streamer。**

- 转到Debug（调试）选项卡，单击**Enable Data Streamer**（使能Data Streamer）。将加载EUSART外设。

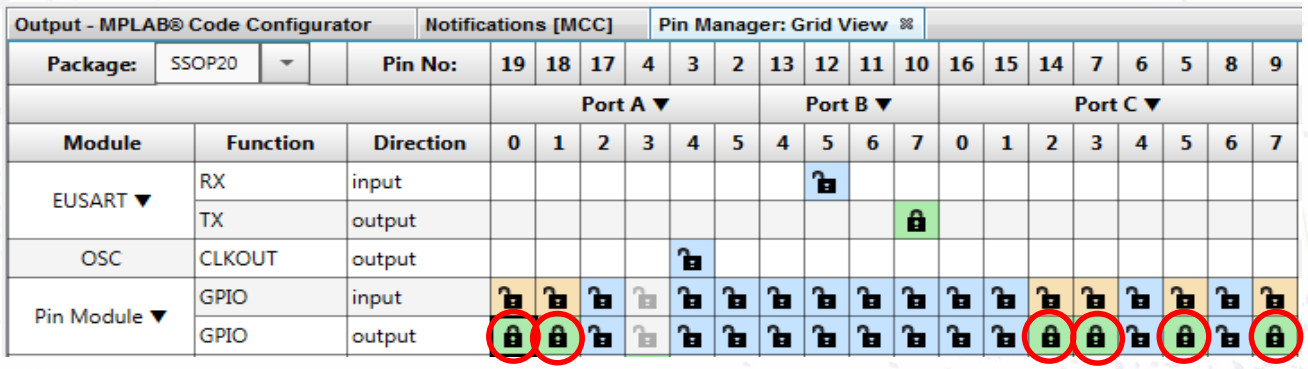


实验1——在MPLAB® X IDE中使用mTouch®库

10 **设置I/O引脚来控制LED。**

- 此电路板上有6个LED，我们希望用它们来指示接近传感器和按钮的状态。引脚排列如下所示：
- 使用引脚管理器网格视图根据引脚模块输出表选择LED引脚。

引脚名称	功能
RC2	LED0
RC3	LED1
RC7	LED2
RC5	LED3
RA1	LED4
RA0	LED5



实验1——在MPLAB® X IDE中使用mTouch®库

- 转到Project Resources -> Pin Module（项目资源 -> 引脚模块），将引脚重命名为LEDx。
- 由于LED为低电平有效，我们需要将引脚设置为高电平来使LED熄灭。
- 为mTouch传感器取消选中所有LED的Analog（模拟）复选框以及WPU。

Pin Module

Easy Setup | Registers | Notifications : 2

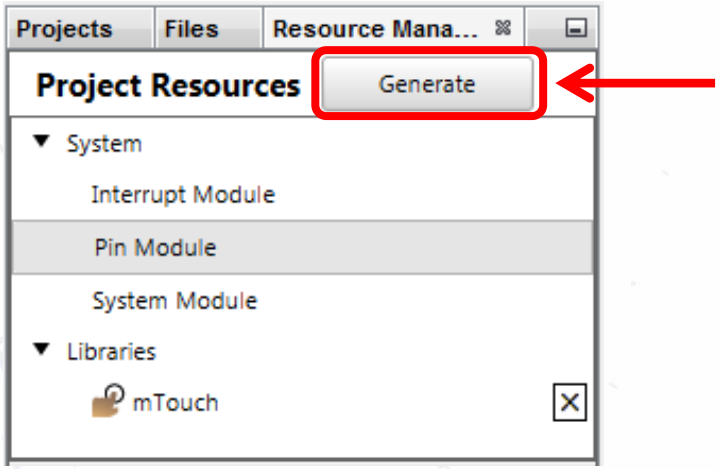
Selected Package : SSOP20

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU
RA0	Pin Module	GPIO	LED5	☑	☐	☑	☐
RA1	Pin Module	GPIO	LED4	☑	☐	☑	☐
RA2	mTouch	AN2		☐	☑	☐	☐
RA5	mTouch	AN20		☐	☑	☐	☐
RB4	mTouch	AN26		☐	☑	☐	☐
RC0	mTouch	AN13		☐	☑	☐	
RC1	mTouch	AN23		☐	☑	☐	
RC2	Pin Module	GPIO	LED0	☑	☐	☑	
RC3	Pin Module	GPIO	LED1	☑	☐	☑	
RC4	mTouch	Tx Guard	LED3	☐	☑	☑	
RC5	Pin Module	GPIO	LED2	☑	☐	☑	
RC6	mTouch	AN14	LED2	☐	☑	☐	
RC7	Pin Module	GPIO	LED3	☑	☐	☑	

实验1——在MPLAB® X IDE中使用mTouch®库

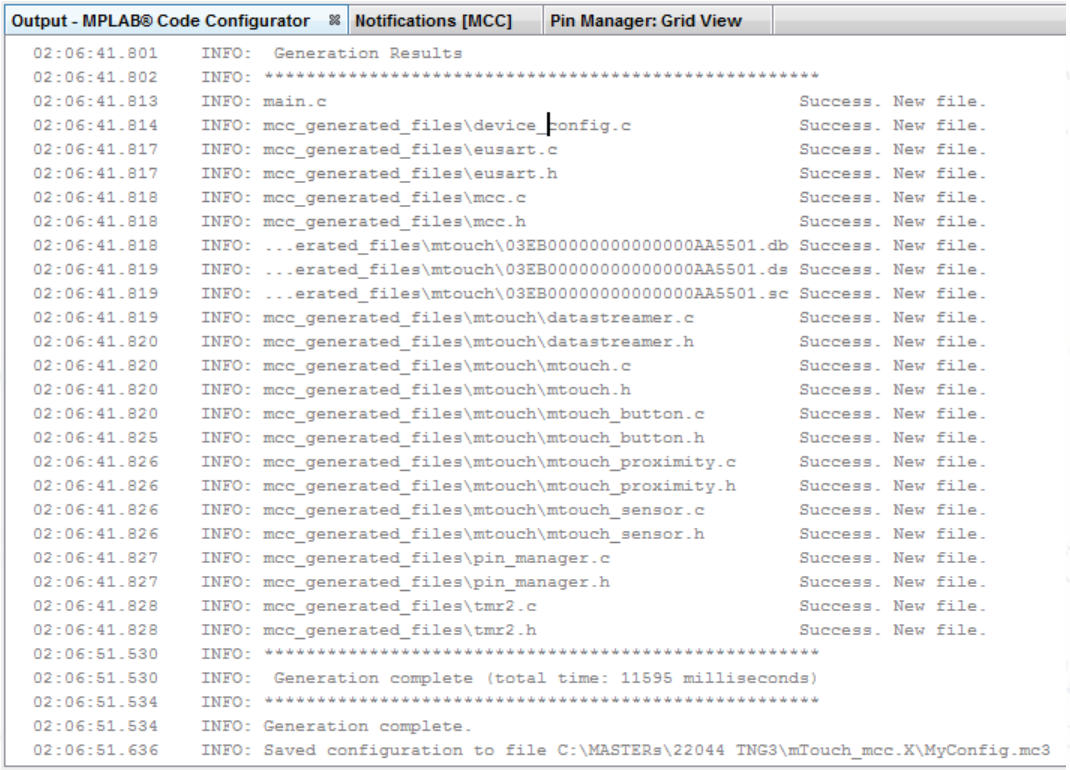
11 生成代码

- 单击Generate（生成）按钮。



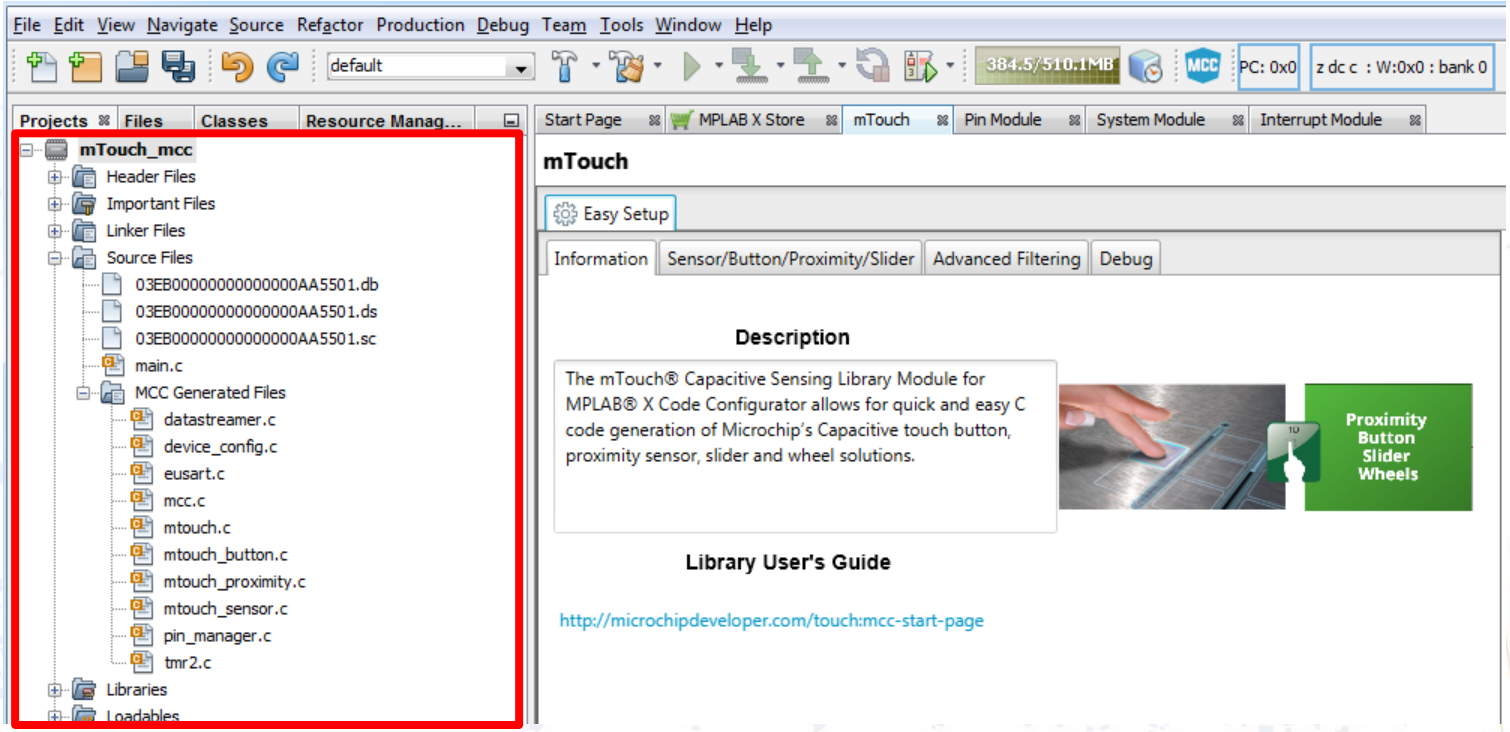
实验1——在MPLAB® X IDE中使用mTouch®库

- 验证MCC工具自动生成的文件列表



实验1——在MPLAB® X IDE中使用mTouch®库

- 查看MCC生成的项目代码



实验1——在MPLAB® X IDE中使用mTouch®库

12 在main.c中调用mTouch service

- 打开生成的main.c文件，将**MTOUCH_Service_Mainloop()**置于while循环，如下所示（突出显示）：

```
65
66      // Enable the Peripheral Interrupts
67      //INTERRUPT_PeripheralInterruptEnable();
68
69      // Disable the Global Interrupts
70      //INTERRUPT_GlobalInterruptDisable();
71
72      // Disable the Peripheral Interrupts
73      //INTERRUPT_PeripheralInterruptDisable();
74
75
76      while (1)
77      {
78          // Add your application code
79          MTOUCH_Service_Mainloop();
80      }
81  }
```

调用mTouch函数

*提示：*在输入首字母后输入<CTRL>空格键可查看以该字母标识开头的函数名称、变量和宏的列表

实验1——在MPLAB® X IDE中使用mTouch®库

回调函数

- 在本例中，我们通过将函数指针传送给setter函数的方式在发生按下/释放事件时使用回调函数。
- 为将API用于mTouch按钮和接近检测，已将mtouch_button.h和mtouch_proximity.h包含在mtouch.h中。

注：MCC已生成回调函数。

mtouch_button.h

```
/*
 * ===========================================================================
 * Global Functions
 * ===========================================================================
 */
    void MTOUCH_Button_SetPressedCallback   (void (*callback)(enum mtouch_button_names button));
    void MTOUCH_Button_SetNotPressedCallback(void (*callback)(enum mtouch_button_names button));
```

mtouch_proximity.h

```
/*
 * ===========================================================================
 * Global Functions
 * ===========================================================================
 */
    void MTOUCH_Proximity_SetActivatedCallback   (void (*callback)(enum mtouch_proximity_names prox));
    void MTOUCH_Proximity_SetNotActivatedCallback(void (*callback)(enum mtouch_proximity_names prox));
```

实验1——在MPLAB® X IDE中使用mTouch®库

13 设置自己的回调函数

- 然后我们为按钮和接近传感器编写按下/释放回调函数，并且

设置触摸事件的回调

```
62  /*
63                          Main application
64   */
65  void main(void)
66  {
67      // initialize the device
68      SYSTEM_Initialize();
69
70      //...14 lines
84
85      MTOUCH_Button_SetPressedCallback(processButtonTouch);
86      MTOUCH_Button_SetNotPressedCallback(processButtonRelease);
87      MTOUCH_Proximity_SetActivatedCallback(processProximityActivate);
88      MTOUCH_Proximity_SetNotActivatedCallback(processProximityNotActivate);
89
90      while (1)
91      {
92          // Add your application code
93          MTOUCH_Service_Mainloop();
94      }
95  }
```

- 最后将这些函数指针传送给每个事件的setter函数，如下所示（突出显示）：

实验1——在MPLAB® X IDE中使用mTouch®库

实现按钮和接近事件的函数

注：我们将在下一步填入这些函数。
所有文字都可以在提供的文件“MCC_text2type.txt”中找到

```
46  #include "mcc_generated_files/mcc.h"
47
48  void processButtonTouch(enum mtouch_button_names button)
49  {
50  }
51
52  void processButtonRelease(enum mtouch_button_names button)
53  {
54  }
55
56  void processProximityActivate(enum mtouch_proximity_names prox)
57  {
58  }
59
60  void processProximityNotActivate(enum mtouch_proximity_names prox)
61  {
62  }
63
64  /*
65                  Main application
66   */
67  void main(void)
68  {
69      // initialize the device
70      SYSTEM_Initialize();
71
72      //...14 lines
86
87      MTOUCH_Button_SetPressedCallback(processButtonTouch);
88      MTOUCH_Button_SetNotPressedCallback(processButtonRelease);
89      MTOUCH_Proximity_SetActivatedCallback(processProximityActivate);
90      MTOUCH_Proximity_SetNotActivatedCallback(processProximityNotActivate);
91
92      while (1)
93      {
94          // Add your application code
95          MTOUCH_Service_Mainloop();
96
97      }
98  }
```

实验1——在MPLAB® X IDE中使用mTouch®库

14 **使用mTouch按钮和接近检测来控制LED。**

- 设置回调函数后，您需要根据不同按钮或接近检测的事件实现LED控制逻辑。完整的main.c代码如下所示：

示例代码

```
switch(button)
  {
     case Button0: LED1_SetLow();break;          // LED1 active low = ON
     case Button1: LED2_SetLow();break;
     case Button2: LED3_SetLow();break;
     case Button3: LED4_SetLow();break;
     case Button4: LED5_SetLow();break;
     default: break;
  }


  if(prox == Proximity0)
       LED0_SetLow();                  // LED0 active low = ON
```

实验1——在MPLAB® X IDE中使用mTouch®库

- 使用mTouch按钮和接近检测来控制LED（续）

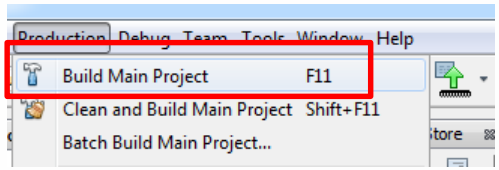
根据不同接近检测/按钮的事件实现控制LED的逻辑

```
#include "mcc_generated_files/mcc.h"

void processButtonTouch(enum mtouch_button_names button)
{
    switch(button)
    {
        case Button0: LED1_SetLow();break;
        case Button1: LED2_SetLow();break;
        case Button2: LED3_SetLow();break;
        case Button3: LED4_SetLow();break;
        case Button4: LED5_SetLow();break;
        default: break;
    }
}

void processButtonRelease(enum mtouch_button_names button)
{
        switch(button)
    {
        case Button0: LED1_SetHigh();break;
        case Button1: LED2_SetHigh();break;
        case Button2: LED3_SetHigh();break;
        case Button3: LED4_SetHigh();break;
        case Button4: LED5_SetHigh();break;
        default: break;
    }
}

void processProximityActivate(enum mtouch_proximity_names prox)
{
    if(prox == Proximity0)
        LED0_SetLow();
}

void processProximityNotActivate(enum mtouch_proximity_names prox)
{
    if(prox == Proximity0)
        LED0_SetHigh();
}
```

实验1——在MPLAB® X IDE中使用mTouch®库

15 编译项目

- 按 

或F11 

- 您已成功编译

Output | Notifications [MCC] | Pin Manager: Grid View

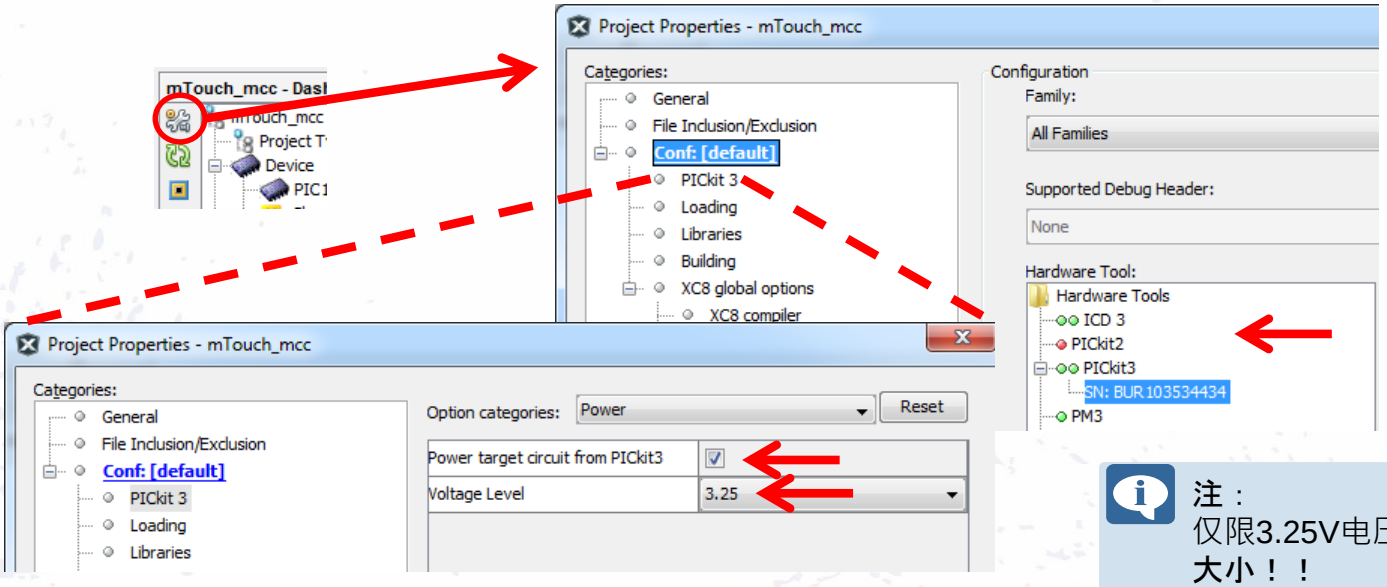
MPLAB® Code Configurator | mTouch_mcc (Clean, Build, ...)

```
Memory Summary:
    Program space        used   FECh (  4076) of  2000h words   ( 49.8%)
    Data space           used    FCh (   252) of   1F0h bytes   ( 50.8%)
    EEPROM space         None available
    Data stack space     used     0h (     0) of   100h bytes   (  0.0%)
    Configuration bits   used     2h (     2) of     2h words   (100.0%)
    ID Location space    used     0h (     0) of     4h bytes   (  0.0%)

make[2]: Leaving directory 'C:/MASTERs/22044 TNG3/mTouch_mcc.X'
make[1]: Leaving directory 'C:/MASTERs/22044 TNG3/mTouch_mcc.X'

BUILD SUCCESSFUL (total time: 11s)
Loading code from C:/MASTERs/22044 TNG3/mTouch_mcc.X/dist/default/production/mTouch_mcc.X.production.hex...
Loading completed
```

实验1——在MPLAB® X IDE中使用mTouch®库

16 编程电路板

- 将电路板与编程器（例如，PICkit™ 3）连接
- 打开Project Properties（项目属性）

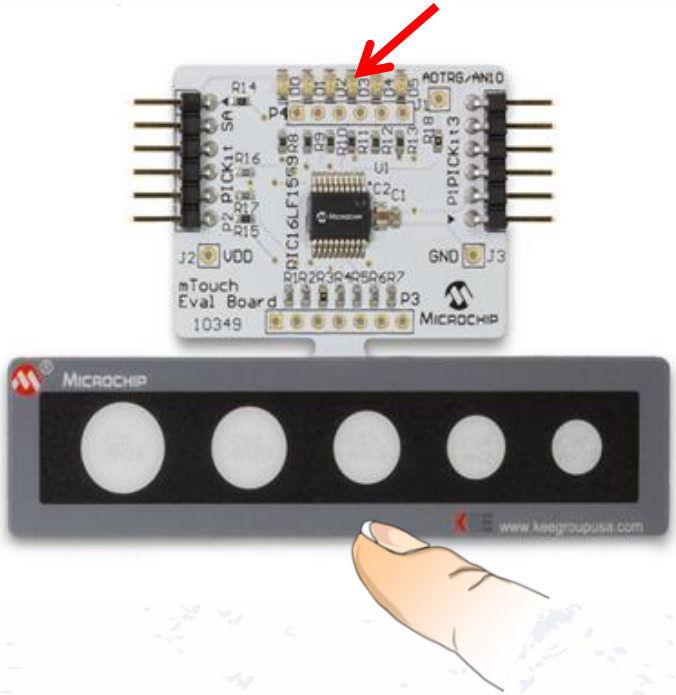


注：
仅限3.25V电压
大小！！

- 单击编程图标编程电路板

实验1——在MPLAB® X IDE中使用mTouch®库

- 当您触摸传感器焊盘时，用于接近检测和触摸的LED将点亮



实验1——在MPLAB® X IDE中使用mTouch®库

恭喜您！
您已经完成了实验1！

附加步骤

- 请遵循主讲人介绍的准则及mTouch库参数调节方法进行调节，以实现防水和防噪声优化。
- 实验5中更详细地介绍了防水和防噪声调节。

实验1总结

- 使用MCC创建新项目
- 选择传感器和配置引脚
- 生成项目
- 添加代码以通过LED指示触摸
- 让触摸轻松上手

MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

MICROCHIP
MASTERs 2018

START和QTouch®模块化库

Atmel START简介

MICROCHIP MASTERs 2018

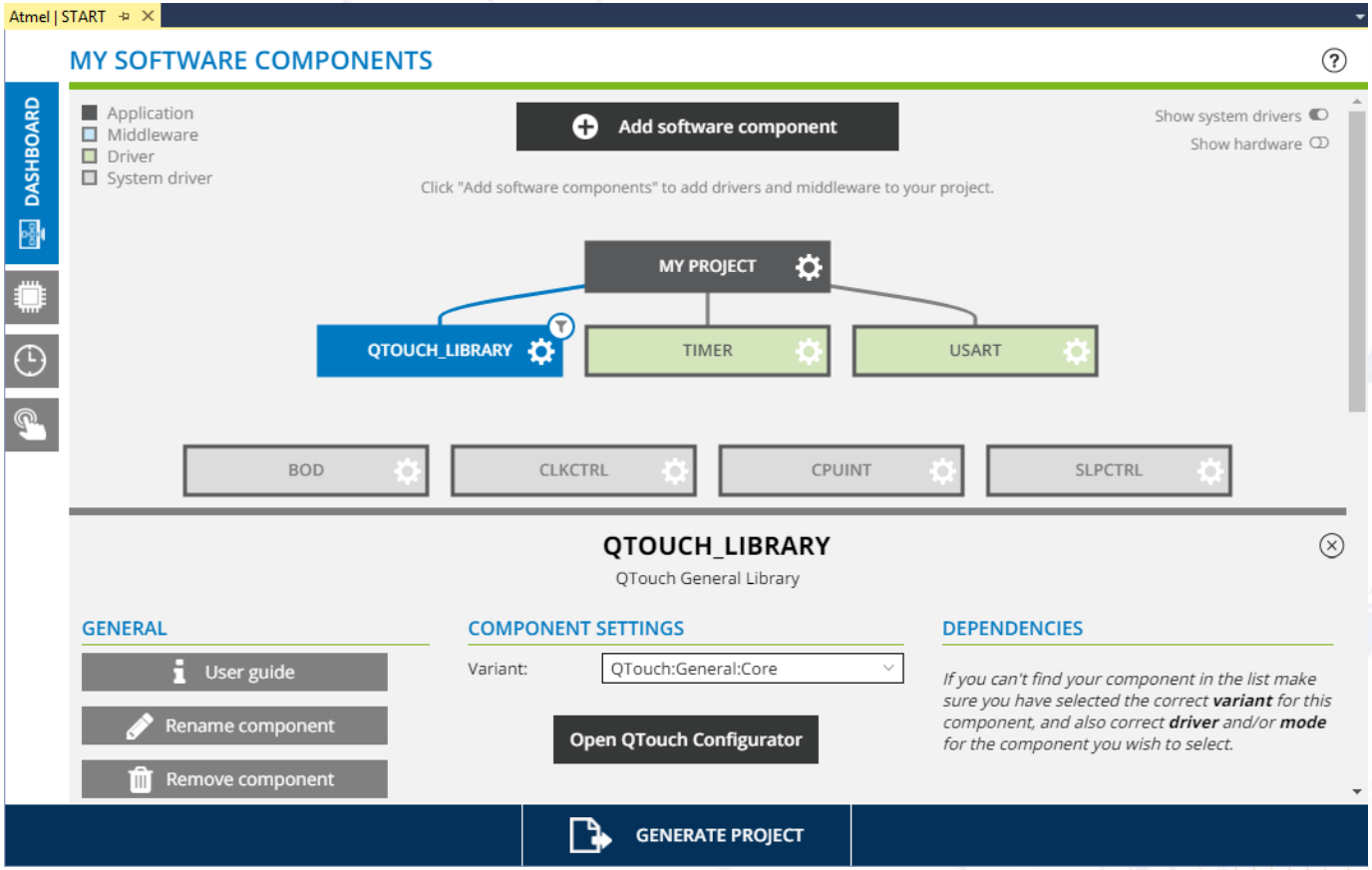
QTouch®解决方案START代码配置器

仪表板

引脚复用配置器

时钟配置器

QTouch配置器



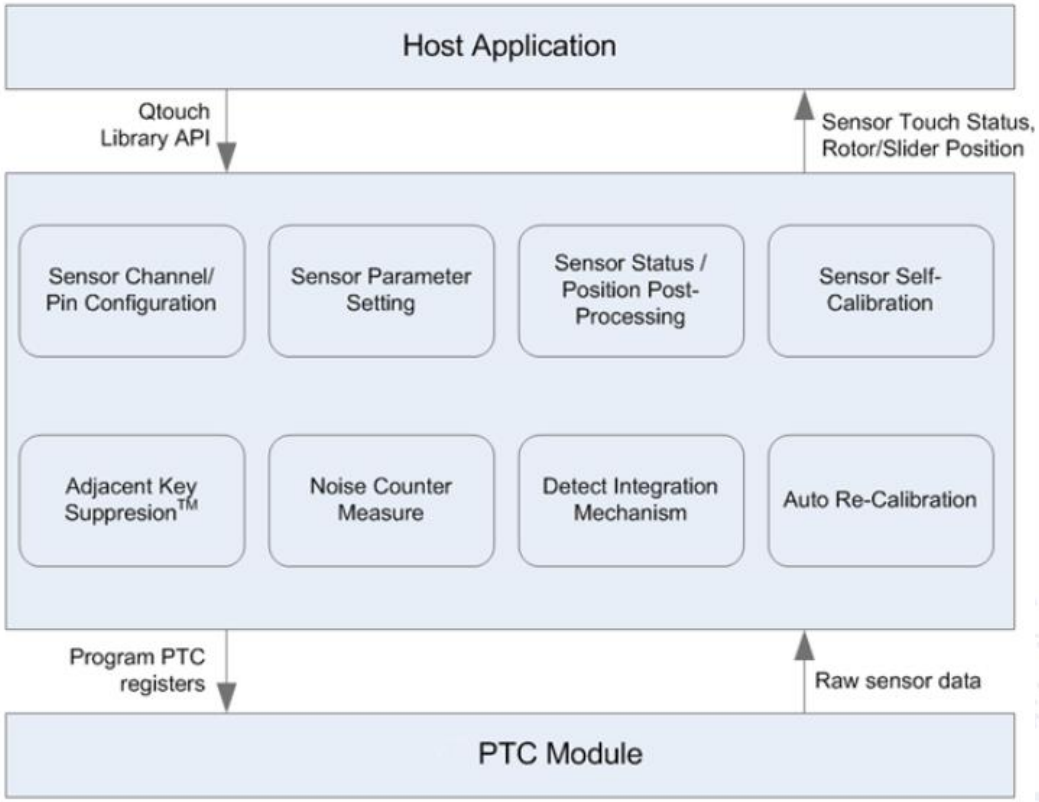
START QTouch®解决方案QTouch®电容式传感库

- START QTouch®电容式触摸库支持具有外设触摸控制器（PTC）的器件，例如AVR® 8位MCU以及SAM 32位MCU和MPU。
- 由START图形用户界面（GUI）生成。
- 适用于所有MCU的Web工具
  - start.atmel.com
- 触摸API有助于缩短开发时间
- 凭借触摸库和IDE，您可以将数十年的触摸专业知识应用到设计中

注：有关详细信息，请参见附录

QTouch®模块化库用户指南

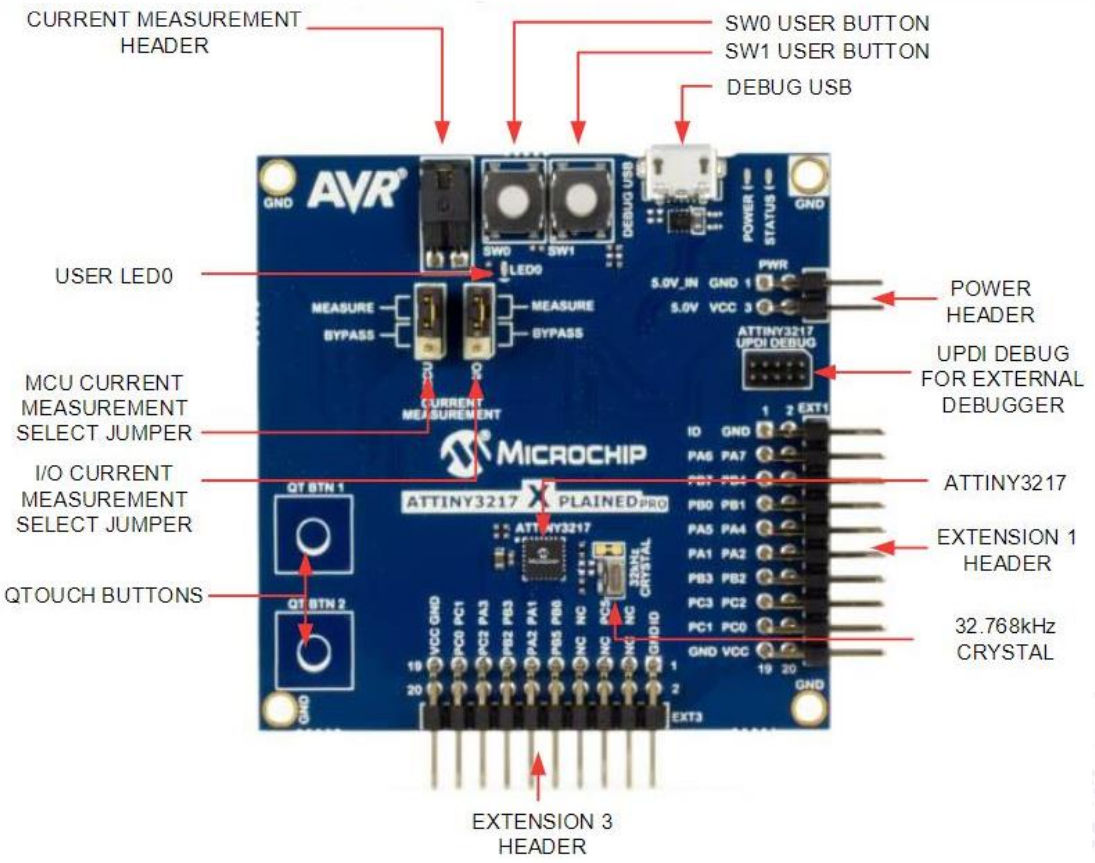
QTouch®模块化库说明



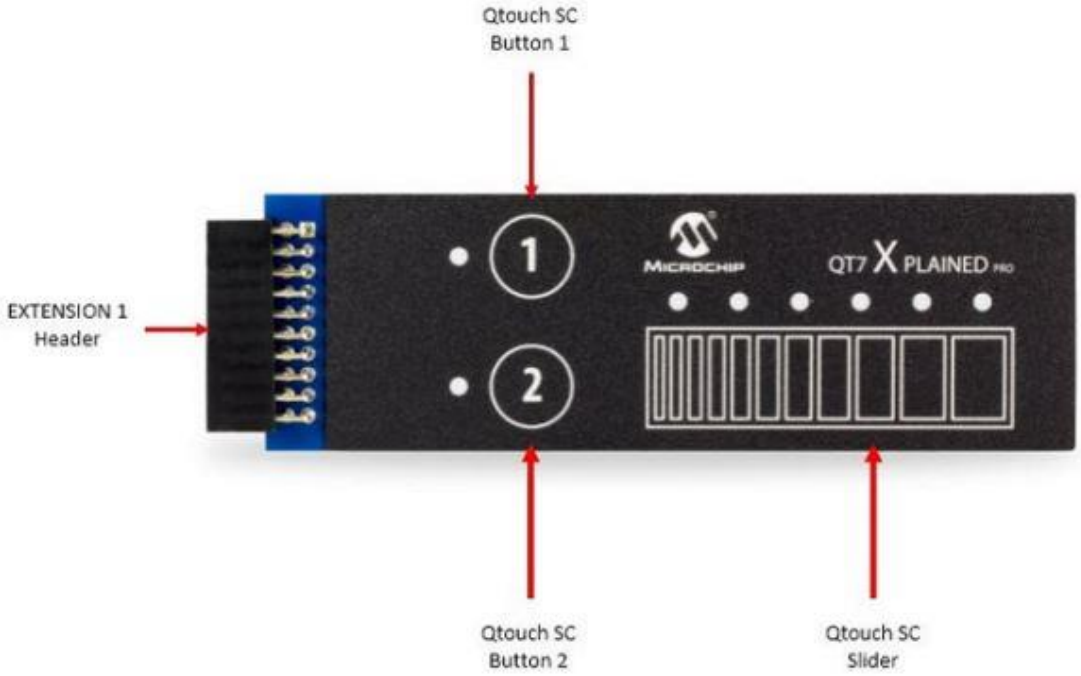
MICROCHIP
MASTERs 2018

实验2：使用Atmel Start和QTouch®模块化库为ATTINY3217-XPRO和ATQT7-XPRO创建项目

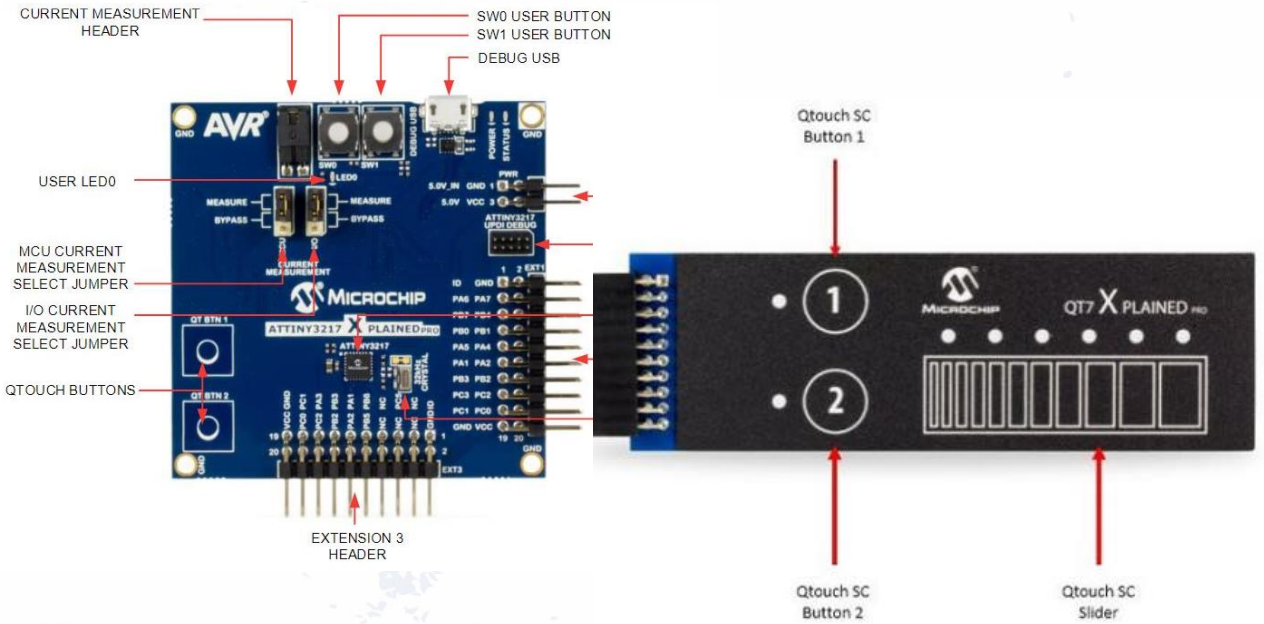
ATTINY3217-XPRO



ATQT7-XPRO

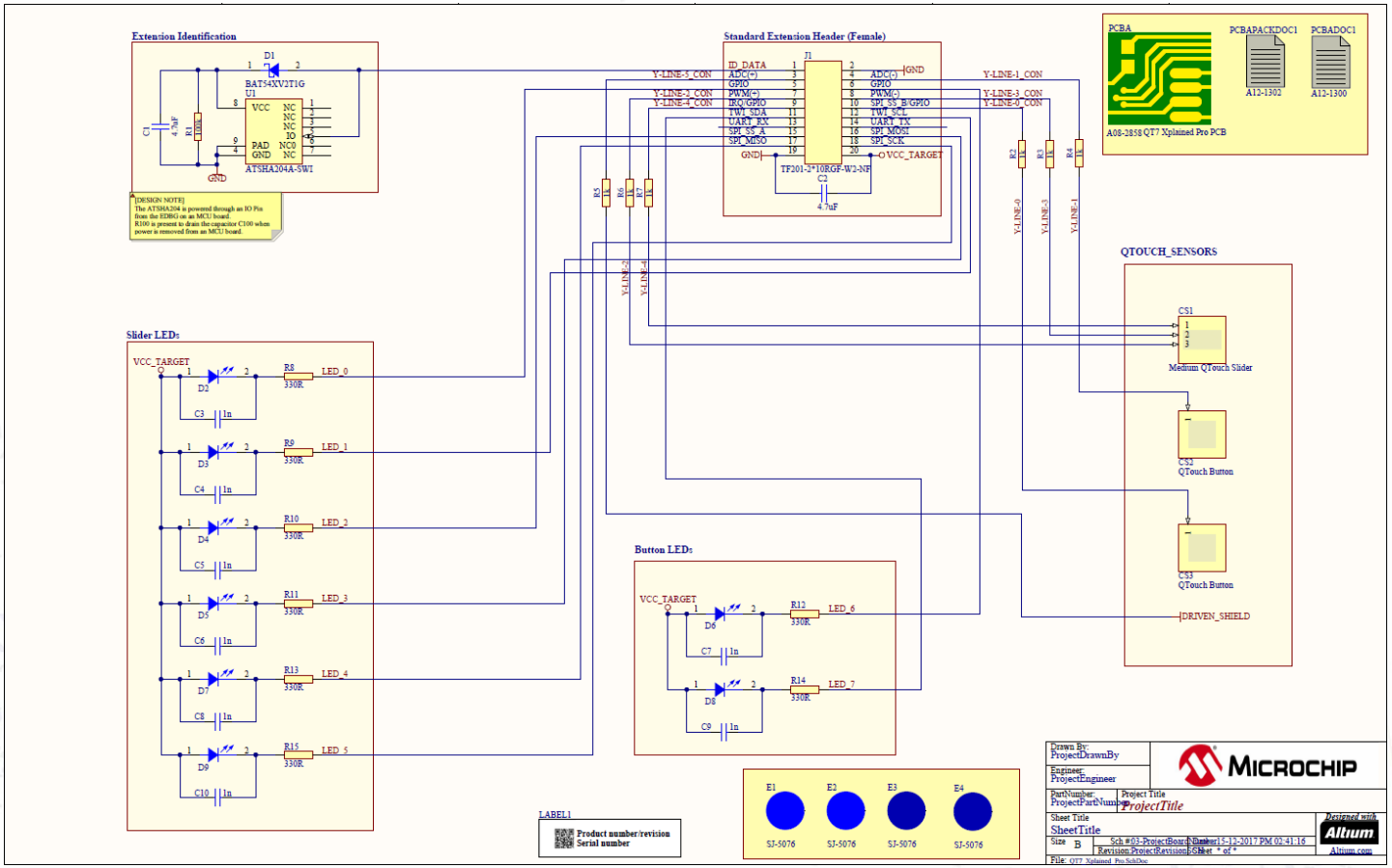


实验设置


CURRENT MEASUREMENT HEADER
SW0 USER BUTTON
SW1 USER BUTTON
DEBUG USB
USER LED0
MCU CURRENT MEASUREMENT SELECT JUMPER
I/O CURRENT MEASUREMENT SELECT JUMPER
QTOUCH BUTTONS
EXTENSION 3 HEADER
Qtouch SC Button 1
Qtouch SC Button 2
Qtouch SC Slider

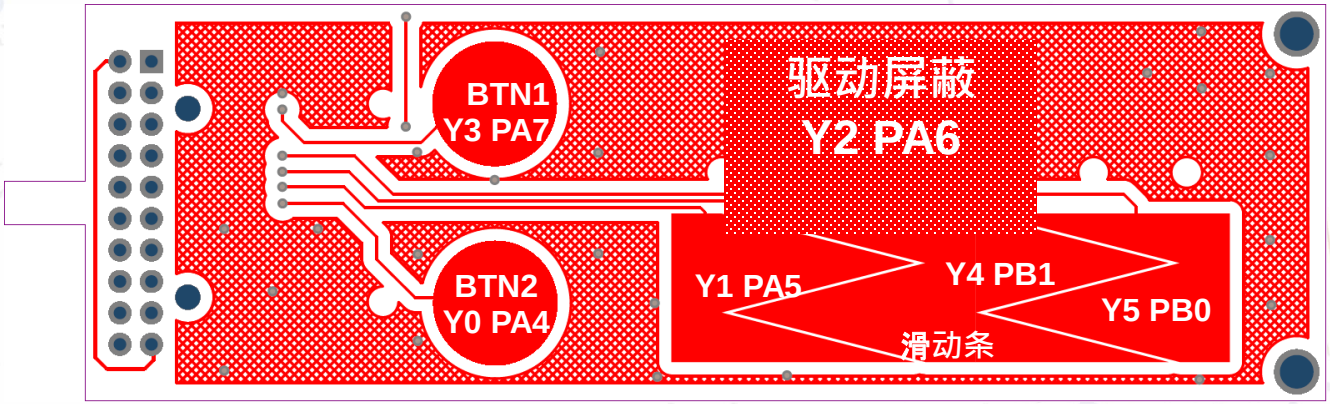
MICROCHIP
MASTERs 2018

ATQT7-XPRO原理图



ATQT7-XPRO PCB布局：传感器

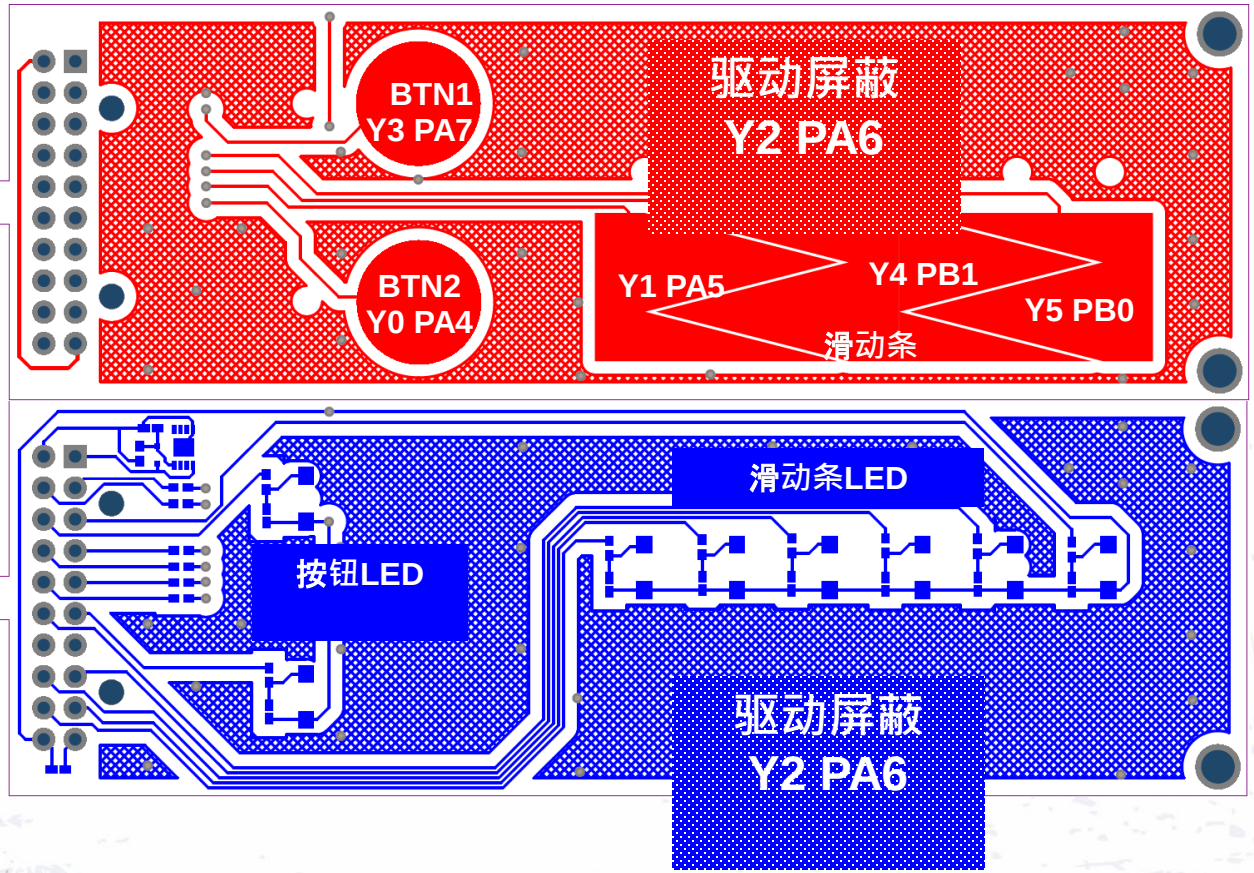
- 观察布局，电路板上有多少触摸传感器？
- 按钮在哪里？
- 滑动条？
- 传感器如何映射到ATtiny3217？
- 其他？



ATQT7-XPRO PCB布局：2层

顶层

底层



QTouch® ATQT7评估工具包连接

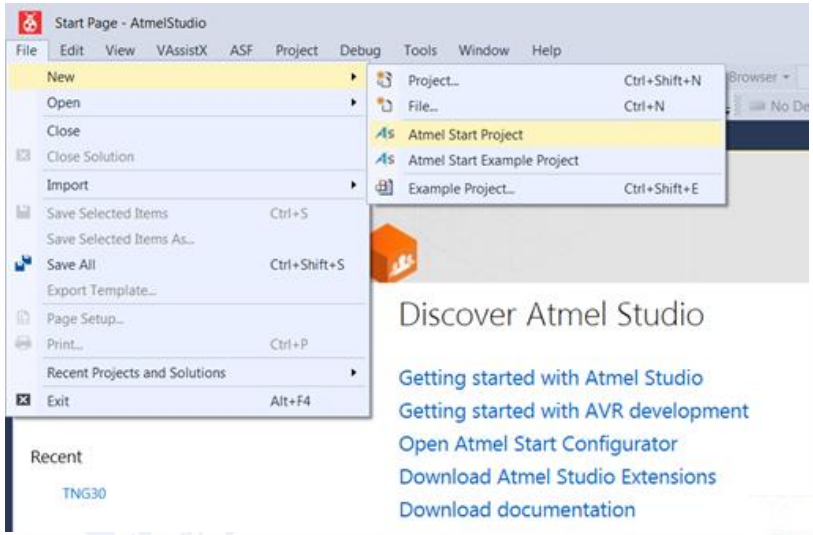
传感器类型	通道ID	引脚选择
按钮0	0	Y3（PA7）
按钮1	1	Y0（PA4）
滑动条0	2	Y1（PA5）
	3	Y4（PB1）
	4	Y5（PB0）
驱动屏蔽		Y2（PA6）

输出	引脚名称
LED_0_BUT	PB4
LED_1_BUT	PA1
LED_0_SLIDER	PB7
LED_1_SLIDER	PA2
LED_2_SLIDER	PC3
LED_3_SLIDER	PC2
LED_4_SLIDER	PC1
LED_5_SLIDER	PC0

实验2目标

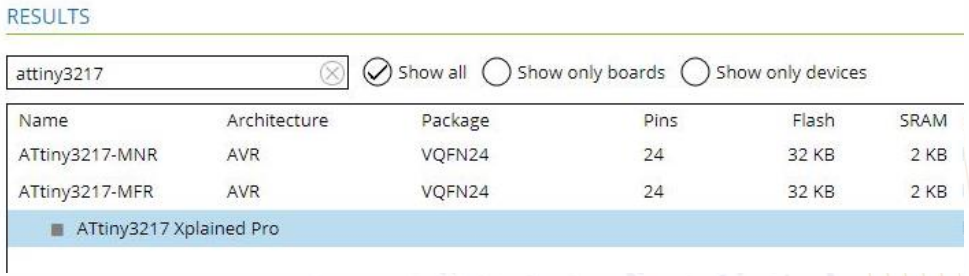
- 使用START和QTouch®模块化库创建一个新项目
- 选择传感器和配置引脚
- 添加代码以通过LED指示触摸

实验2——Atmel Studio中的QTouch® START项目创建

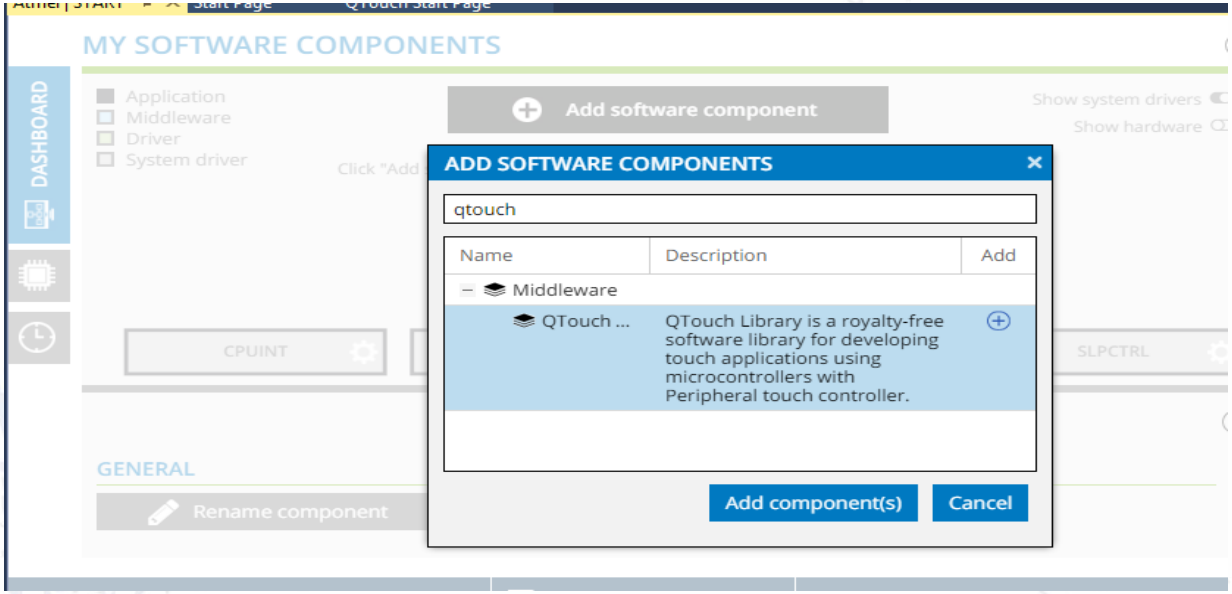


打开Atmel Studio，单击File → New → Atmel Start Project（文件→新建→**Atmel Start项目**）

搜索ATtiny3217并选择ATtiny3217 Xplained pro，然后单击Create New Project（创建新项目）

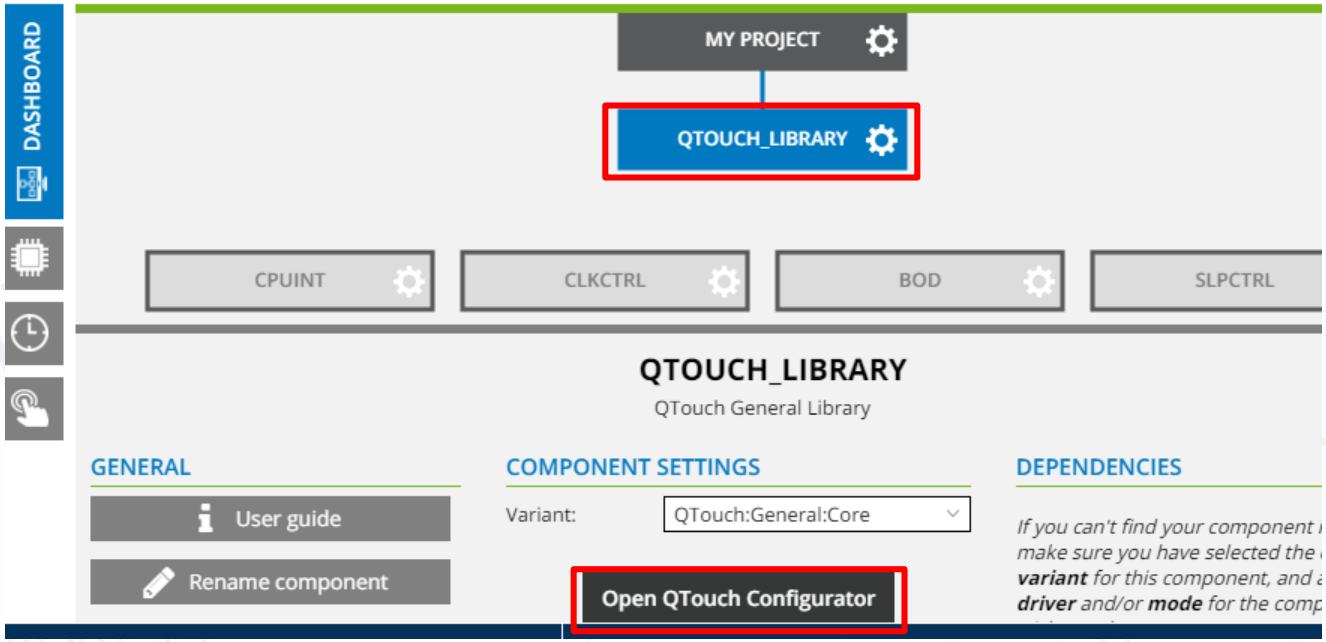


实验2——Atmel Studio中的QTouch® START项目创建



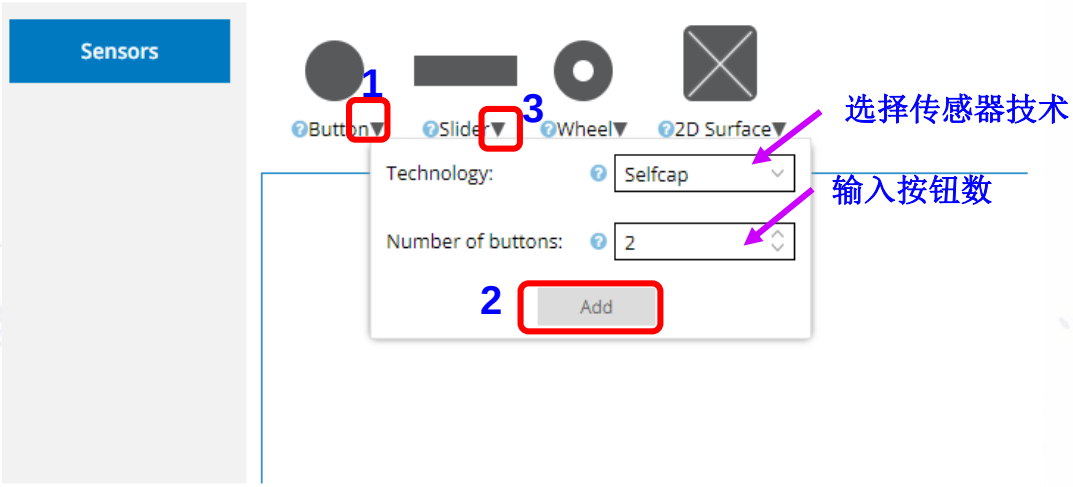
- 单击Add Software components（添加软件组件）
- 搜索QTouch组件，然后将其选中
- 单击Add Components（添加组件）

实验2——Atmel Studio中的QTouch® START项目创建

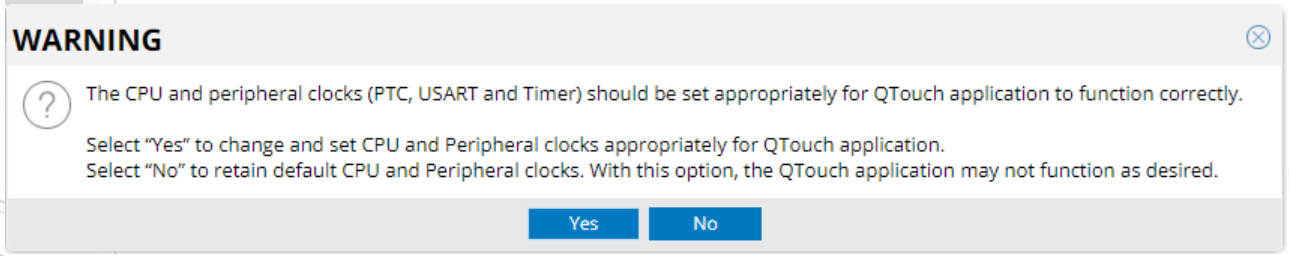


- 单击QTouch库
- 单击Open QTouch Configurator（打开QTouch配置器）

实验2——Atmel Studio中的QTouch® START项目创建

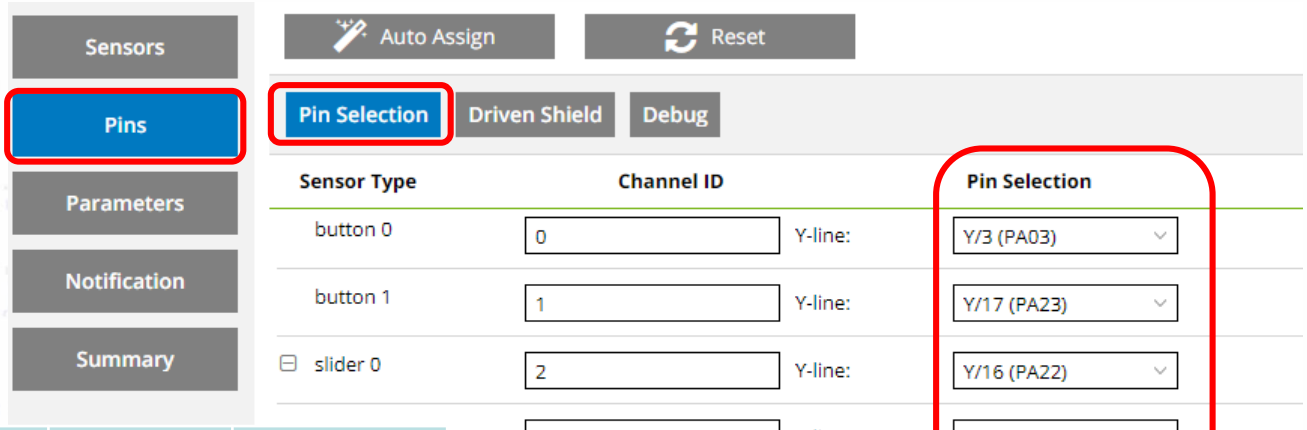


- 根据QT7选择按钮数量和滑动条



- 阅读警告消息并单击Yes（是）

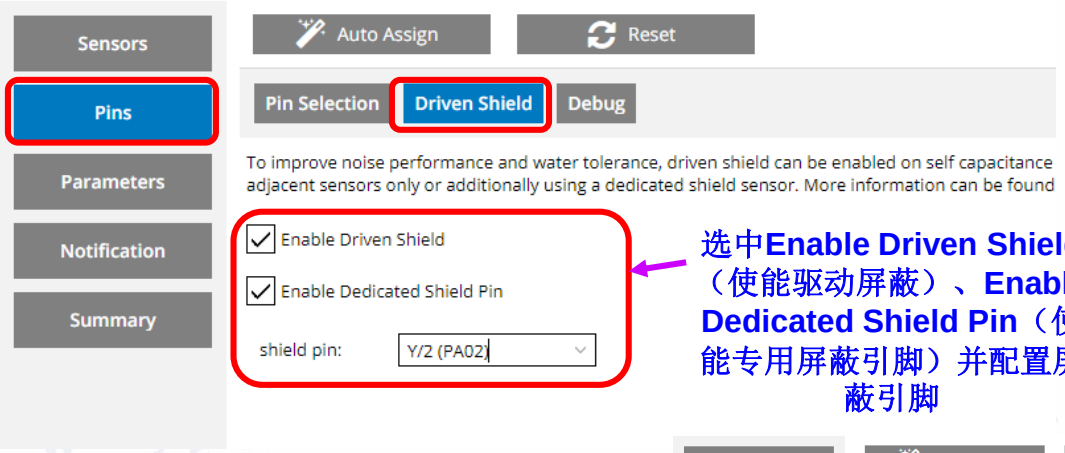
实验2——Atmel Studio中的QTouch® START项目创建



传感器类型	通道ID	引脚选择
按钮0	0	Y3（PA7）
按钮1	1	Y0（PA4）
滑动条0	2	Y1（PA5）
	3	Y4（PB1）
	4	Y5（PB0）
驱动屏蔽		Y2（PA6）

选择PTC引脚以匹配QT7

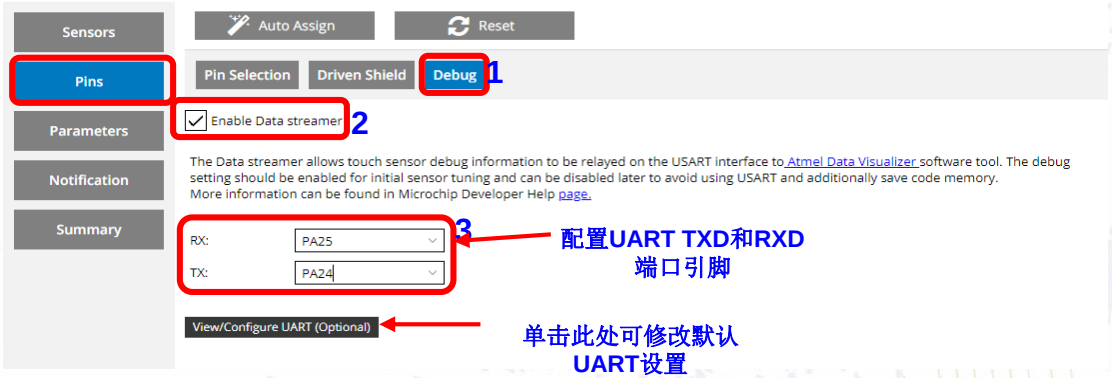
实验2——Atmel Studio中的QTouch® START项目创建



- 使能和配置驱动屏蔽

选中**Enable Driven Shield**（使能驱动屏蔽）、**Enable Dedicated Shield Pin**（使能专用屏蔽引脚）并配置屏蔽引脚

- 使能和配置调试引脚（Data Streamer）

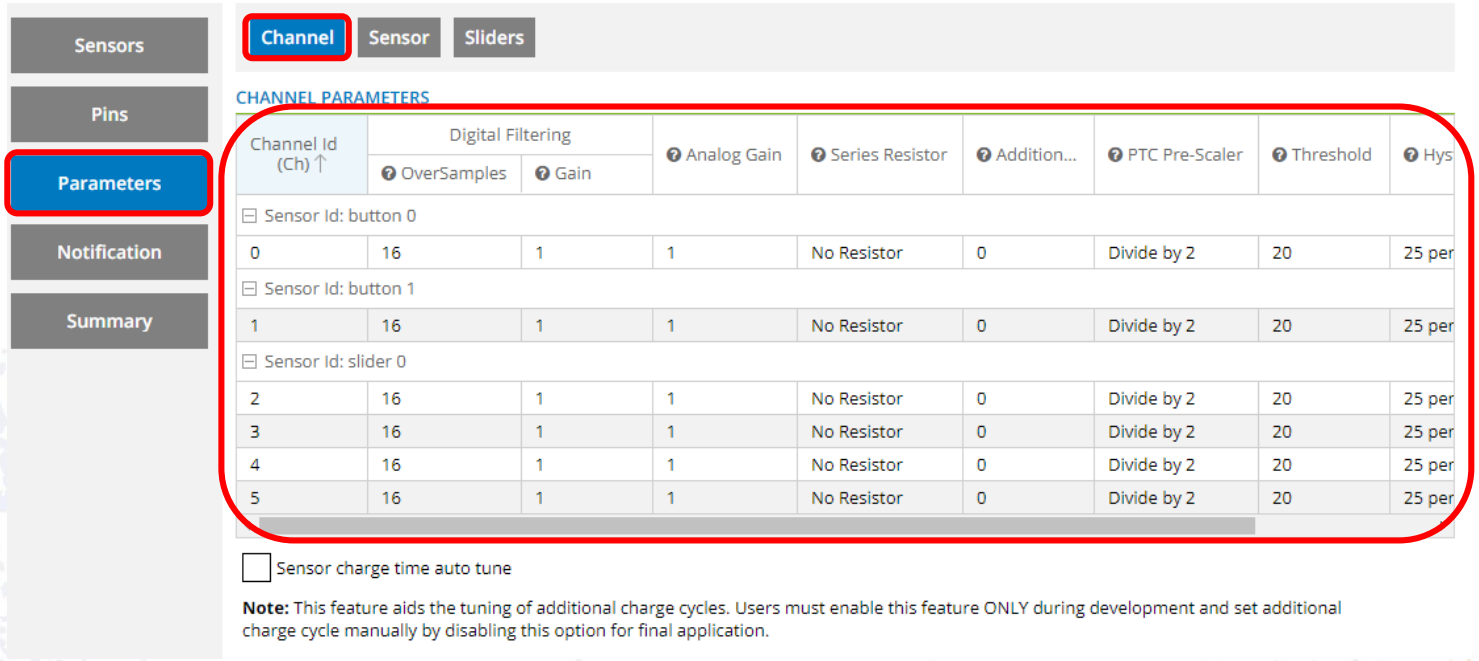


配置UART TXD和RXD端口引脚

单击此处可修改默认UART设置

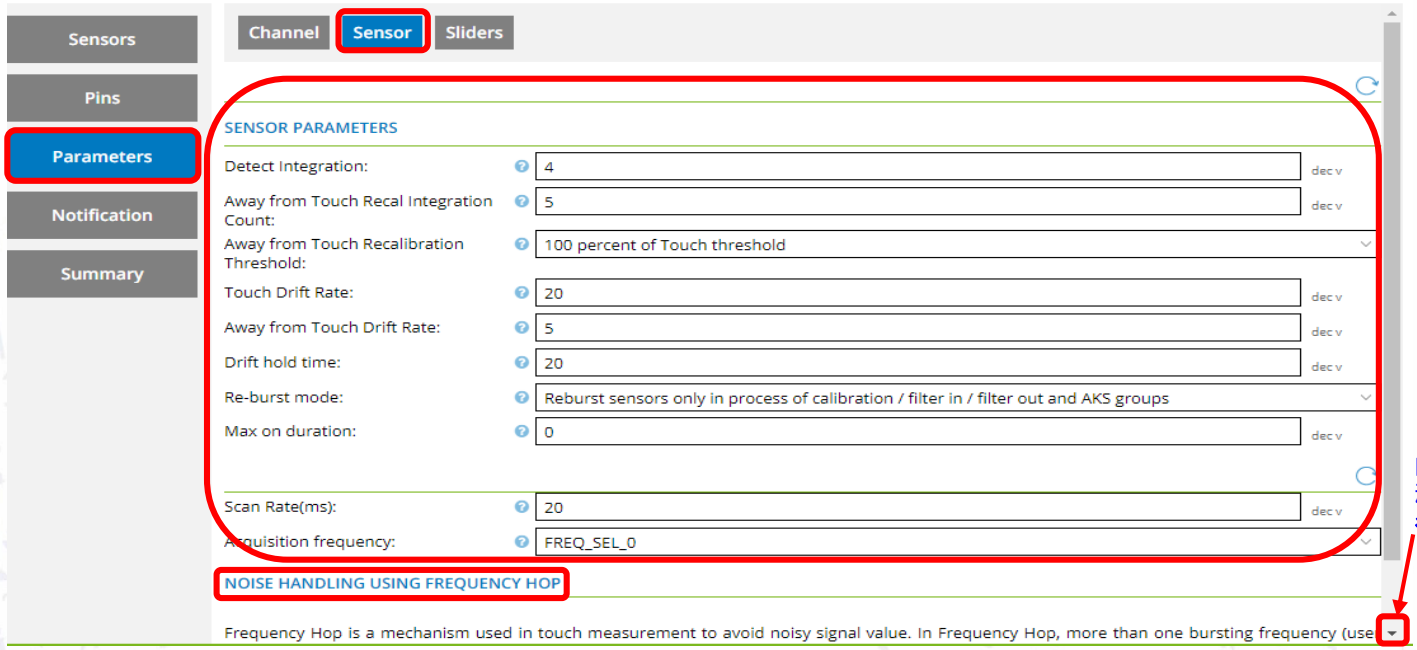
MICROCHIP MASTERs 2018

实验2——Atmel Studio中的QTouch® START项目创建



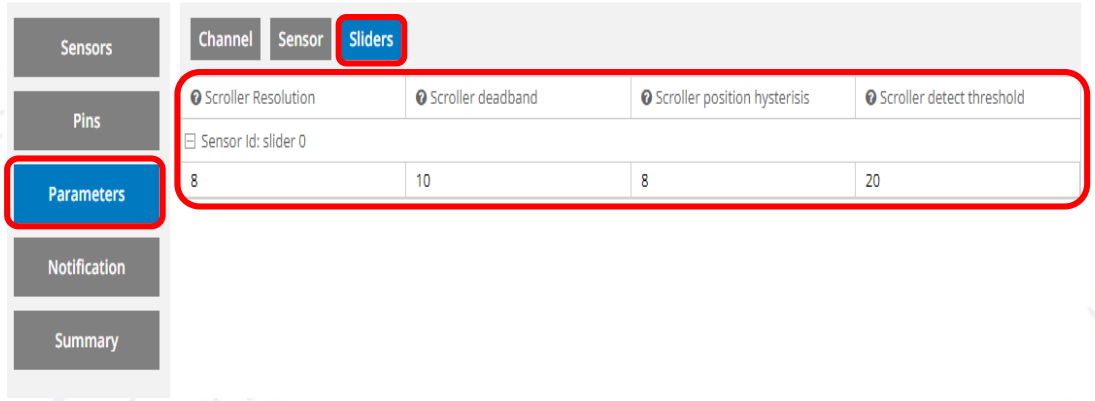
- 配置通道参数：过采样、增益、PTC预分频比和AKS_GROUP等

实验2——Atmel Studio中的QTouch® START项目创建

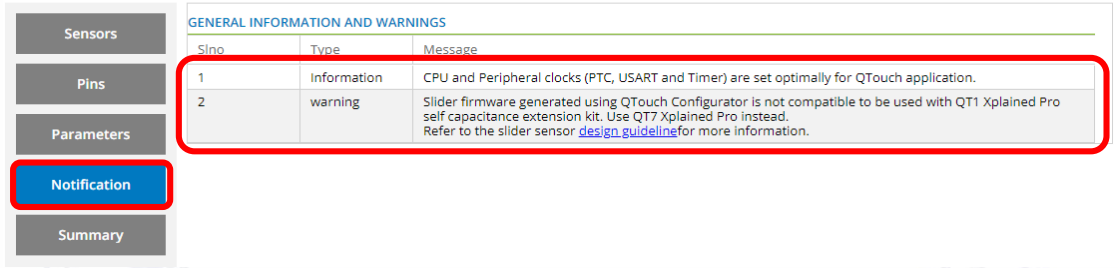


- 配置传感器参数：检测积分、漂移和噪声处理等

实验2——Atmel Studio中的QTouch® START项目创建

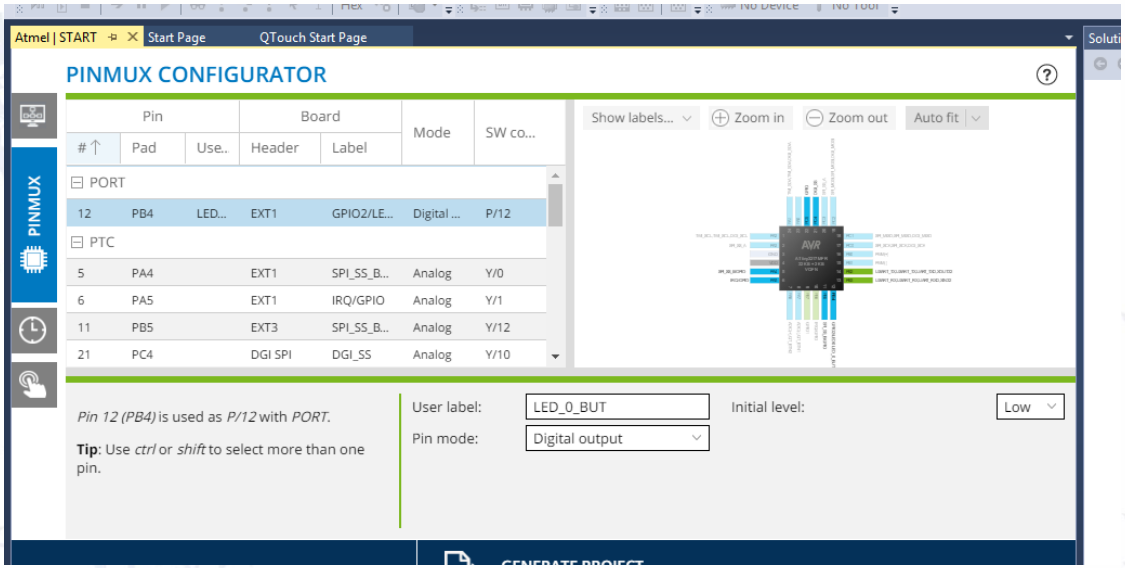


- 单击Sliders（滑动条）选项卡可配置滑动条参数



- 单击Notification（通知）选项卡可查看通知

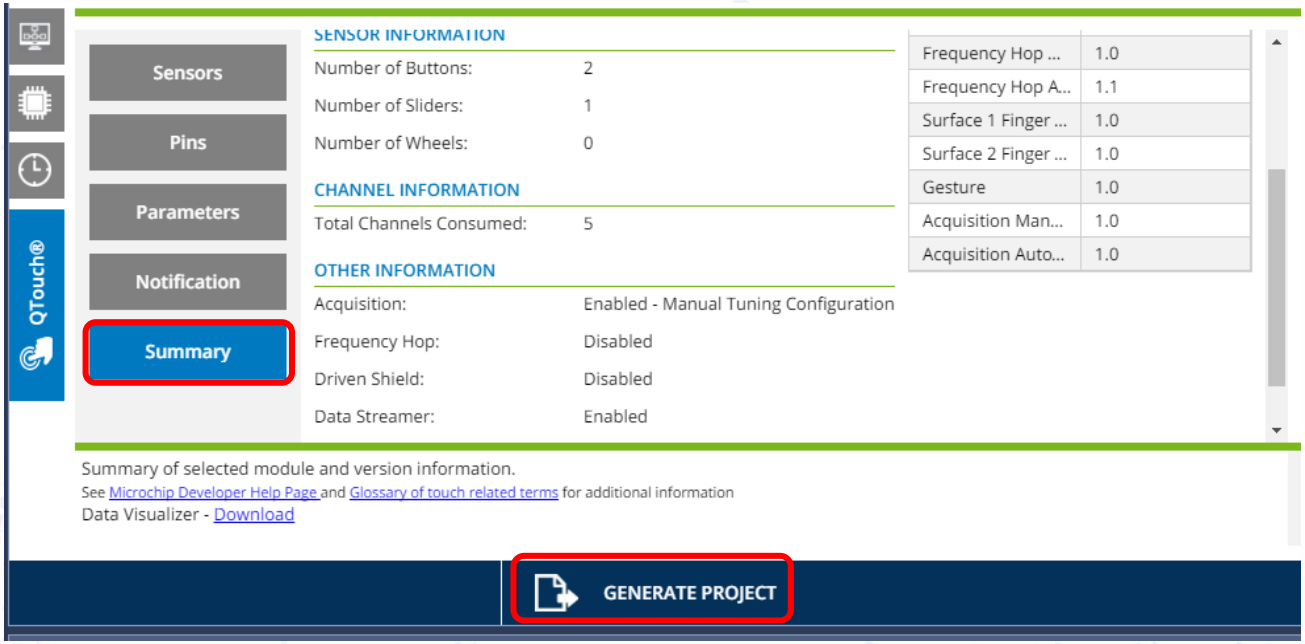
实验2——Atmel Studio中的QTouch® START项目创建



Output	Pin Name
LED_0_BUT	PB4
LED_1_BUT	PA1
LED_0_SLIDER	PB7
LED_1_SLIDER	PA2
LED_2_SLIDER	PC3
LED_3_SLIDER	PC2
LED_4_SLIDER	PC1
LED_5_SLIDER	PC0

- 单击左侧的PINMUX（引脚复用）选项卡，将所有LED驱动GPIO配置为数字输出，并根据需要进行标记。

实验2——Atmel Studio中的QTouch® START项目创建



- 单击Summary（总结）选项卡并验证配置。
- 单击Generate Project（生成项目）生成项目。

实验2总结

- 使用START和QTouch®模块化库创建一个新项目
- 选择传感器和配置引脚
- 添加代码以通过LED指示触摸
- 让触摸轻松上手

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

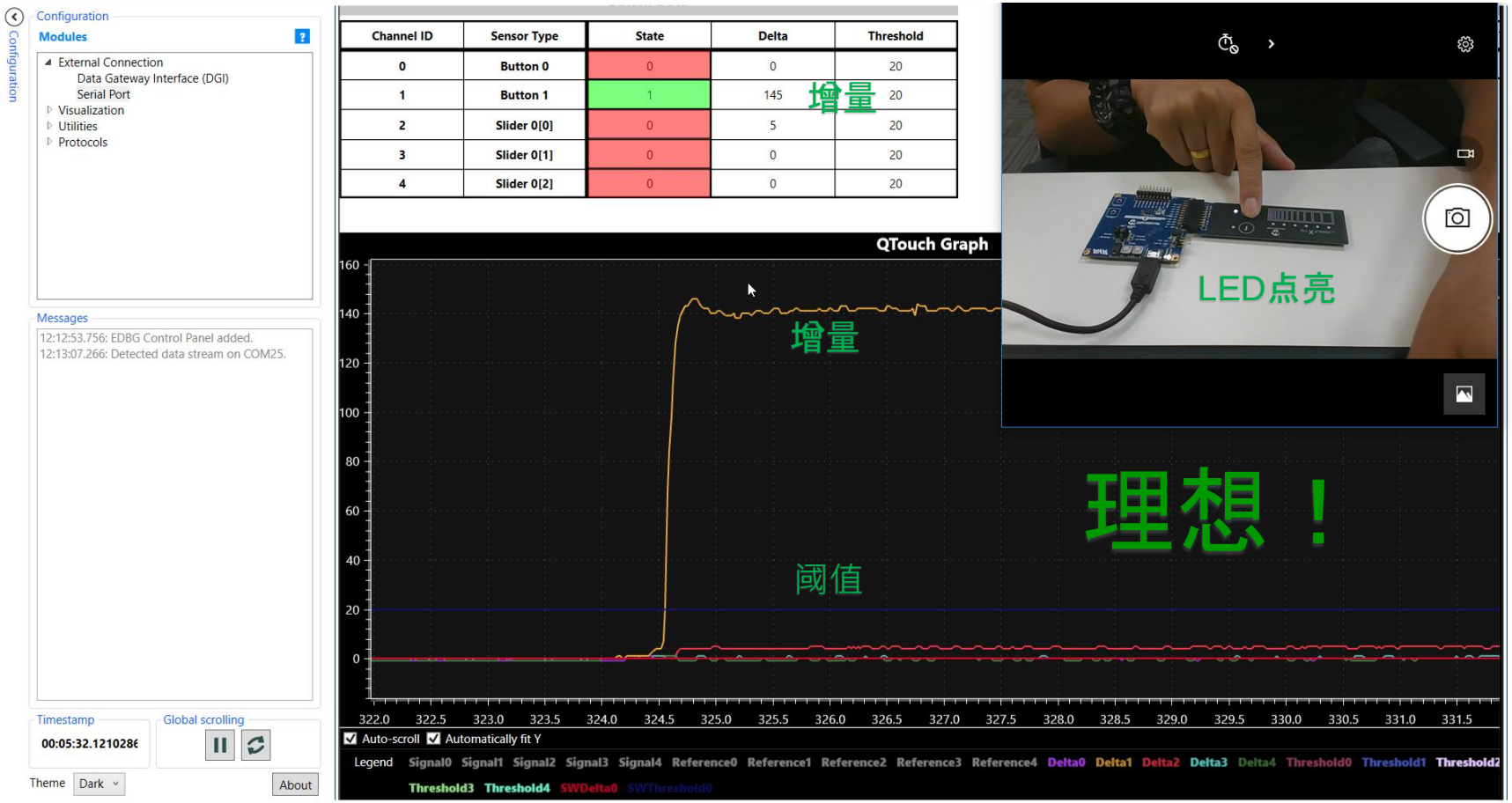
Data Visualizer简介

Data Visualizer的说明

- **Data Visualizer**是用于处理和可视化数据的程序。
- 它可以从各种来源（例如，Xplained Pro板上的嵌入式调试器数据网关接口和COM端口）接收数据。
- 它具有图形功能、示波器、终端以及带有按钮、滑动条和各种指示器的可配置仪表板。
- 它可以解码协议，相应地显示，并记录到文件中。
- 在这里，我们使用Data Visualizer查看实时触摸数据。

实验3：
使用Data Visualizer，以数字和图形的形式显示实时触摸数据

为何选择Data Visualizer？



为何选择Data Visualizer？
5 mm盖板

MICROCHIP MASTERs 2018

实验3目标

- 实现USART通信
- 配置Data Visualizer
- 显示原始和经过处理的触摸数据

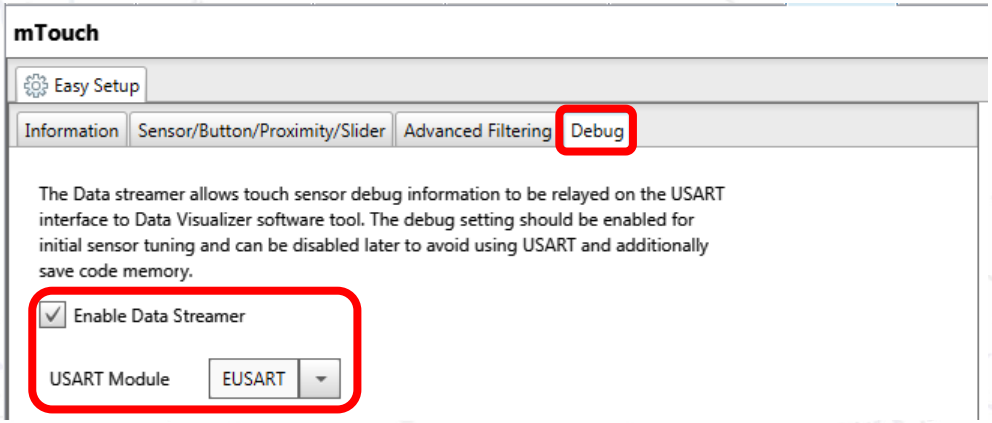
实验3——实现Data Visualizer

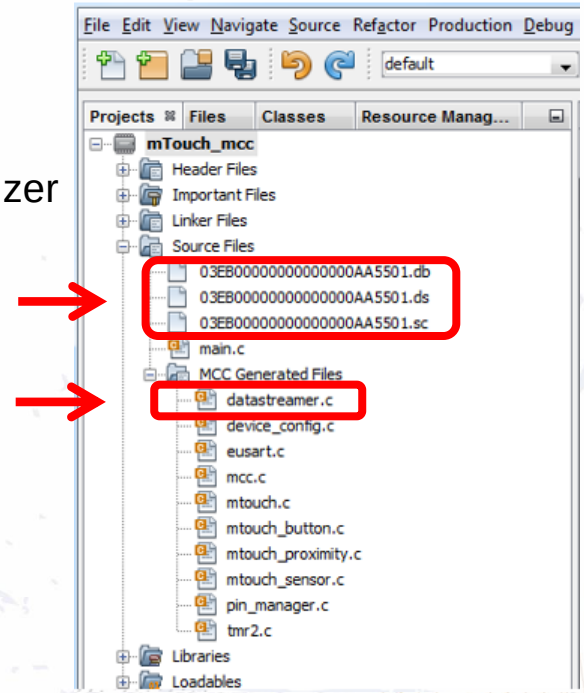
- 如何实现UART并与Data Visualizer连接？
- MCC mTouch®和START QTouch®配置器均需要您选中Debug - Enable Data Streamer（调试 - 使能Data Streamer）功能。
- 如果要生成最终生产代码，可以取消选中Debug - Enable Data Streamer功能。

实验3——mTouch®项目Data Visualizer

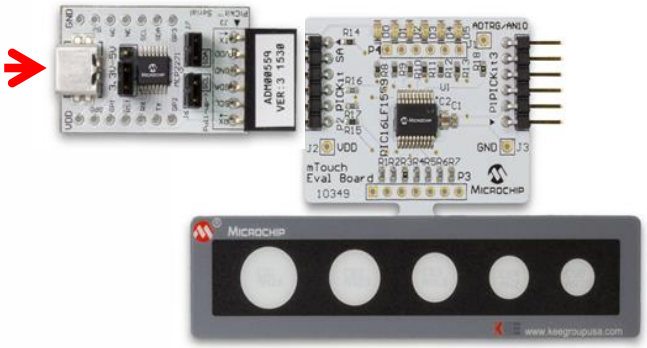
使能Data Streamer（在实验1中完成）：

- 选中MCC mTouch Debug – Enable Data Streamer
- 生成Datastreamer代码
- 生成用于定义数据格式以及如何进行显示的Data Visualizer配置文件



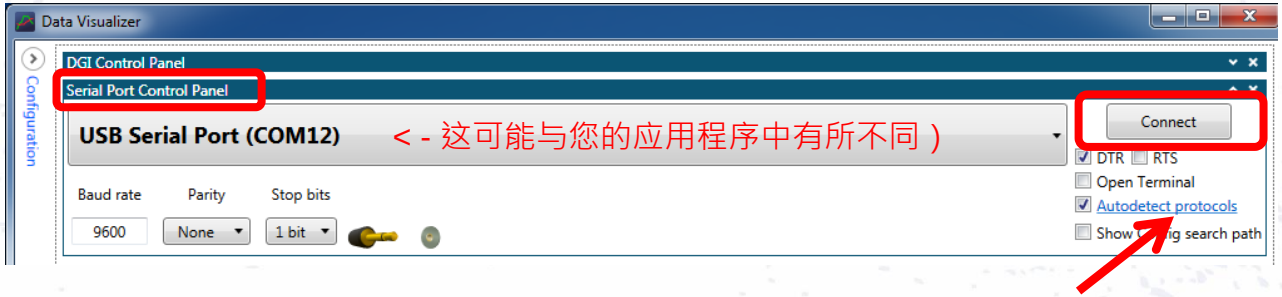


实验3——mTouch®项目Data Visualizer



实现Data Visualizer：

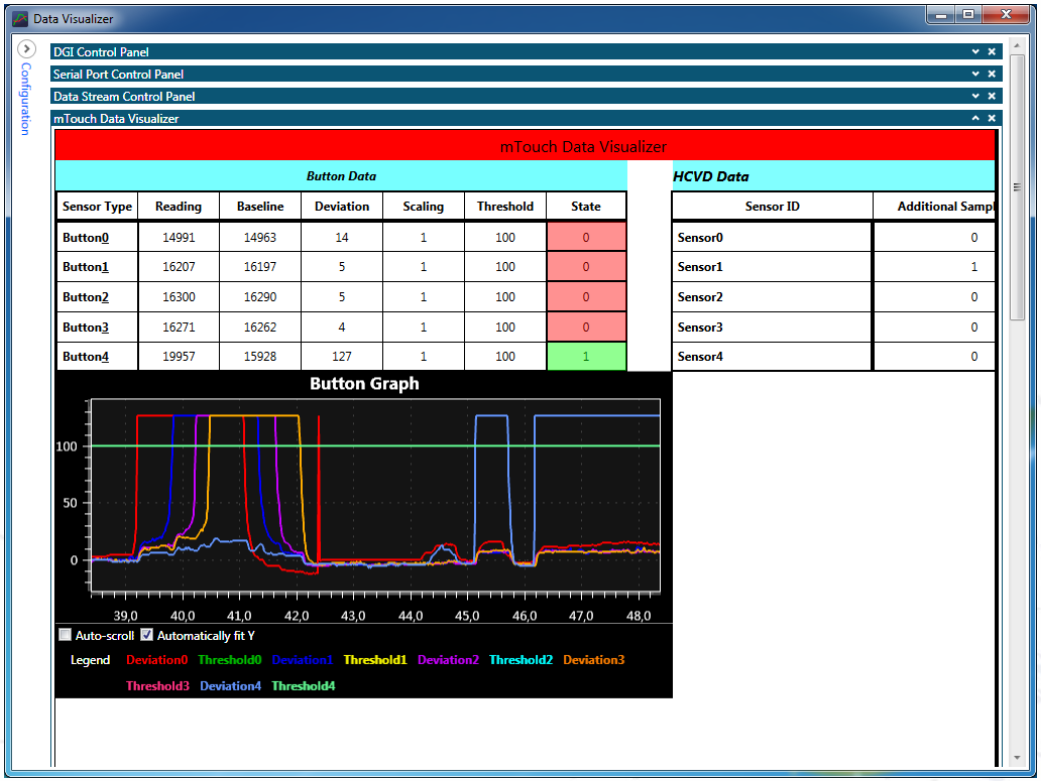
- 将MCP2221转接模块连接到USB端口。
- 打开Data Visualizer（独立版本）。
- 选择Serial Port Control Panel（串行端口控制面板）。
- 为您的硬件选择COM端口。
- 单击*Autodetect protocols*（自动检测协议）并选择Datastreamer files（Datastreamer文件）文件夹。
- 连接串行端口...并等待几秒钟以使波特率同步。



c:\MASTERs\C19H05\mTouch_mcc.X\mcc_generated_files\mtouch\

实验3——mTouch®项目Data Visualizer

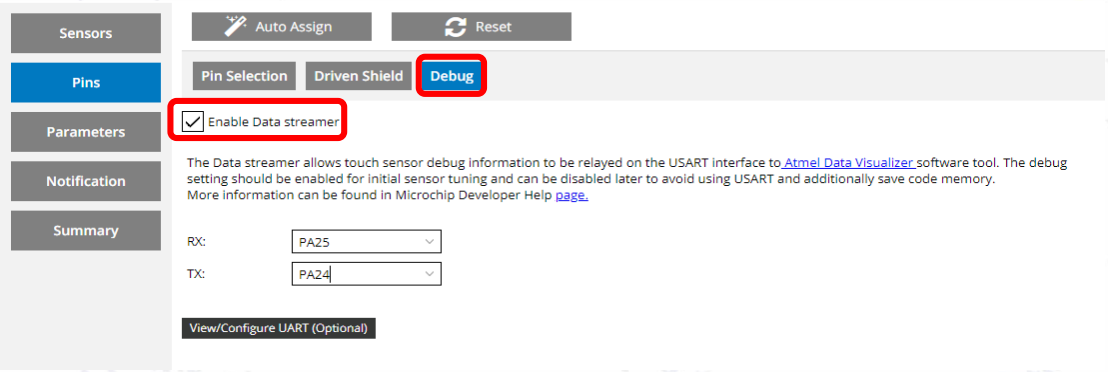
显示实时mTouch技术值：

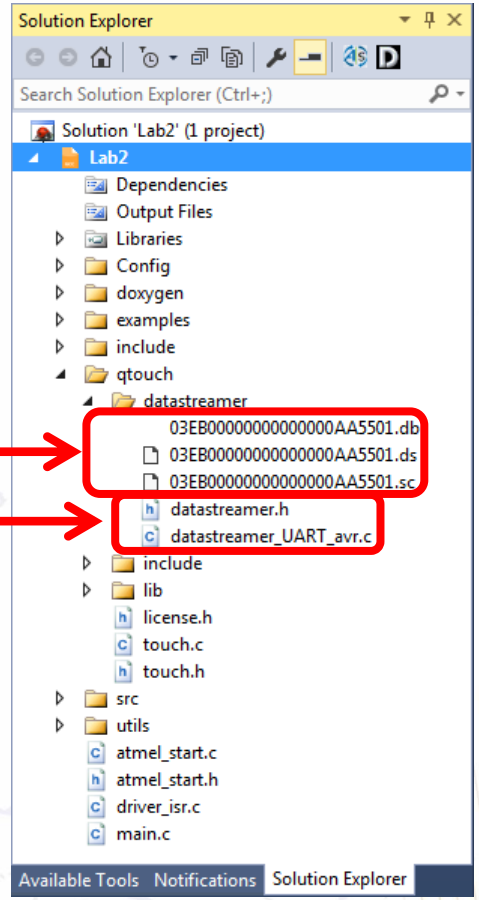


实验3——QTouch® 项目Data Visualizer

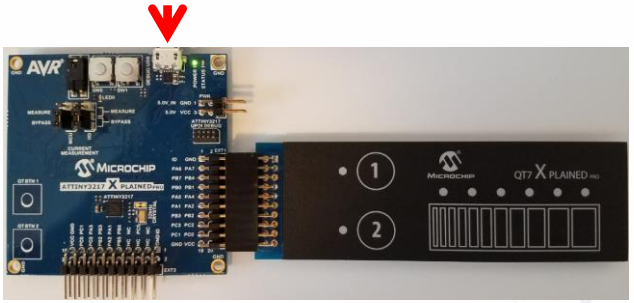
使能Data Streamer（在实验2中完成）：

- 选中START QTouch Debug – Enable Data Streamer
- 生成Datastreamer代码
- 生成用于定义数据格式以及如何进行显示的Data Visualizer配置文件



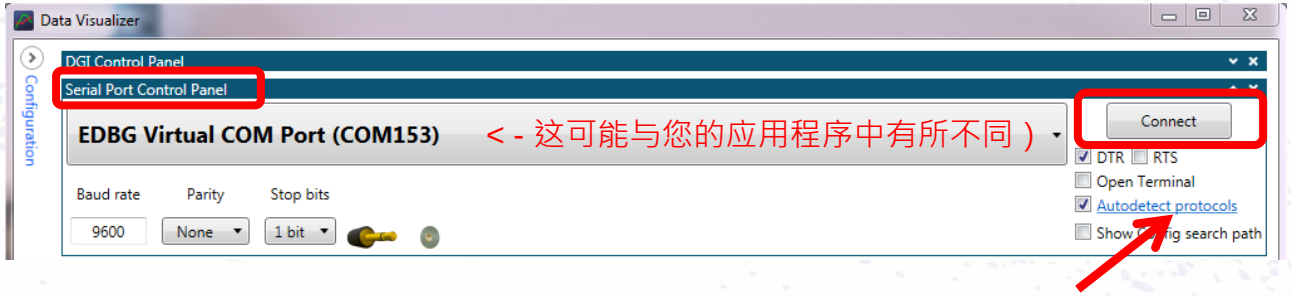


实验3——QTouch® 项目Data Visualizer



实现Data Visualizer：

- 将ATtiny3217-XPRO连接到USB端口。
- 打开Data Visualizer（独立版本）。
- 选择Serial Port Control Panel（串行端口控制面板）。
- 为您的硬件选择COM端口。
- 单击*Autodetect protocols*（自动检测协议）并选择Datastreamer files（Datastreamer文件）文件夹。
- 连接串行端口...并等待几秒钟以使波特率同步。



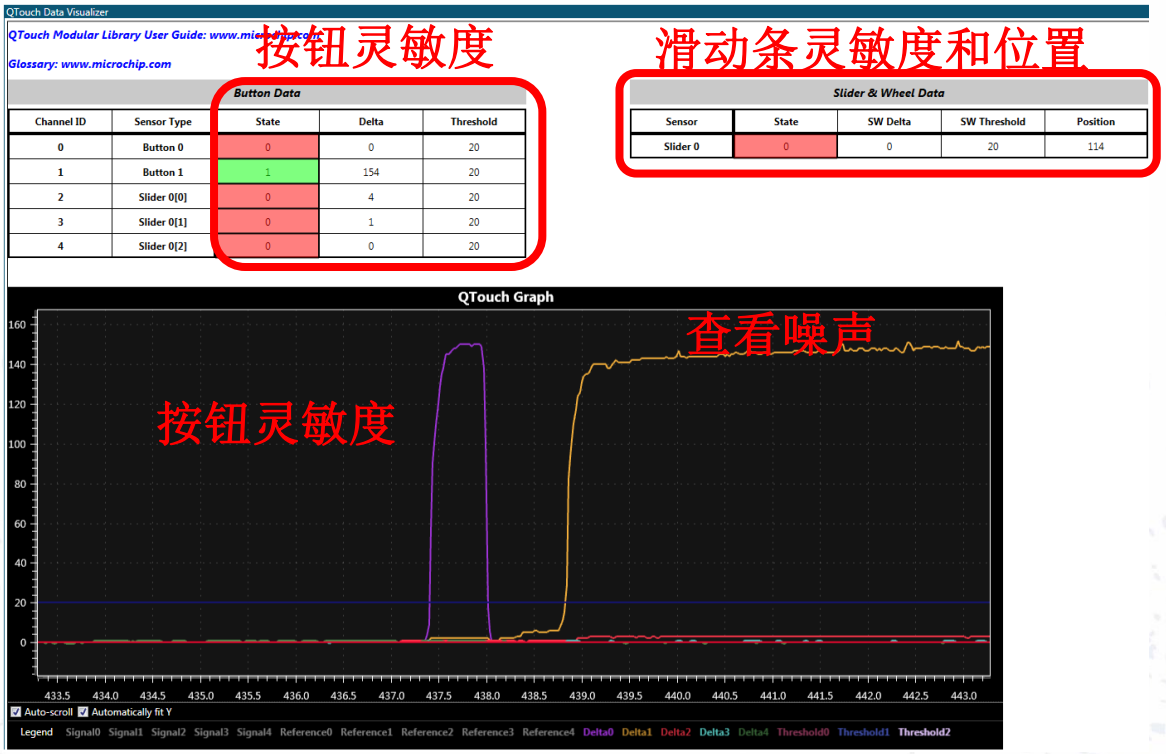
C:\Masters\C19H05\Lab2\Lab2\qtouch\datastreamer

实验3——QTouch® 项目Data Visualizer

显示实时QTouch值：

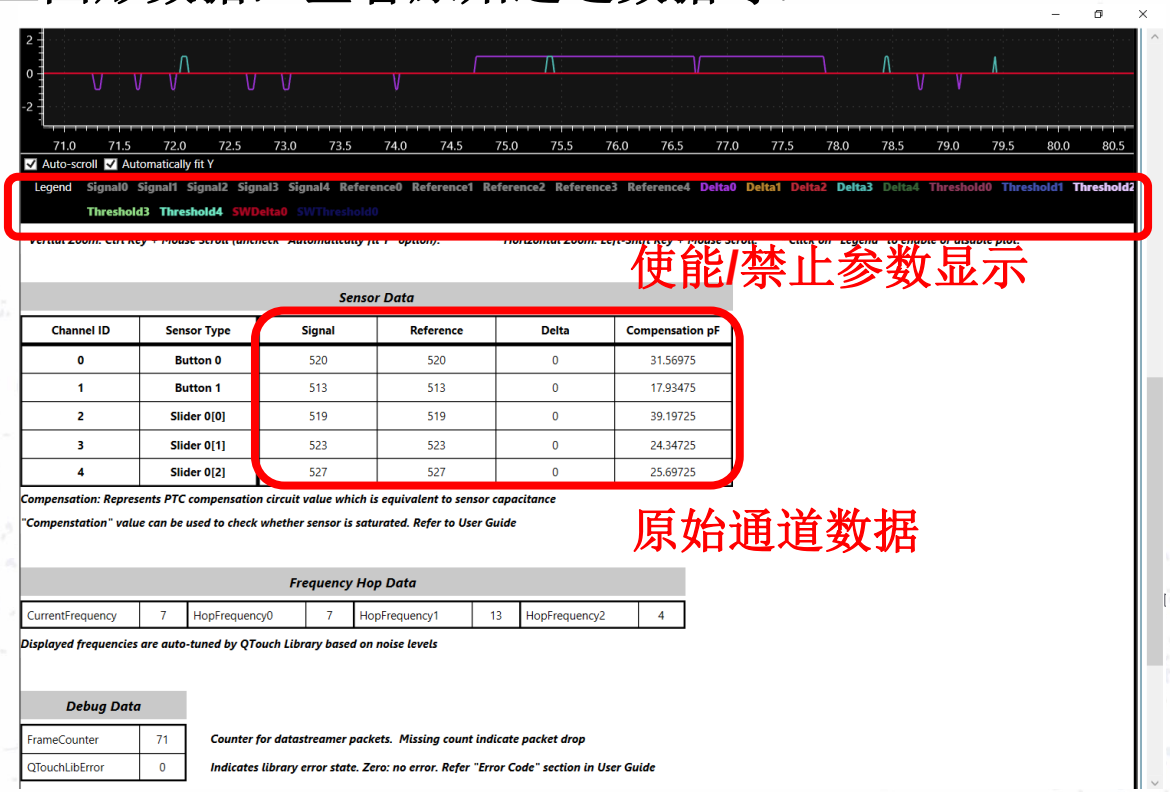
实验3——使用Data Visualizer

查看按钮和滑动条灵敏度、滑动条位置和噪声等：



实验3——使用Data Visualizer

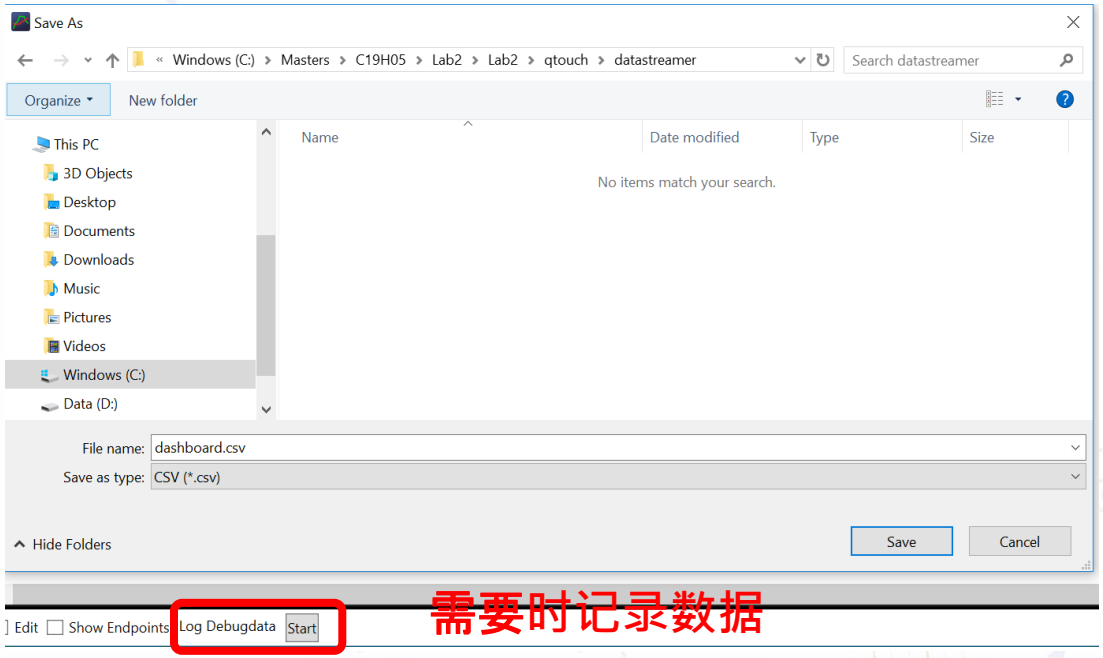
使能/禁止图形数据，查看原始通道数据等：



实验3——使用Data Visualizer

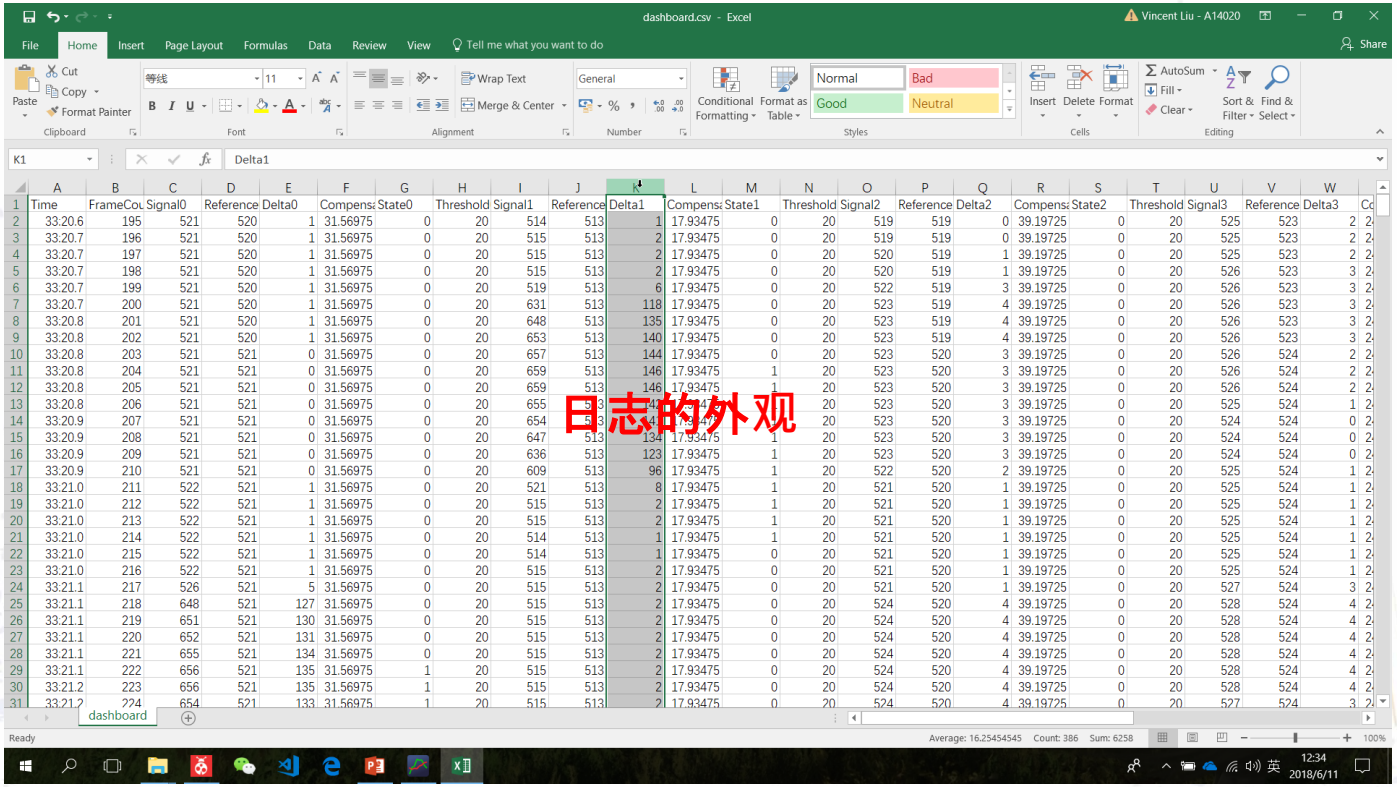
将调试数据记录到文件中：

- 单击*Log Debugdata Start*（开始记录调试数据）
- 输入文件名
- 执行测试
- 单击*Log Debugdata Stop*（停止记录调试数据）
- 文件保存在上面选择的*Autodetect protocols*（自动检测协议）文件夹中。



实验3——使用Data Visualizer

查看调试数据日志：



Time	FrameCou	Signal0	Reference	Delta0	Compens	State0	Threshold	Signal1	Reference	Delta1	Compens	State1	Threshold	Signal2	Reference	Delta2	Compens	State2	Threshold	Signal3	Reference	Delta3
33.20.6	195	521	520	1	31.56975	0	20	514	513	1	17.93475	0	20	519	519	0	39.19725	0	20	525	523	2
33.20.7	196	521	520	1	31.56975	0	20	515	513	2	17.93475	0	20	519	519	0	39.19725	0	20	525	523	2
33.20.7	197	521	520	1	31.56975	0	20	515	513	2	17.93475	0	20	520	519	1	39.19725	0	20	525	523	2
33.20.7	198	521	520	1	31.56975	0	20	515	513	2	17.93475	0	20	520	519	1	39.19725	0	20	526	523	3
33.20.7	199	521	520	1	31.56975	0	20	519	513	6	17.93475	0	20	522	519	3	39.19725	0	20	526	523	3
33.20.7	200	521	520	1	31.56975	0	20	631	513	118	17.93475	0	20	523	519	4	39.19725	0	20	526	523	3
33.20.8	201	521	520	1	31.56975	0	20	648	513	135	17.93475	0	20	523	519	4	39.19725	0	20	526	523	3
33.20.8	202	521	520	1	31.56975	0	20	653	513	140	17.93475	0	20	523	519	4	39.19725	0	20	526	523	3
33.20.8	203	521	521	0	31.56975	0	20	657	513	144	17.93475	0	20	523	520	3	39.19725	0	20	526	524	2
33.20.8	204	521	521	0	31.56975	0	20	659	513	146	17.93475	1	20	523	520	3	39.19725	0	20	526	524	2
33.20.8	205	521	521	0	31.56975	0	20	659	513	146	17.93475	1	20	523	520	3	39.19725	0	20	526	524	2
33.20.8	206	521	521	0	31.56975	0	20	655	513	142	17.93475	1	20	523	520	3	39.19725	0	20	525	524	1
33.20.9	207	521	521	0	31.56975	0	20	654	513	141	17.93475	1	20	523	520	3	39.19725	0	20	524	524	0
33.20.9	208	521	521	0	31.56975	0	20	647	513	134	17.93475	1	20	523	520	3	39.19725	0	20	524	524	0
33.20.9	209	521	521	0	31.56975	0	20	636	513	123	17.93475	1	20	523	520	3	39.19725	0	20	524	524	0
33.20.9	210	521	521	0	31.56975	0	20	609	513	96	17.93475	1	20	522	520	2	39.19725	0	20	525	524	1
33.21.0	211	522	521	1	31.56975	0	20	521	513	8	17.93475	1	20	521	520	1	39.19725	0	20	525	524	1
33.21.0	212	522	521	1	31.56975	0	20	515	513	2	17.93475	1	20	521	520	1	39.19725	0	20	525	524	1
33.21.0	213	522	521	1	31.56975	0	20	515	513	2	17.93475	1	20	521	520	1	39.19725	0	20	525	524	1
33.21.0	214	522	521	1	31.56975	0	20	514	513	1	17.93475	1	20	521	520	1	39.19725	0	20	525	524	1
33.21.0	215	522	521	1	31.56975	0	20	514	513	1	17.93475	0	20	521	520	1	39.19725	0	20	525	524	1
33.21.0	216	522	521	1	31.56975	0	20	515	513	2	17.93475	0	20	521	520	1	39.19725	0	20	525	524	1
33.21.1	217	526	521	5	31.56975	0	20	515	513	2	17.93475	0	20	521	520	1	39.19725	0	20	527	524	3
33.21.1	218	648	521	127	31.56975	0	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	528	524	4
33.21.1	219	651	521	130	31.56975	0	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	528	524	4
33.21.1	220	652	521	131	31.56975	0	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	528	524	4
33.21.1	221	655	521	134	31.56975	0	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	528	524	4
33.21.1	222	656	521	135	31.56975	1	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	528	524	4
33.21.2	223	656	521	135	31.56975	1	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	528	524	4
33.21.2	224	654	521	133	31.56975	1	20	515	513	2	17.93475	0	20	524	520	4	39.19725	0	20	527	524	3

实验3总结

- 实现USART通信
- 配置Data Visualizer
- 显示原始和经过处理的触摸数据

MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

实验4：使用Data Visualizer和QTouch®模块化库参数来调节触摸系统

调节参数概述

- 灵敏度：
  增益和阈值
- 噪声抑制：
  检测积分（DI）、滤波等级、跳频和迟滞
- 取决于传感器：
  充电时间/预分频比
- 环境：
  漂移补偿
- 取决于应用：
  扫描速率、AKS和最大开启持续时间

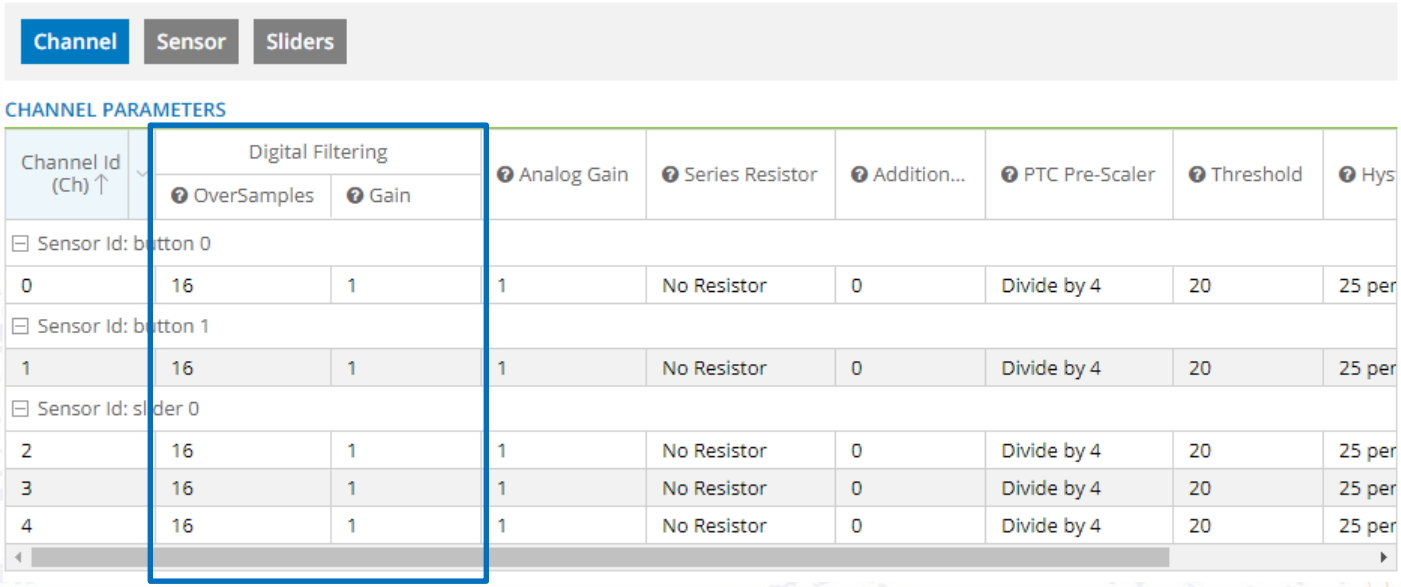
覆盖厚度

- 我们将为ATQT7-XPRO添加塑料覆盖层。
- 这将如何影响触控性能？
- 应更改哪些调节参数？

调节参数的说明

通道配置参数

- 数字滤波
  - 过采样
  - 增益



Channel Id (Ch)	Digital Filtering		Analog Gain	Series Resistor	Addition...	PTC Pre-Scaler	Threshold	Hys
	OverSamples	Gain						
Sensor Id: button 0								
0	16	1	1	No Resistor	0	Divide by 4	20	25 per
Sensor Id: button 1								
1	16	1	1	No Resistor	0	Divide by 4	20	25 per
Sensor Id: slider 0								
2	16	1	1	No Resistor	0	Divide by 4	20	25 per
3	16	1	1	No Resistor	0	Divide by 4	20	25 per
4	16	1	1	No Resistor	0	Divide by 4	20	25 per

调节参数的说明

通道配置参数

- 模拟增益

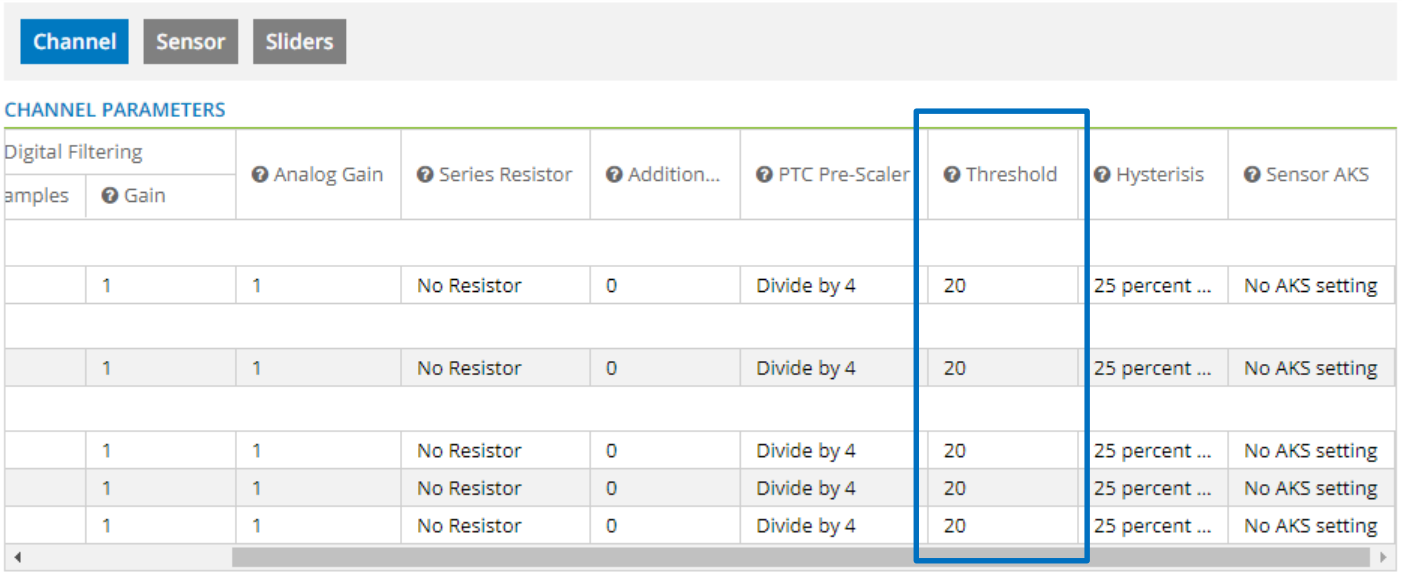
Channel | Sensor | Sliders

CHANNEL PARAMETERS

Channel Id (Ch)	Digital Filtering		Analog Gain	Series Resistor	Addition...	PTC Pre-Scaler	Threshold	Hys
	OverSamples	Gain						
Sensor Id: button 0								
0	16	1	1	No Resistor	0	Divide by 4	20	25 per
Sensor Id: button 1								
1	16	1	1	No Resistor	0	Divide by 4	20	25 per
Sensor Id: slider 0								
2	16	1	1	No Resistor	0	Divide by 4	20	25 per
3	16	1	1	No Resistor	0	Divide by 4	20	25 per
4	16	1	1	No Resistor	0	Divide by 4	20	25 per

调节参数的说明
通道配置参数

- 传感器阈值

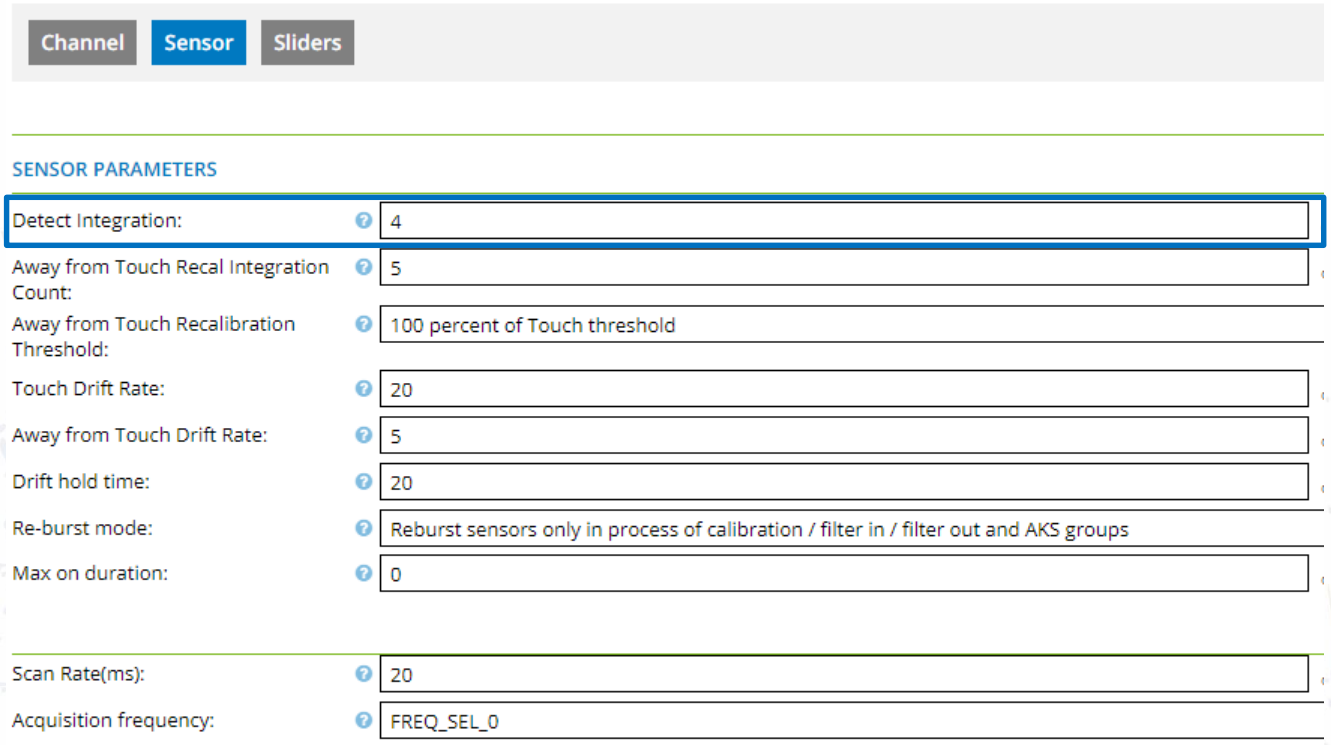


Digital Filtering amples	Gain	Analog Gain	Series Resistor	Addition...	PTC Pre-Scaler	Threshold	Hysterisis	Sensor AKS
	1	1	No Resistor	0	Divide by 4	20	25 percent ...	No AKS setting
	1	1	No Resistor	0	Divide by 4	20	25 percent ...	No AKS setting
	1	1	No Resistor	0	Divide by 4	20	25 percent ...	No AKS setting
	1	1	No Resistor	0	Divide by 4	20	25 percent ...	No AKS setting
	1	1	No Resistor	0	Divide by 4	20	25 percent ...	No AKS setting

调节参数的说明

传感器参数

- 检测积分（DI）



调节参数的说明
通道配置参数

1、Driver Sheild 2、取最大值

- 传感器AKS（相邻按键抑制）组

Channel | Sensor | S

CHANNEL PARAMETERS

Digital Filtering							
amples	Gain					Hysterisis	Sensor AKS
	1	1				25 percent ...	No AKS setting
	1	1				25 percent ...	No AKS setting
	1	1				25 percent ...	No AKS setting
	1	1				25 percent ...	No AKS setting
	1	1	No Resistor	0	Divide by 4	20	25 percent ...	No AKS setting

touch.h

```
#define DEF_NUM_SENSORS (5)

/* Defines Key Sensor setting
 * {Sensor Threshold, Sensor Hysterisis,
Sensor AKS}
 */
#define KEY_0_PARAMS
{
    20, HYST_25, NO_AKS_GROUP
}
```

阈值调节

阈值 = 60，增益A = 1，增益D = 1
覆盖层厚度 = 1 mm

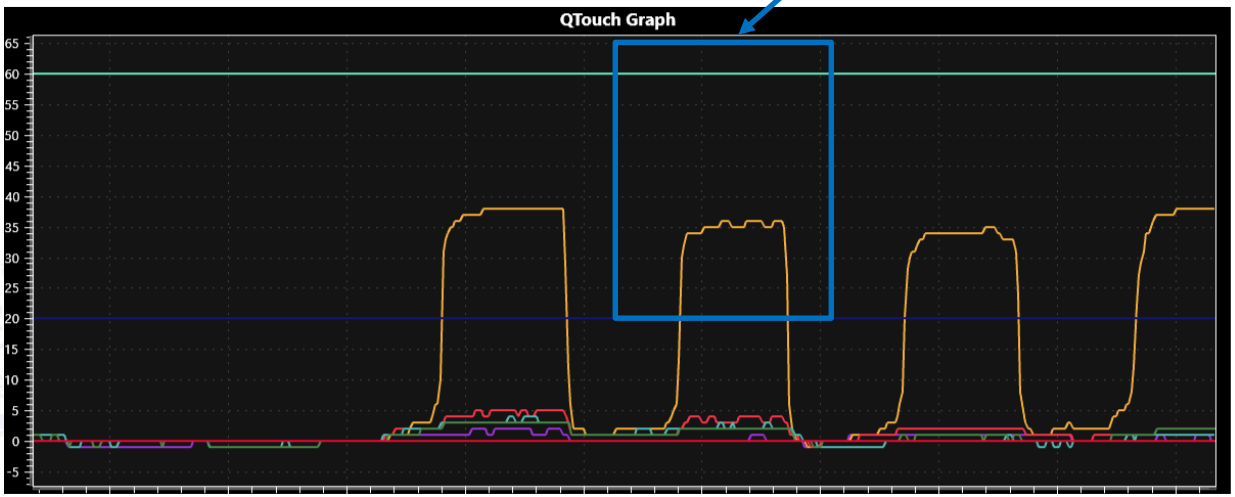
Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	1	60
1	Button 1	0	38	60
2	Slider 0[0]	0	1	60
3	Slider 0[1]	0	1	60
4	Slider 0[2]	0	2	60

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	0	0	20	0

显示灵敏度及其与阈值的差值



阈值调节

阈值 = 20，增益A = 1，增益D = 1
覆盖层厚度 = 1 mm

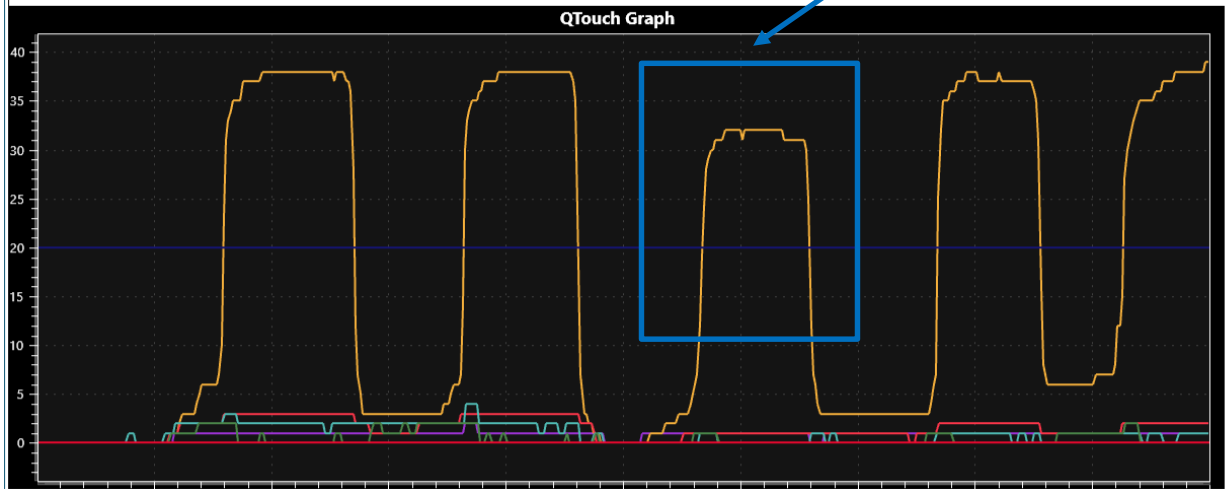
Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	1	20
1	Button 1	1	38	20
2	Slider 0[0]	0	2	20
3	Slider 0[1]	0	1	20
4	Slider 0[2]	0	0	20

Slider & Wheel Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	0	0	20	0

降低阈值可以识别触摸



观察灵敏度

阈值 = 30，增益A = 1，增益D = 1

覆盖层厚度 = 2 mm

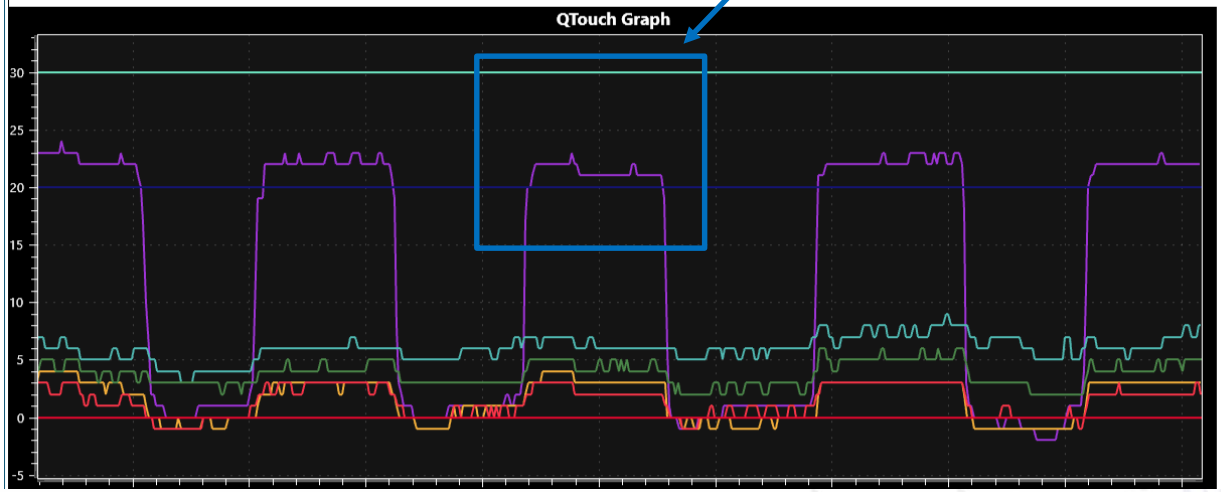
Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	0	22	30
1	Button 1	0	3	30
2	Slider 0[0]	0	2	30
3	Slider 0[1]	0	7	30
4	Slider 0[2]	0	6	30

Slider 0 Value Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	0	0	20	0

增加覆盖层厚度会降低灵敏度



阈值调节

阈值 = 20，增益A = 1，增益D = 1
覆盖层厚度 = 2 mm

Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	1	25	20
1	Button 1	0	6	20
2	Slider 0[0]	0	2	20
3	Slider 0[1]	0	5	20
4	Slider 0[2]	0	5	20

Slider Data

Sensor	State	SW Delta	SW Threshold	Position
Slider 0	0	0	20	0

信号远高于阈值
灵敏度足够

数字增益调节

阈值 = 40，增益A = 1，增益D = 2

覆盖层厚度 = 2 mm

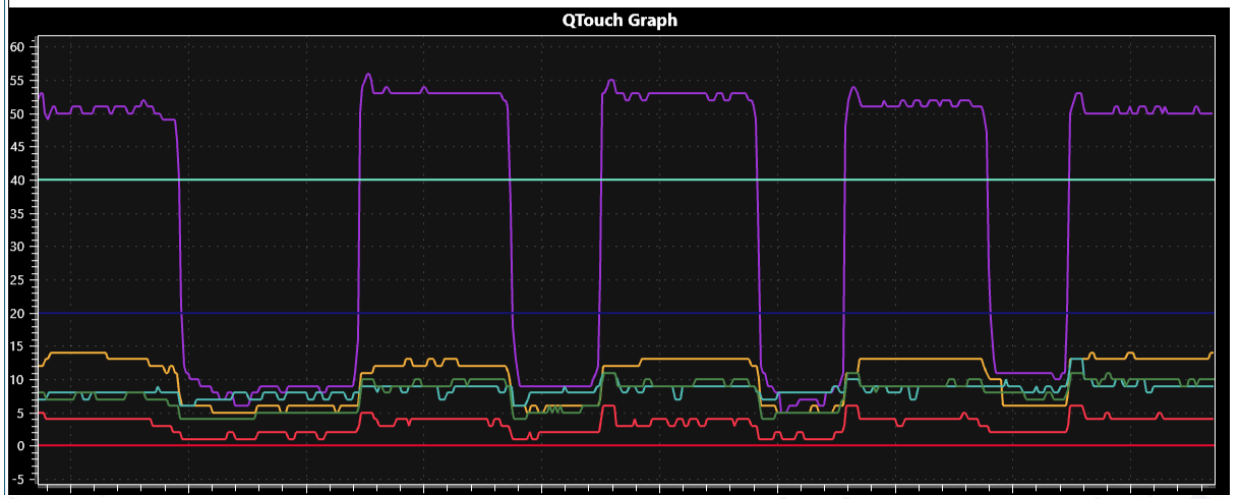
Button Data

Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	1	51	40
1	Button 1	0	13	40
2	Slider 0[0]	0	5	40
3	Slider 0[1]	0	9	40
4	Slider 0[2]	0	9	40

Slider 0 Status Data

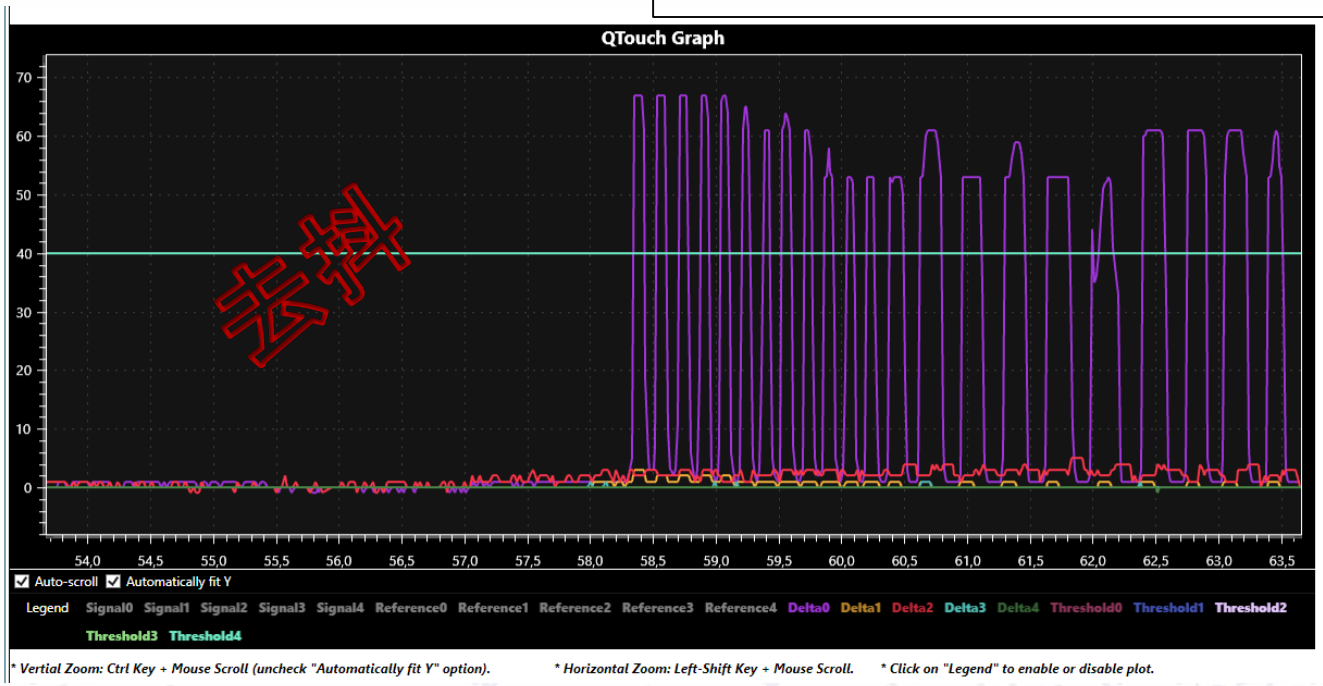
Sensor	State	SW Delta	SW Threshold	Position
Slider 0	0	0	20	0

可通过增大增益来提高灵敏度



检测积分的说明

阈值 = 40，增益A = 1，增益D = 2
覆盖层厚度 = 2 mm
检测积分DI = 1



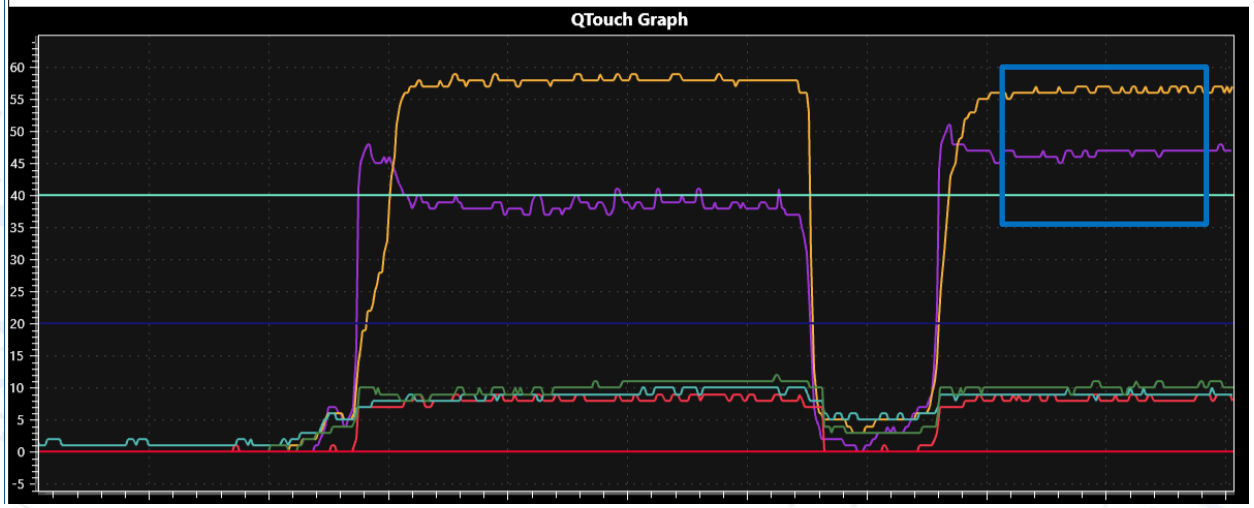
- 观察触摸响应速度的改善情况

相邻按键抑制

阈值 = 40，增益A = 1，增益D = 2
覆盖层厚度 = 2 mm
AKS_GROUP_1

- 为了阻止同时触摸检测，将按键分配给一个AKS组

Button Data				
Channel ID	Sensor Type	State	Delta	Threshold
0	Button 0	1	47	40
1	Button 1	0	57	40
2	Slider 0[0]	0	8	40
3	Slider 0[1]	0	9	40
4	Slider 0[2]	0	10	40



实验4总结

- 说明基本调节参数
- 对不同覆盖层进行高级调节
- 在**Data Visualizer**上可视化结果

课程安排

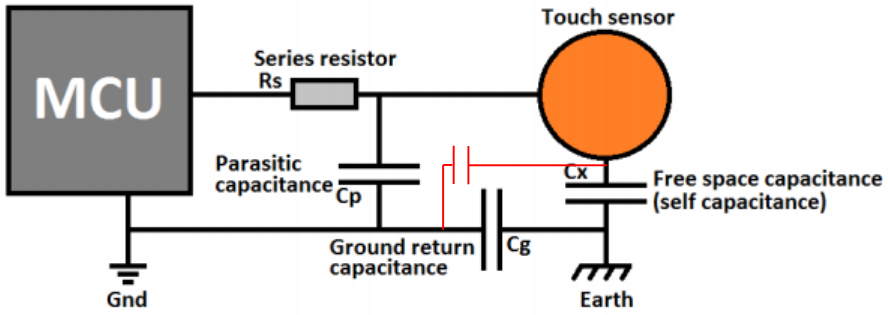
- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

实验5：
实现触摸库的耐水和抗噪声功能

驱动屏蔽概述

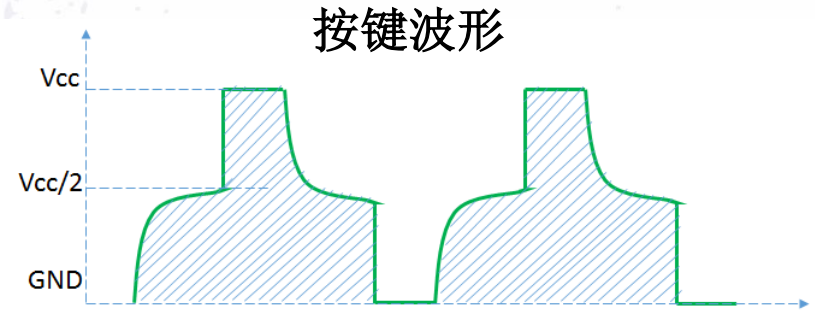
驱动屏蔽——需要什么？

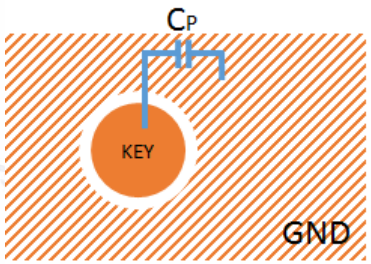
- 传感器周围的地具有良好的抗噪声性
- 传感器周围的地会降低防潮性
- 需要用某事物替换地，以实现相同的防噪声、防潮目的



驱动屏蔽——工作原理？

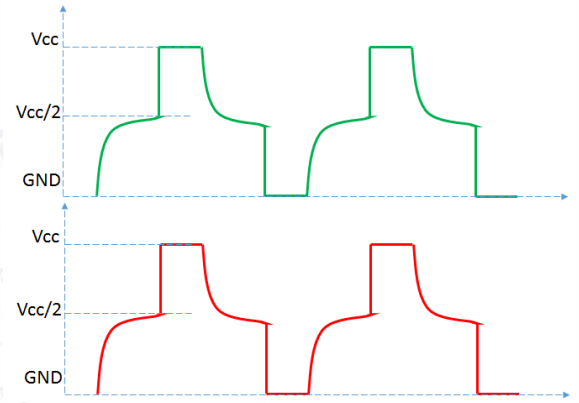
- 由于传感器电极和地面Cp之间形成电位差。
- Cp仅在阴影区域存在。*(q = C * V)*
- 当按键和GND之上有水时，Cp增加，因为水有助于耦合更多的场。





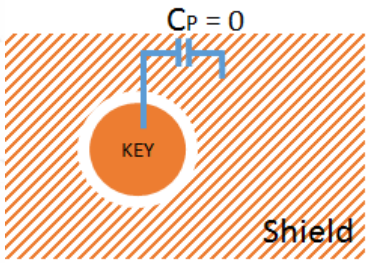
驱动屏蔽——工作原理？

- 驱动屏蔽以与传感器电极相同的电位驱动。
- Cp为0，因为没有电位差。
- 传感器和屏蔽层之间的水不会导致Cp发生任何变化，因为Cp不存在。



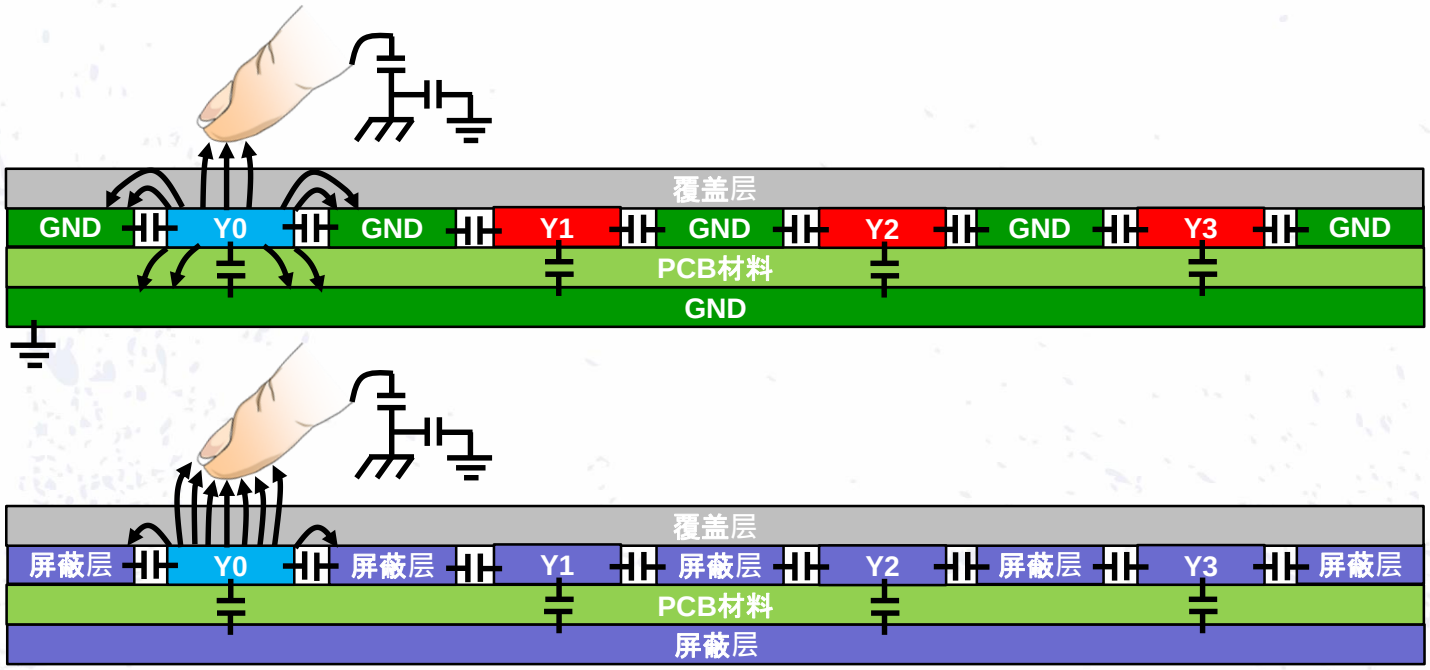
按键波形

屏蔽波形



驱动屏蔽——SNR得到改善

- 场不会被附近的地面偏转（返回路径）
- 场有效地到达表面——提高灵敏度

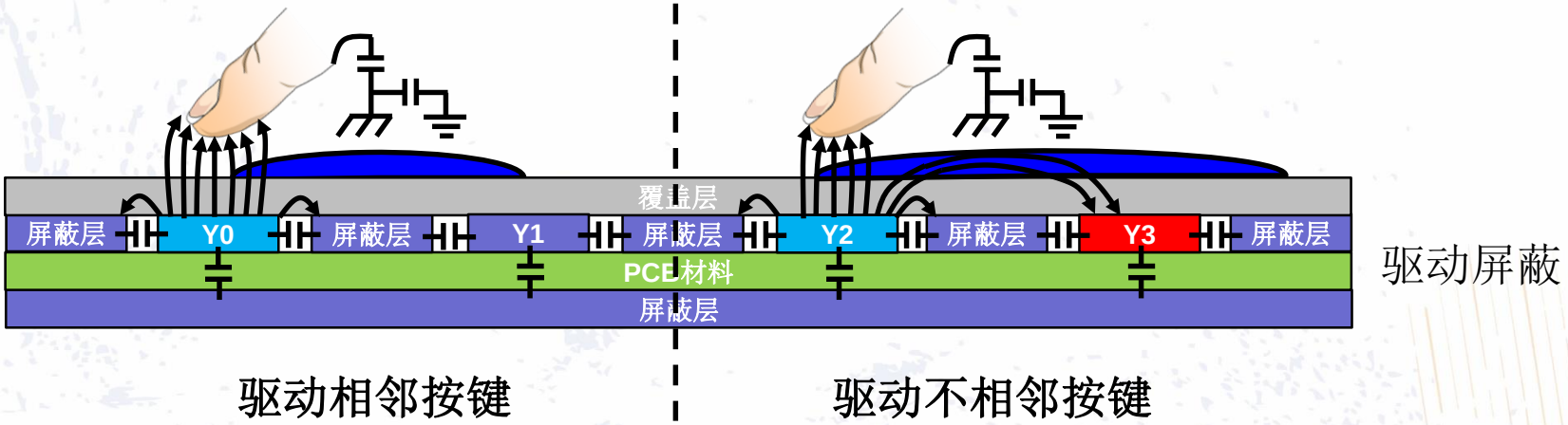


GND屏蔽

驱动屏蔽

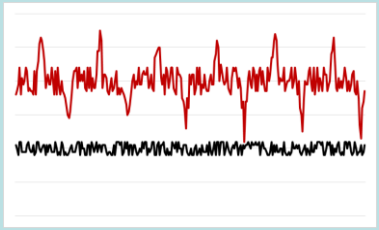
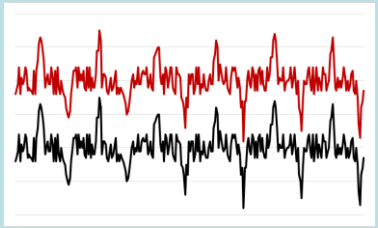
驱动屏蔽——潮湿时的性能

- 由于湿气不会在传感器和GND之间形成耦合，因此湿气不会引发虚假检测
- 相邻按键也与驱动屏蔽一起驱动，以改善防潮性能



噪声和调节对策概述

电磁噪声的类型

	差分	共模
		
辐射	• 荧光灯 • 靠近前面板的射频发射器	• 靠近电源线的射频发射器
传导	• 前面板上的LED • 应用中的电机	• 电源噪音 • 开关模式PSU • 与含噪声设备的有线通信

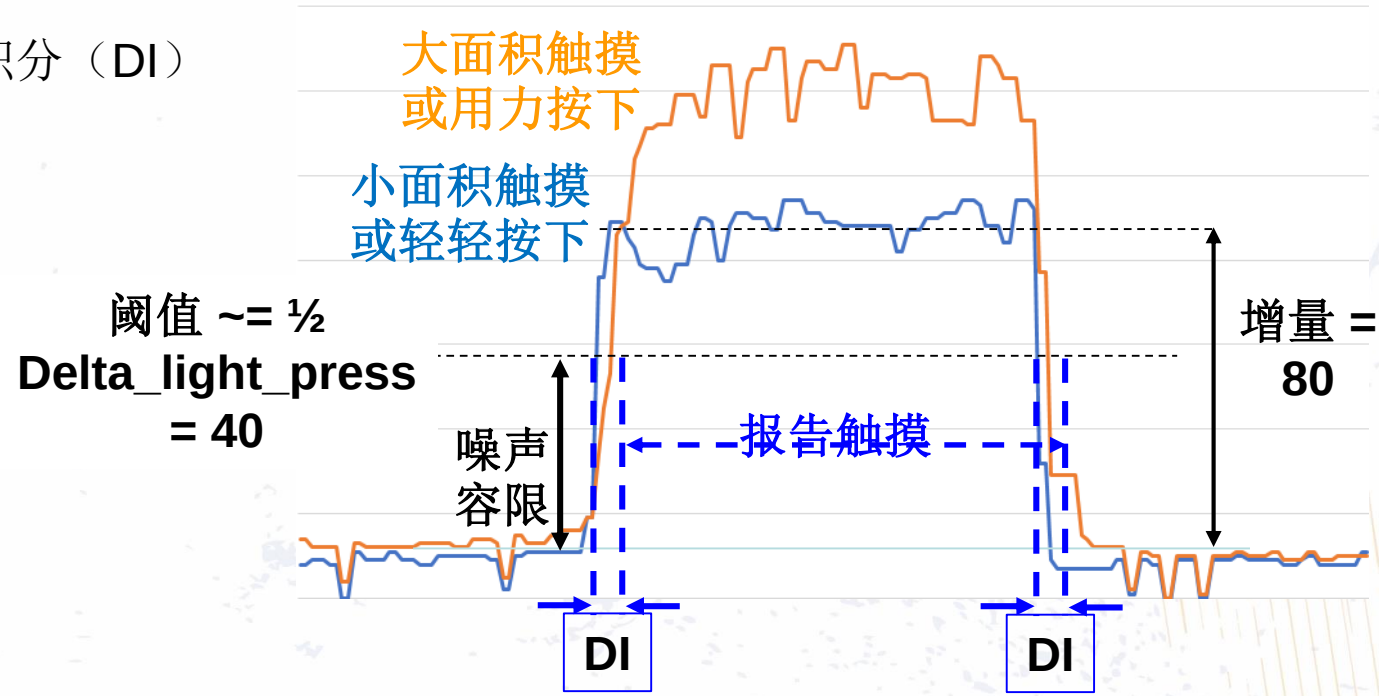
抗噪声对策

- 传感器调节不足会导致传感器性能不佳和抗噪声测试失败.....所以要了解应用中的噪声源！！！
- 解决方案不是单一的，而是通过多种技术在抑制噪声和实现可接受的系统性能之间进行折衷。

噪声对策	滤波效果最佳的噪声
串联电阻	传导噪声
过采样	辐射/传导噪声
自动过采样	传导噪声
跳频	辐射/传导噪声
检测阈值/滞后 滞后=积分	辐射/传导噪声
检测集成	辐射噪声

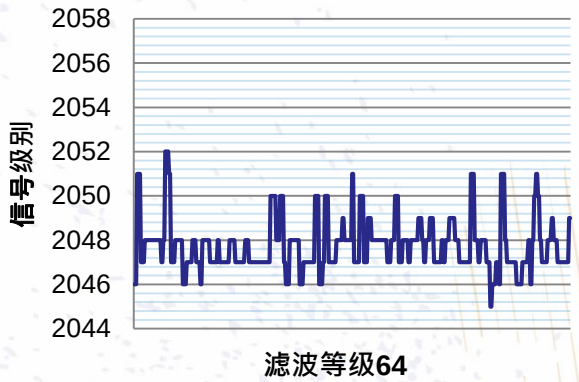
阈值和检测积分

- 基本调节参数
  - 阈值
  - 检测积分（DI）



滤波等级

- 定义采样的次数
- 滤波等级增加
  - 提高信噪比（SNR）
  - 延长采集时间
- 选择取决于噪声等级
- 可按每个通道配置
- 可配置范围为1至64
  （FILTER_LEVEL_1、FILTER_LEVEL_2、FILTER_LEVEL_4、FILTER_LEVEL_8、FILTER_LEVEL_16、FILTER_LEVEL_32和FILTER_LEVEL_64）



跳频

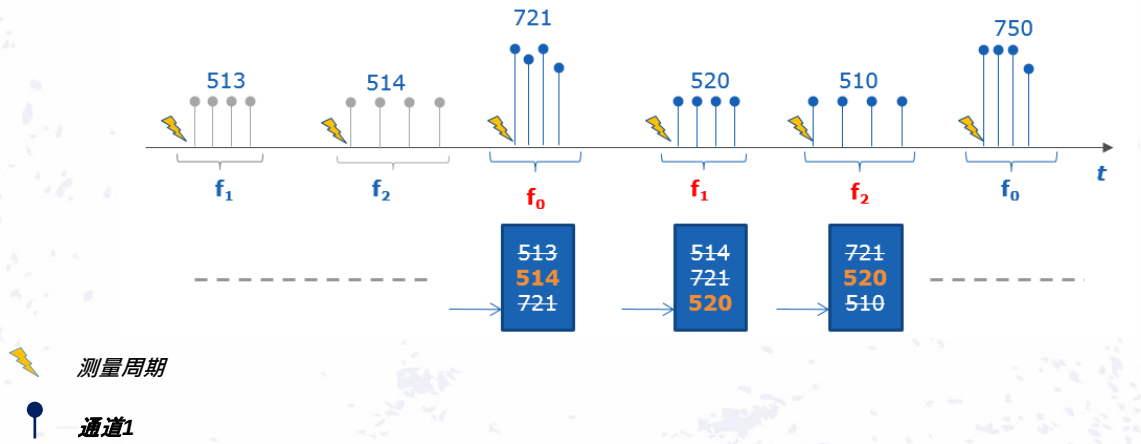
- 如果噪声频率接近采集频率，则触摸信号的噪声等级增加
- 必须改变采集频率以降低噪声
- 用户必须从包含16个频率的列表中选择一组频率（包含N个）
- 触摸库提供使用不同频率执行连续触摸测量的选项
- 对得到的N个信号值应用中值滤波器

跳频

- 示例配置：

  滤波等级 = 4，3个频率（N = 3）

- 噪声频率接近f0采集频率



跳频

- 要选择正确的频率，用户需要使用不同的频率组合进行迭代
- 选择频率可降低噪声等级
- 适合**固定**噪声频率
- 增加频率数量（N）
  - 提高抗噪声能力
  - 延长响应时间
  - 增大存储器（RAM）消耗

跳频自动调节

- 监视每个频率的噪声等级
- 在噪声等级满足以下条件时，动态更换最嘈杂的频率，
  - 一致
  - 高于阈值
- 适用于**可变**噪声频率

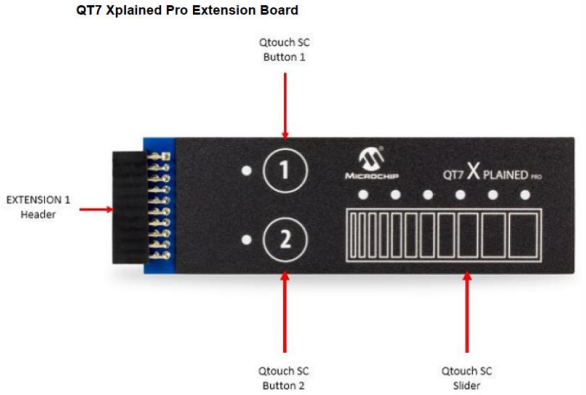
跳频自动调节

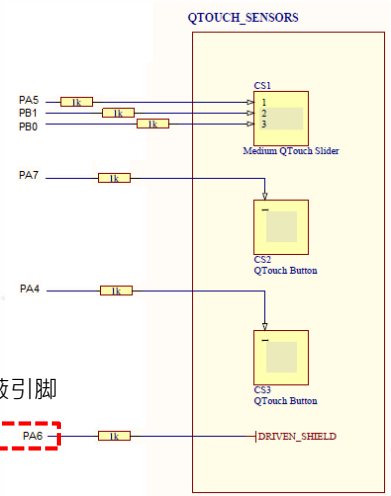
- 跳频与自动调节之间的差异

跳频	跳频自动调节
手动调节频率	自动调节频率
适用于频率固定的噪声	适用于频率变化的噪声
闪存和RAM存储器较小	闪存和RAM存储器较大
用户定义的频率的列表	使用的所有频率

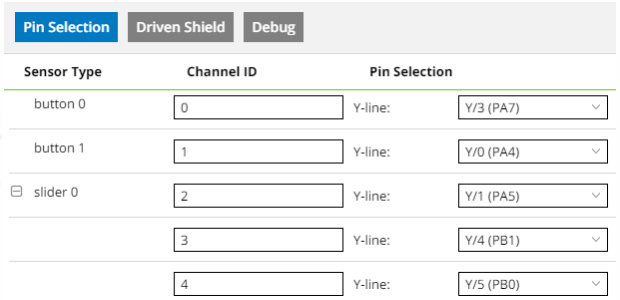
QT7触摸和驱动屏蔽

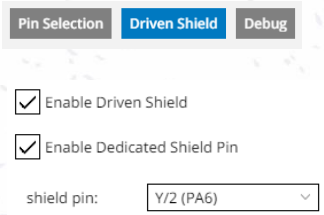
- QT7 Xplained板引脚配置





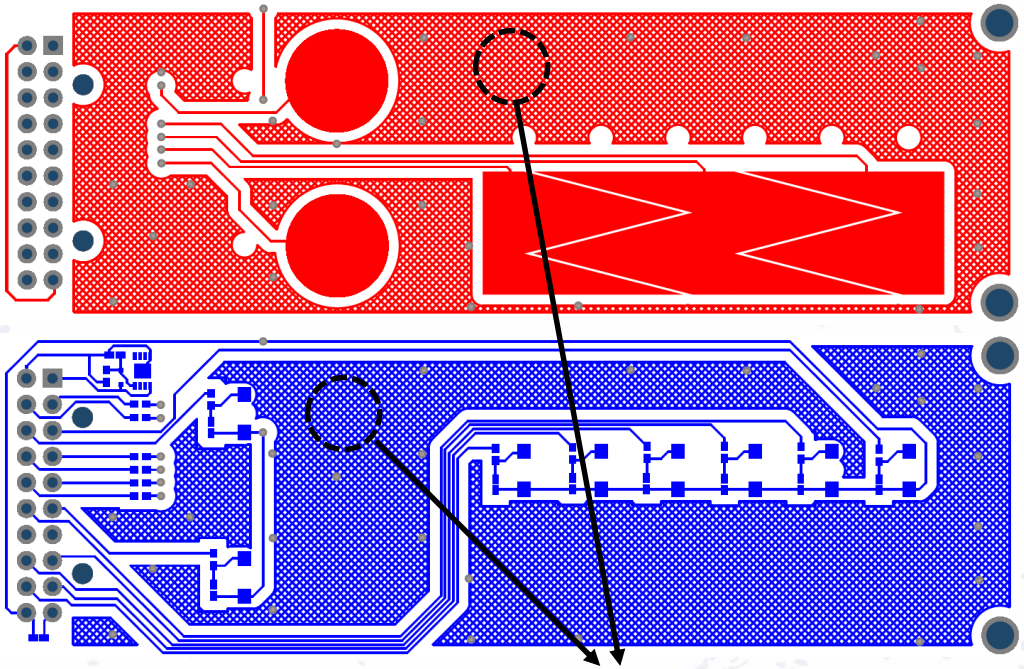
*PA6为驱动屏蔽引脚





QT7触摸和驱动屏蔽

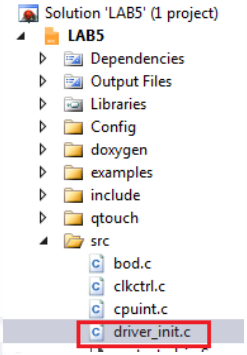
- QT7 Xplained板具有驱动屏蔽模式。



*驱动屏蔽模式（PA6）

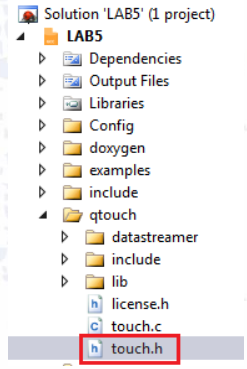
用源代码使能驱动屏蔽

- 通过修改源代码手动使能驱动屏蔽。



- 在“driver_init.c”中使用相应的代码进行以下设置，禁止PA6作为逻辑低电平输出

```
void system_init()
{
    .
    .
    /* PA6 is set as a log low output to emulate a grounded shield */
    /* Task 1a: Disable PA6 as a logic low output */
    PA6_set_dir(PORT_DIR_OUT);
    PA6_set_level(false);
    .
    .
}
```



- 在“touch.h”中用相应的常量替换NODE_SELFCAP

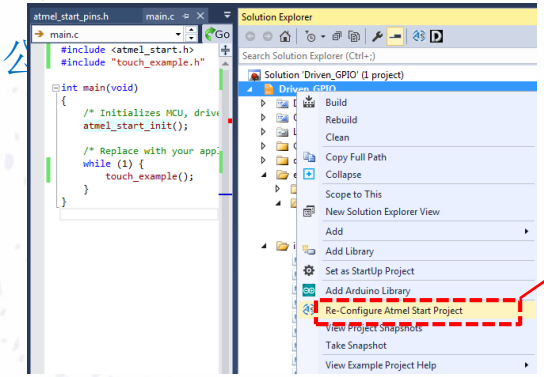
```
/* Task 1b: Enable Driven Shield */
#define DEF_SENSOR_TYPE NODE_SELFCAP
```

- 在“touch.h”中用相应的定义替换以下内容，使能驱动屏蔽

```
#define USE_DRIVEN_SHIELD   0
```

通过“Re-Configure..”（重新配置..）菜单使能驱动屏蔽

- 通过“Re-Configure Atmel Start Project”（重新配置Atmel Start项目）菜单使能和禁止驱动屏蔽

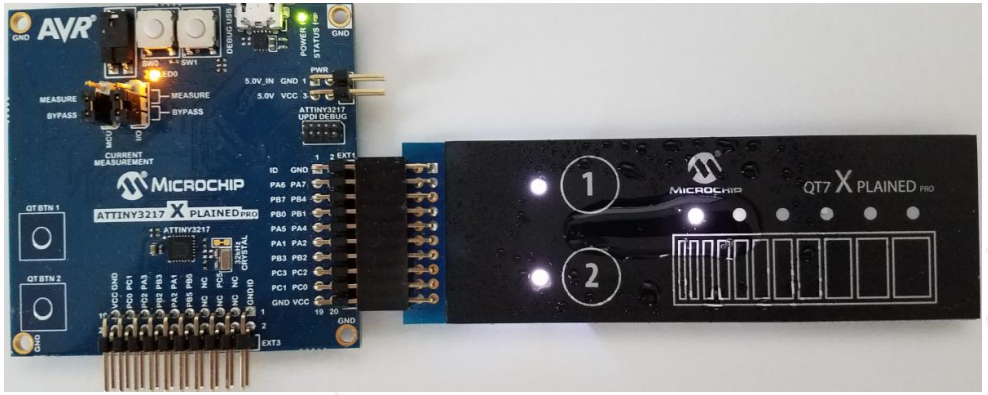


*您可以使用此菜单设置和更改触摸参数
*您也可以使能和禁止驱动屏蔽。

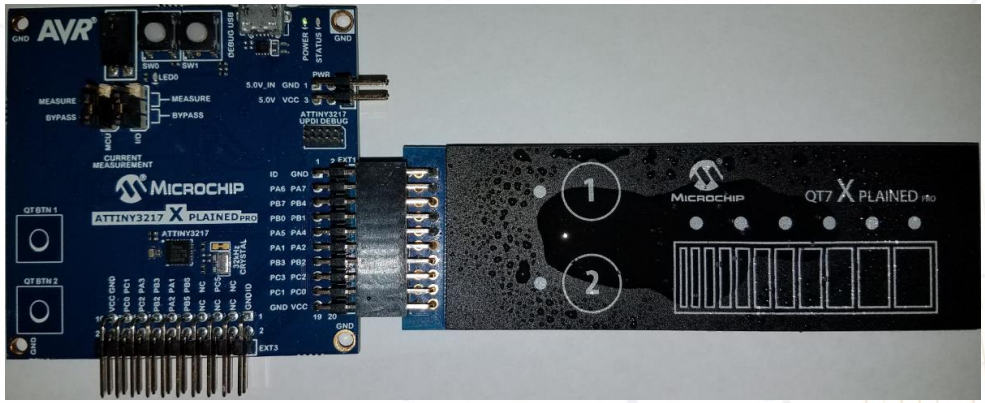
- 在“http://start.atmel.com/”上创建和修改涉及驱动屏蔽的项目公共版本（不适用于本实验）

通过驱动屏蔽实现防潮性

- 接地屏蔽

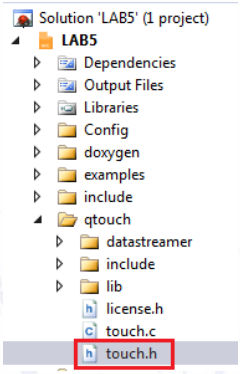


- 驱动屏蔽



源代码中的调节参数

- 您可以通过修改源代码手动更改触摸参数



- 更改“touch.h”中的阈值

```
/* Task 2a: Increase Threshold */
/* Defines Key Sensor setting
 * {Sensor Threshold, Sensor Hysterisis, Sensor AKS}
 */
#define KEY_0_PARAMS
    {
        20 HYST_25, NO_AKS_GROUP
    }
```

- 更改“touch.h”中的FILTER_LEVEL

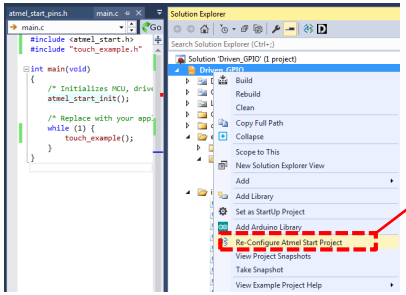
```
/* Defines node parameter setting
 * {Shield line, Y-line, Charge Share Delay, NODE_RSEL_PRSC(series resistor, prescaler), NODE_G(Analog Gain , Digital
 * Gain), filter level}
 */
#define NODE_0_PARAMS
    {
        Y(2) | Y(0) | Y(4) | Y(1) | Y(5), Y(3), 0, NODE_RSEL_PRSC(RSEL_VAL_0, PRSC_DIV_SEL_4),
            NODE_GAIN(GAIN_1, GAIN_1), FILTER_LEVEL_16
    }
```

- 在“touch.h”中用相应的定义更改DEF_FREQ_AUTOTUNE_ENABLE，使能频率自动调节

```
/* Task 4b: Enable FREQUENCY HOP AUTO TUNE */
/* Enable / Disable the frequency hop auto tune
 * Range: 0 / 1
 * Default value: 1
 */
#define DEF_FREQ_AUTOTUNE_ENABLE 0
```

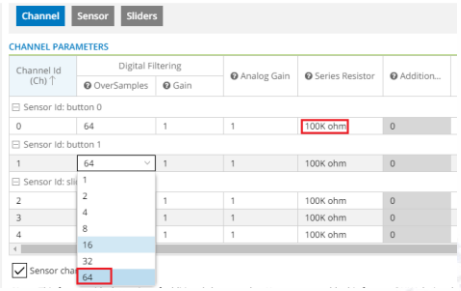
通过“Re-Configure..”菜单控制调节参数

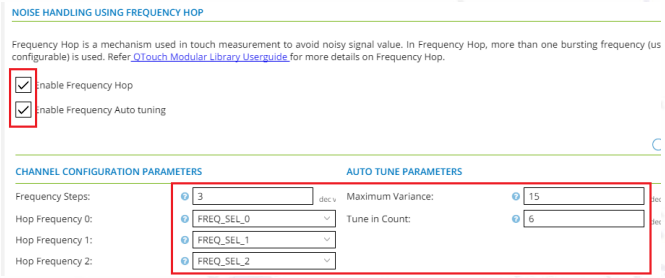
- 您可以通过“Re-Configure Atmel Start Project”菜单控制调节参数
  公共版本（不适用于本实验）



*您可以使用此菜单设置和更改触摸参数

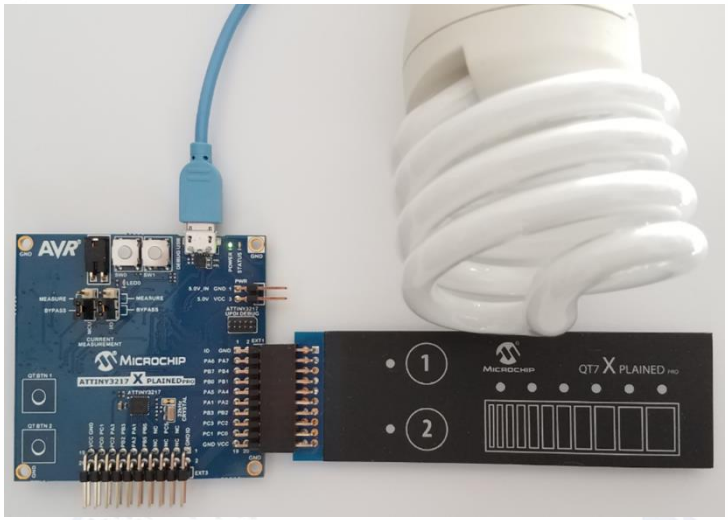
- 您可以更改过采样、串联电阻和跳频，如下所示。

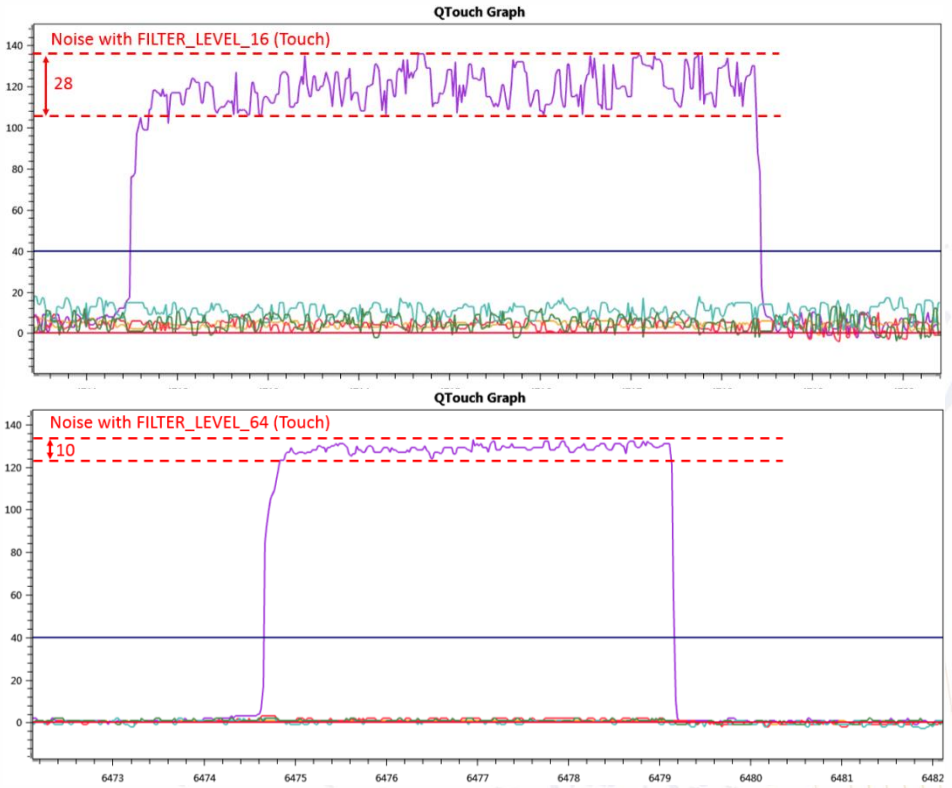




- 在“http://start.atmel.com/”上创建和修改项目
  公共版本（不适用于本实验）

通过调节参数提高抗噪性





实验5总结

- 实现耐水性
- 提高对荧光灯的抗噪性
- 使用**Data Visualizer**

MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

随堂问题！

- **mTouch®和QTouch®库有什么区别？**
  - 技巧问题：使用哪款产品是由您对哪款MCU最熟悉、最有经验决定的。
  - 它们在1D和2D触摸、耐水性和抗噪声性方面性能相当。
  - 不同的代码设计器工具可产生性能相当的代码。

本课程使用的开发工具

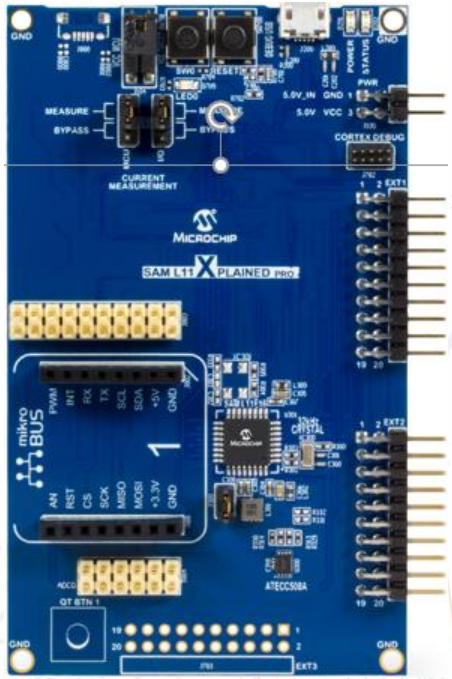
- 低成本**mTouch®**评估工具包[DM160227]
- **PICkit™ 3**编程器[PG164130]
- **ATtiny3217 Xplained Pro** [ATTINY3217-XPRO]
- **ATQT7 Xplained Pro** [ATQT7-XPRO]

全新开发工具！

- **ATSAML10/L11 Xplained Pro**

[ATSAML10-XPRO]和[ATSAML11-XPRO]

- 32 MHz Arm Cortex® M23内核
- 最高64 KB闪存/16 KB SRAM
- picoPower®技术
- 增强型PTC（20 SC，10x10 MC）
- Arm® TrustZone®技术



全新开发工具！

- **PICkit™ 4在线调试器[PG164140]**
- 在器件允许的范围内尽可能快地编程
- 目标电压为1.20V至5.5V
- 通过流数据网关支持4线JTAG和串行线调试
- 向后兼容使用2线JTAG和ICSP的系统
- 使用SD卡的Programmer-to-Go（PTG）

MICROCHIP
MASTERs 2018

其他培训链接

mTouch®电容式传感库

- 触摸传感
- **MCC** mTouch®电容式传感库
- 低成本**mTouch®**评估工具包示例

Atmel QTouch®电容式传感库

- Atmel **START** QTouch®模块化库
- Atmel **Studio**
- 使用**Data Visualizer**可视化触摸调试数据

mTouch®库
应用笔记

- **CVD相关的应用笔记：**
  - AN1478：《mTouch触摸传感解决方案采集方法电容分压器》
  - AN1334：Techniques for Robust Touch Sensing Design
  - AN1492：Microchip Capacitive Proximity Design Guide
  - AN1626：Implementing Metal Over Capacitive Touch Sensors
  - AN2098：Water Resistance

QTouch®库
应用笔记

- **PTC相关的应用笔记：**
  - **AT09363**：PTC Robustness Design Guide
  - **AT12405**：Low Power Sensor Design with PTC
  - **QTAN0079**：Buttons, Sliders and Wheels Sensor Design Guideline

MICROCHIP MASTERs 2018

课程安排

- 什么是电容式传感——理论
- 电容式传感电路模型基础
- 抗噪声基础与设计
- 防水基础与设计
- 触摸低功耗基础与设计
- **MCC和mTouch®库——实验1**
- **START和QTouch®模块化库——实验2**
- **Data Visualizer入门——实验3**
- **使用Data Visualizer和触摸库参数来调节触摸系统——实验4**
- **实现触摸库的耐水和抗噪声功能——附加实验5**
- **mTouch®或QTouch®库——使用哪一个？**
- 总结

MICROCHIP MASTERs 2018

课程总结

- 今天介绍了：
  - 电容式传感电路模型
    - 自电容
    - 互电容
  - 为什么电磁噪声和水给电容式触摸设计带来了挑战
  - 有哪些工具和技术可以应对这些挑战
  - 如何设计低功耗触摸

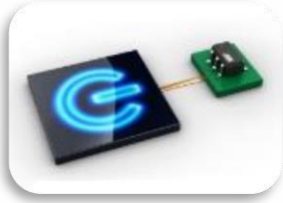
总结

今天您...

- 使用了**mTouch®**和**QTouch®**库中的触摸**API**和相关参数在基于**MCU**的设计中实现稳健的触摸功能。
- 使用了**MCC**和**START**生成项目，以使用**mTouch®**和**QTouch®**模块化库实现触摸按钮。
- 使用了**Data Visualizer**工具为**mTouch®**和**QTouch®**设计调试和优化触摸参数。
- 学习了用于实现防水和防噪声的稳健触摸设计的高级调节设置。

MASTERs期间的其他触摸/手势课程

C19L07 TNG14　使用Microchip 1D/2D/3D触摸技术设计独一无二的用户界面！



谢谢！

“对于客户而言，
界面就是产品”

——人机界面专家兼Apple Macintosh项目发起人Jef Raskin

法律声明

软件：

Microchip软件仅允许用于Microchip产品。此外，Microchip软件的使用受软件附带的版权声明、免责声明以及任何授权许可的限制，无论这些内容是在安装各个程序时阐明还是在头文件或文本文件中公告。

尽管有上述限制，但Microchip和第三方提供的软件的某些组件仍可能被“开源”软件许可覆盖，其中包括要求分发者提供软件源代码的许可。在开源软件许可要求的范围内，许可条款将起主导作用。

注意事项和免责声明：

这些材料和随附信息（例如，包括任何软件以及对第三方公司和第三方网站的引用）仅供参考，并且按“现状”提供。Microchip对第三方公司做出的声明或第三方可能提供的材料或信息不承担任何责任。

MICROCHIP不承担任何形式的保证，无论是明示的、暗示的或法定的，包括有关无侵权性、适销性和特定用途的暗示保证。在任何情况下，对于与MICROCHIP或其他第三方提供的材料或随附信息有关的任何直接或间接的、特殊的、惩罚性的、偶然的或间接的损失、损害或任何类型的开销，MICROCHIP概不承担任何责任，即使MICROCHIP已被告知可能发生损害或损害可以预见。请注意，使用此处所述的知识产权时可能需要第三方许可。

商标：

Microchip的名称和徽标组合、Microchip徽标、AnyRate、AVR、AVR徽标、AVR Freaks、BitCloud、chipKIT、chipKIT徽标、CryptoMemory、CryptoRF、dsPIC、FlashFlex、flexPWR、Heldo、JukeBlox、KeeLoq、Kleer、LANCheck、LINK MD、maXStylus、maXTouch、MediaLB、megaAVR、MOST、MOST徽标、MPLAB、OptoLyzer、PIC、picoPower、PICSTART、PIC32徽标、Prochip Designer、QTouch、SAM-BA、SpyNIC、SST、SST徽标、SuperFlash、tinyAVR、UNI/O及XMEGA均为Microchip Technology Inc.在美国和其他国家或地区的注册商标。

ClockWorks、The Embedded Control Solutions Company、EtherSynch、Hyper Speed Control、HyperLight Load、IntelliMOS、mTouch、Precision Edge和Quiet-Wire均为Microchip Technology Inc.在美国的注册商标。

Adjacent Key Suppression、AKS、Analog-for-the-Digital Age、Any Capacitor、AnyIn、AnyOut、BodyCom、CodeGuard、CryptoAuthentication、CryptoAutomotive、CryptoCompanion、CryptoController、dsPICDEM、dsPICDEM.net、Dynamic Average Matching、DAM、ECAN、EtherGREEN、In-Circuit Serial Programming、ICSP、INICnet、Inter-Chip Connectivity、JitterBlocker、KleerNet、KleerNet徽标、memBrain、Mindi、MiWi、motorBench、MPASM、MPF、MPLAB Certified徽标、MPLIB、MPLINK、MultiTRAK、NetDetach、Omniscient Code Generation、PICDEM、PICDEM.net、PICkit、PICtail、PowerSmart、PureSilicon、QMatrix、REAL ICE、Ripple Blocker、SAM-ICE、Serial Quad I/O、SMART-I.S.、SQI、SuperSwitcher、SuperSwitcher II、Total Endurance、TSHARC、USBCheck、VariSense、ViewSpan、WiperLock、Wireless DNA和ZENA均为Microchip Technology Inc.在美国和其他国家或地区的商标。

SQTP为Microchip Technology Inc.在美国的服务标记。

Silicon Storage Technology为Microchip Technology Inc.在除美国外的国家或地区的注册商标。

GestIC为Microchip Technology Inc.的子公司Microchip Technology Germany II GmbH & Co. & KG在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。