采用 MSP430 Δ-Σ ADC 外设实现高精度测量

Vincent Chan

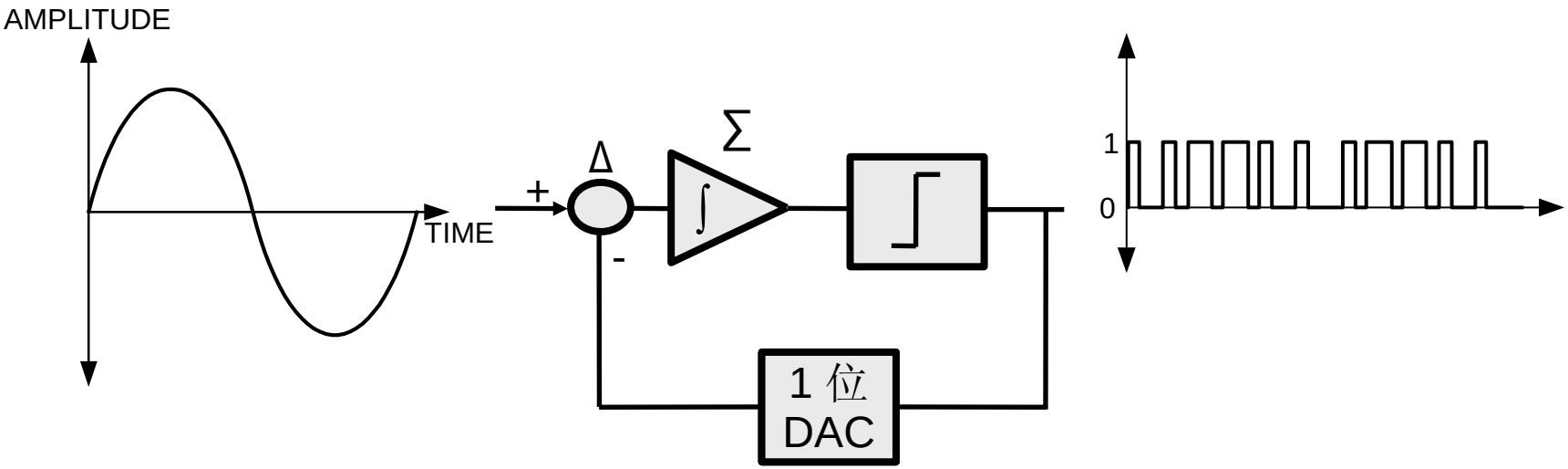
德州仪器亚洲 **MSP430** 市场经理

vince-chan@ti.com

会议议程

- Δ-Σ 的原理与优势
- 了解 **SD16_A**
- 选择 集成了 **ADC** 的 **MSP430**
- 实验练习：**SD16_A** 实际操作

Δ – Σ 原理



简单模拟：**1** 位 **ADC**

- 积分 **(Σ)** 与微分 **(Δ)** 级

复杂的数字：滤波器级

- 过采样输入
- 抽取滤波输出

Δ－Σ 与逐次逼近的对比

- Δ－Σ 的分辨率通常较高
- **1 ksps** Δ－Σ 与 **100 ksps SAR** 对比
- 每次采样的模拟步长冗余是不同的

  SAR 固有

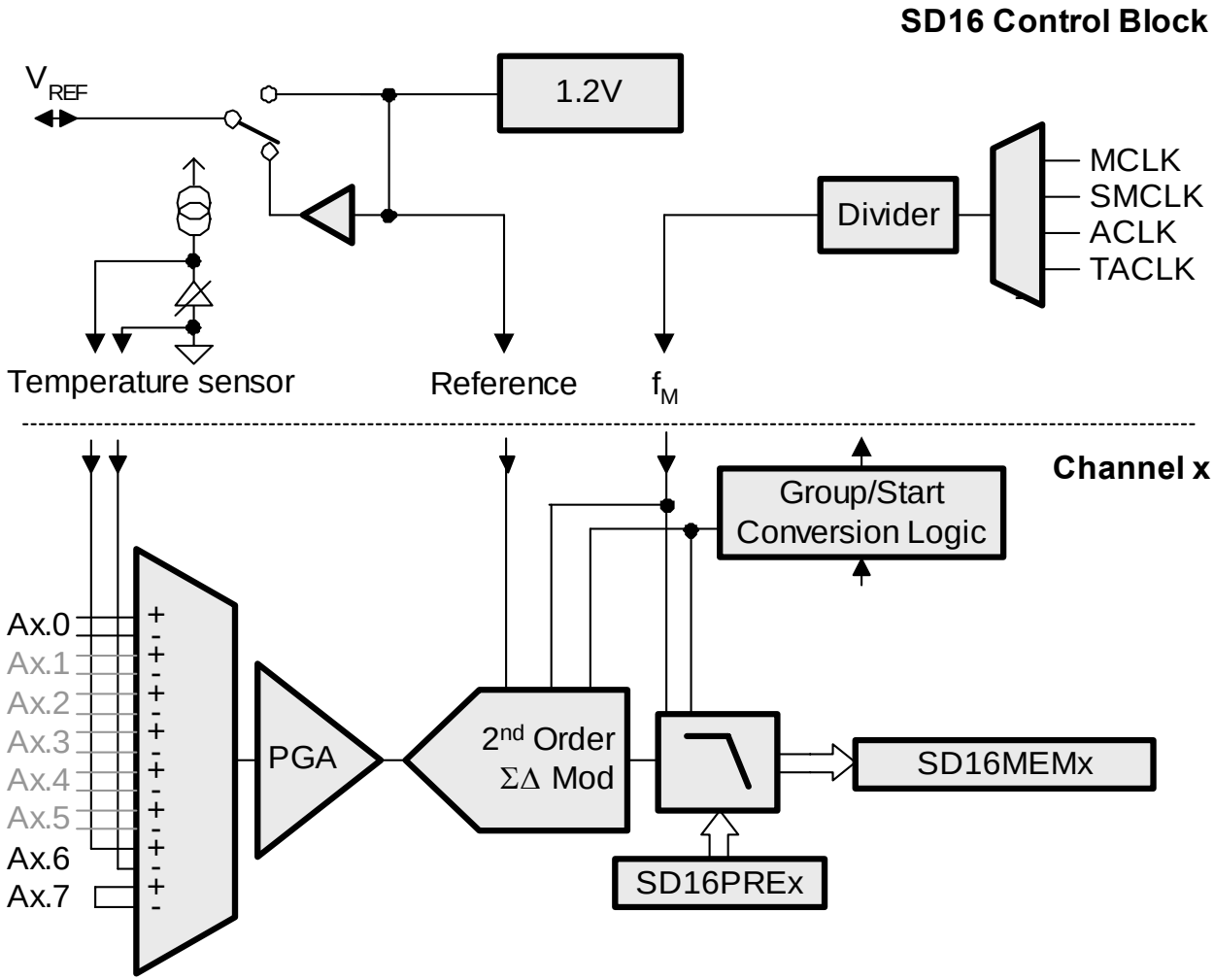
  输入端的步长变化必须通过 Δ－Σ 滤波器级循环
- Δ－Σ 架构的数字化程度达 **90%**：使集成更简便

SD16 概览

- **‘F42x 与 ‘FE42x**
- 多个通道
- 每通道一个外部输入
- 高达 **256** 的 **OSR**

过采样率

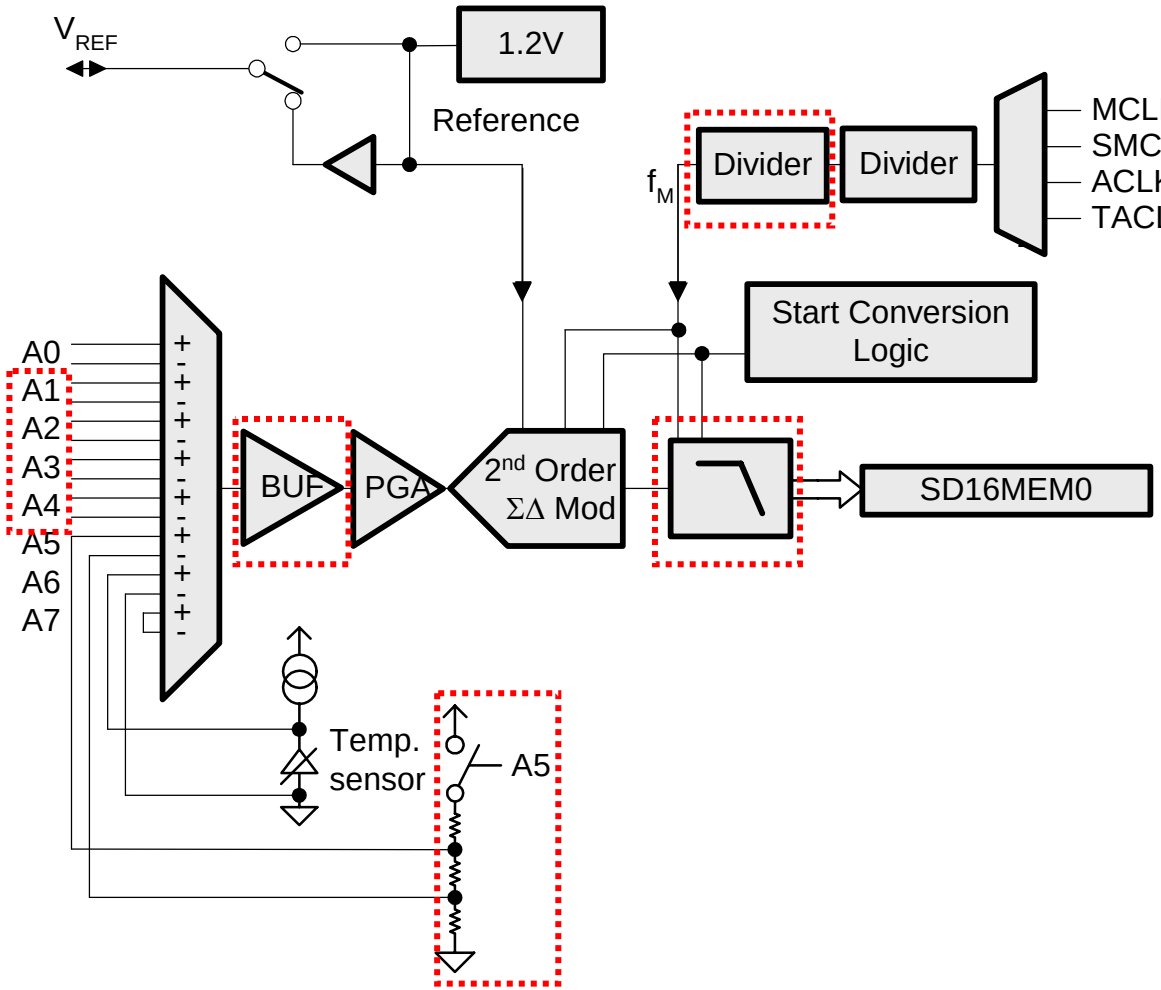
- **1MHz f_M**



SD16_A 概览

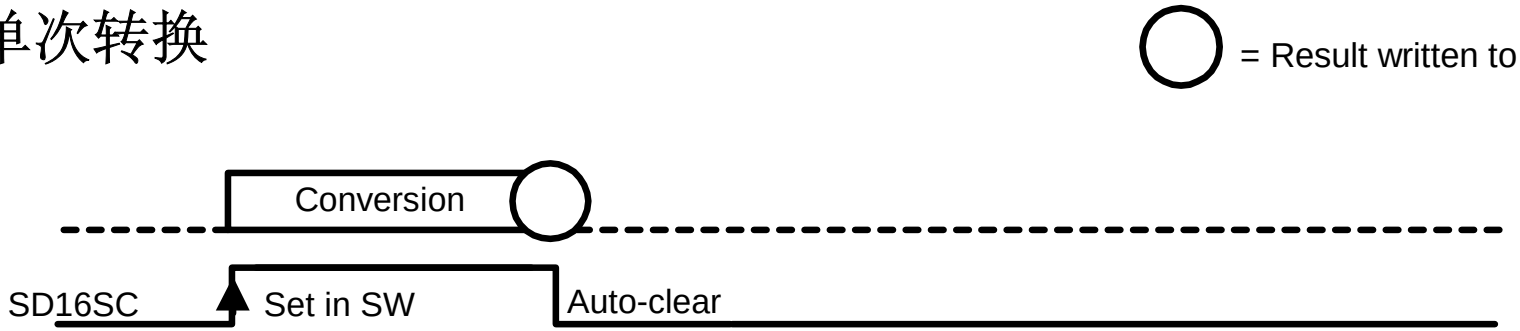
- **‘F42x0 与 ‘F20x3**
- 单通道
- 多个输入对
- 输入缓冲器
- **AV_{CC}** 测量
- **30kHz 至 1.1MHz**
- **f_M** 分频器
- 高达 **1024 OSR**

过采样率

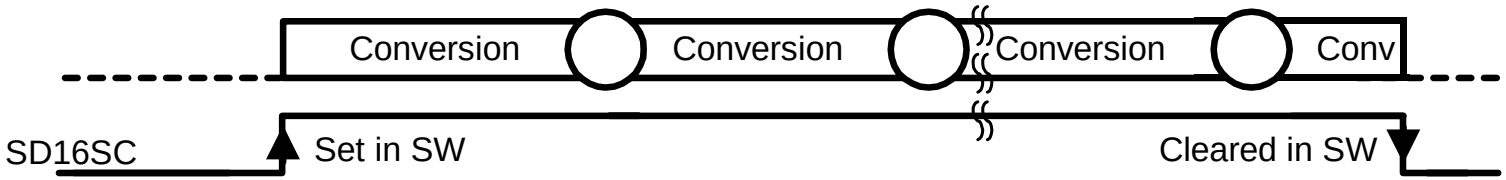


转换模式

- 单次转换



- 连续转换

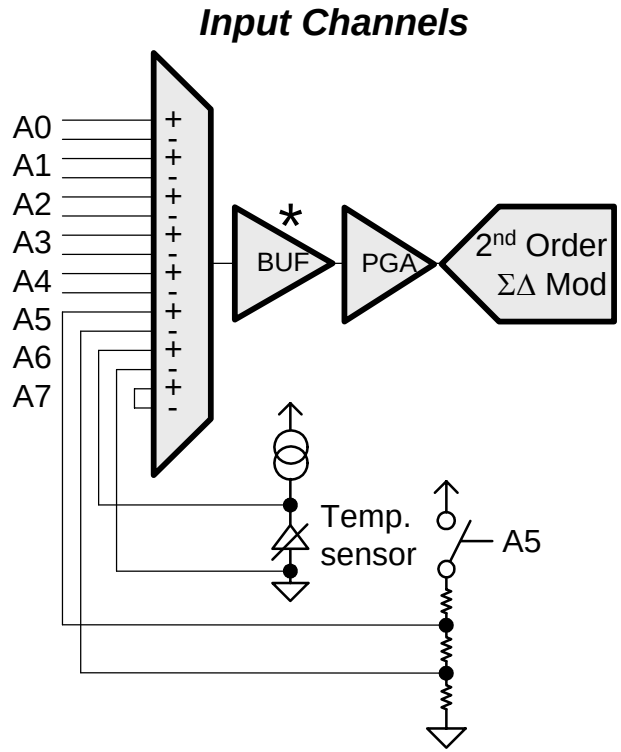


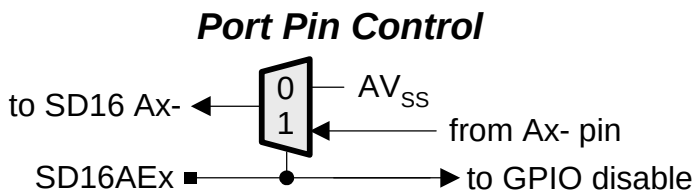
清除 SD16SC 可立即停止转换：

SD16MEMx 中的值会变化，应在停止转换前读取

SD16_A 输入设计

- 四个外部输入对
- 全差分输入
- 内部通道：
  温度
  AV_{CC} / 11
  输入端短接
- 可选电流与高速输入缓冲器
- **PGA：1、2、 4、 8、16 与 32x**
- SD16AEx 位，用于使能内部 A_{IN} 连接至 AV_{SS}
- ***‘F20x3 器件不带缓冲器**



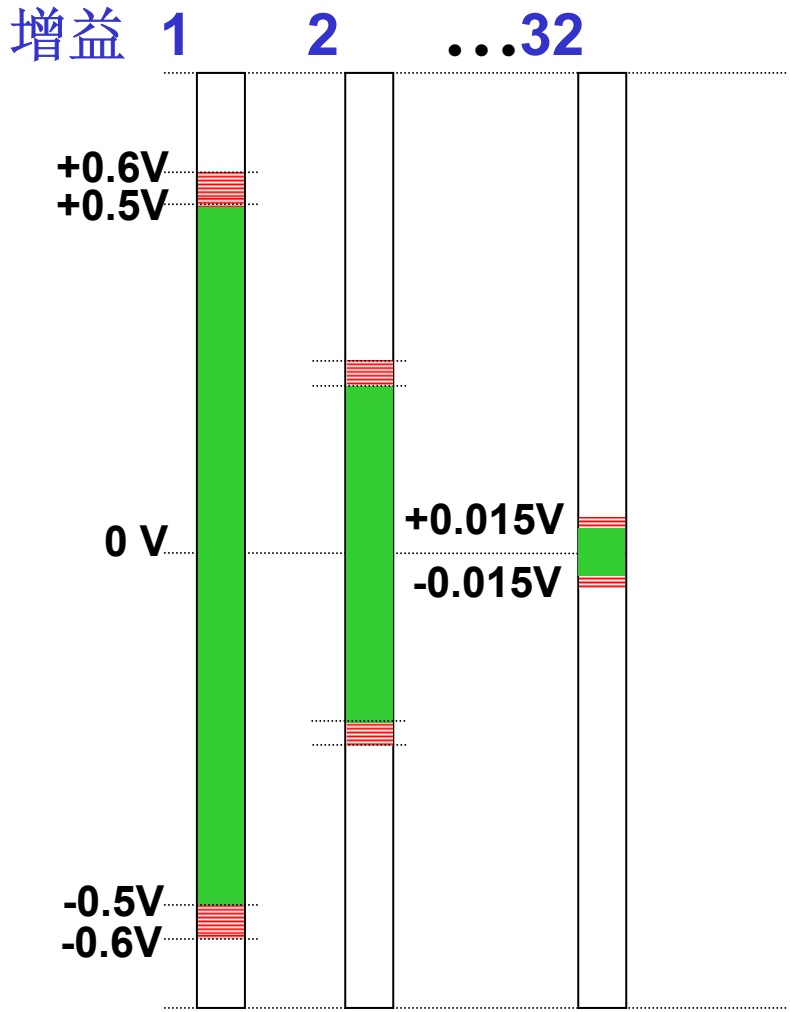


模拟输入范围

- 什么是 V_{REF}?
- **PGA** 如何设置?

- 适用于所有输入和模式

$$V_{FSR} = \frac{V_{ref} / 2}{GAIN_{PGA}}$$

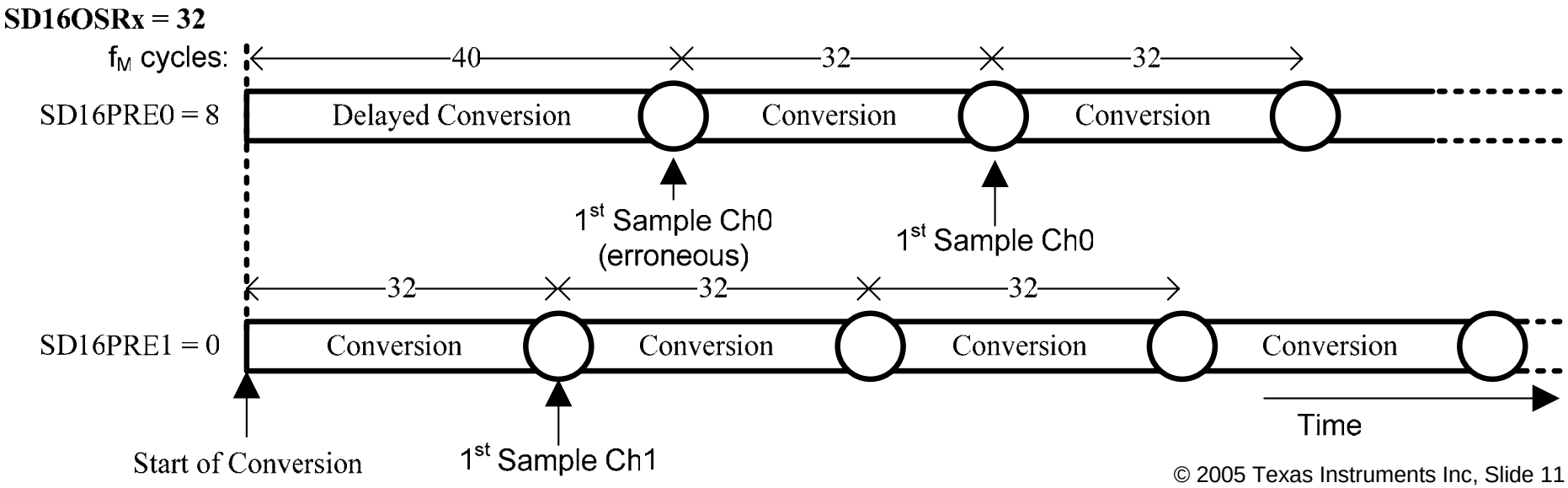


输入选择与通道选择的对比

- **SD16_A**：单通道，每通道 **4** 个外部输入
  MSP430F42x0 与 MSP430F20x3
- **SD16**：**3** 个通道，每通道 **1** 个外部输入
  MSP430FE42x 与 MSP430F42x
- 多通道可并行独立工作
- 多输入进行复用后连接到单通道；所以多输入必须顺序选择，顺序采样

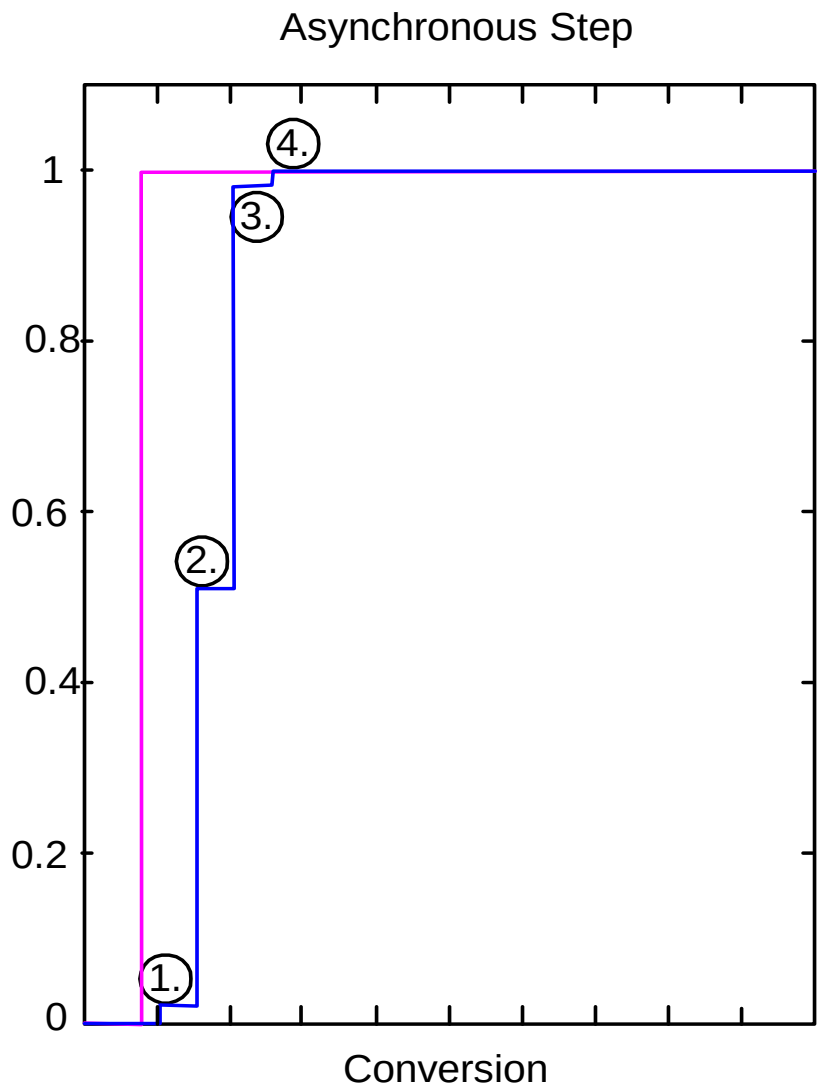
采用预加载：仅适用于多通道

- 抽样滤波器的偏差对应多少个 f_M 时钟周期
- 通道间交错的转换结果
- 用于引入转换流程中的相位延迟，以实现信号补偿
- 举例：电表 **I** 与 **V** 相位补偿的对比

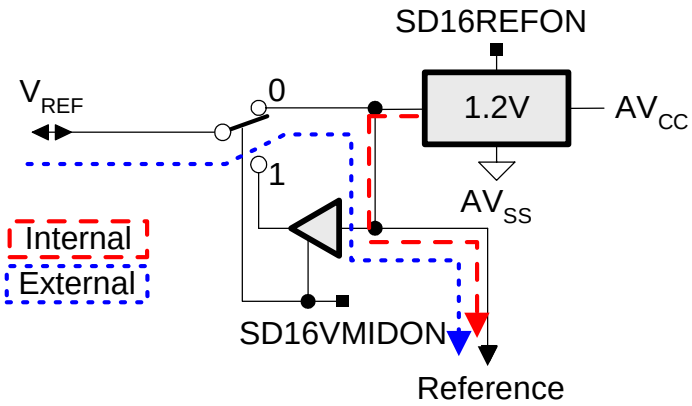


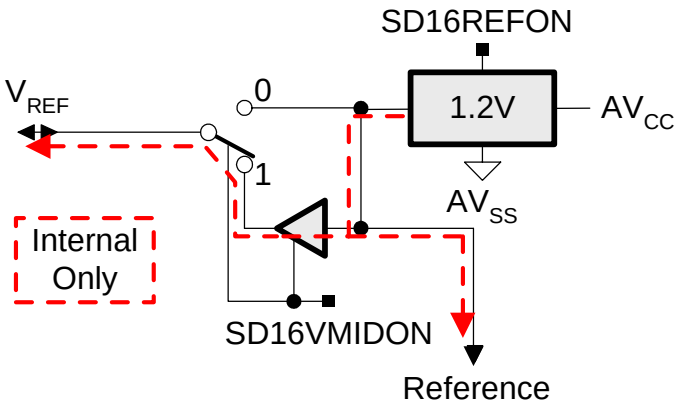
输入步长响应

- 对多路复用切换很重要
- 抽样滤波器必须循查出差值
- **SD16INTDLYx** 设置第一次转换中断的自动稳定时间
- **f_M = 1.048MHz；OSR = 256**
  f_{SAMPLE} = 4096 ksps ->
  $t_{SETTLE(MAX)}$ ~ 732usec



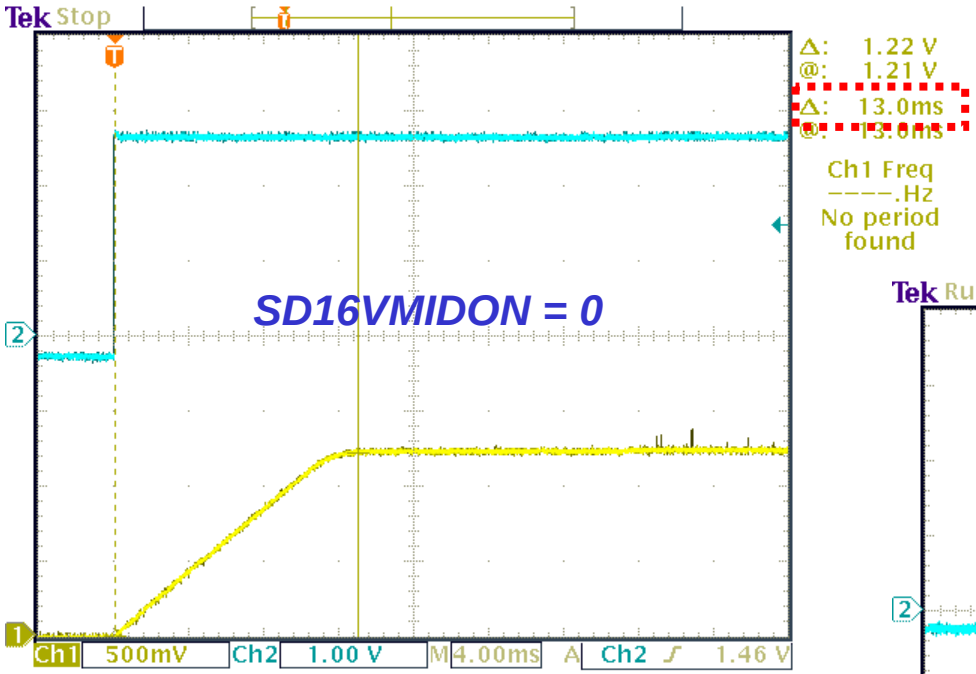
内部参考源

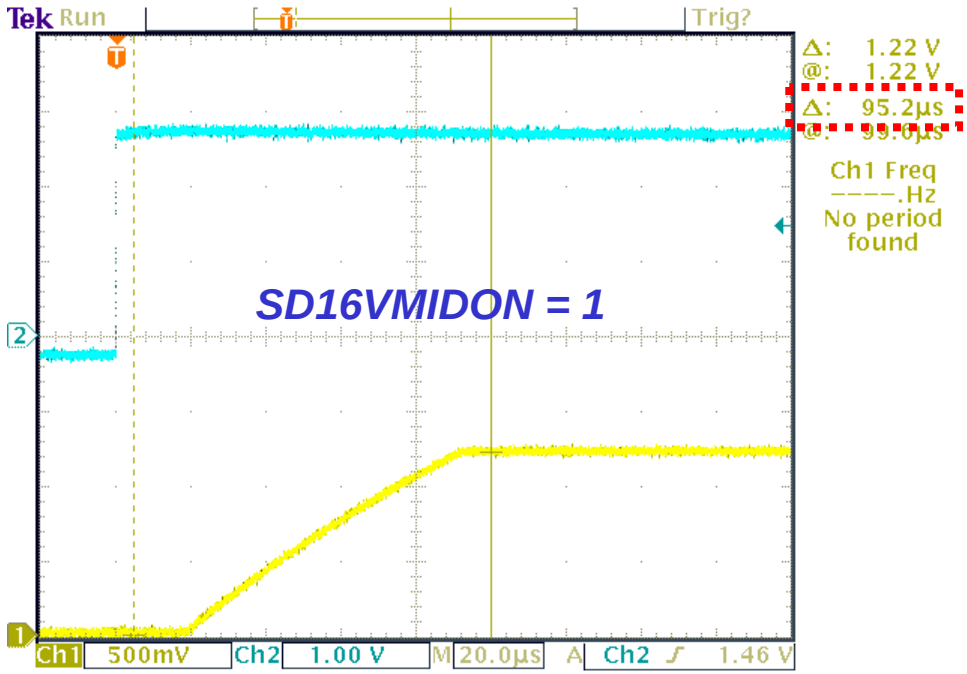




- 内部 **1.2V** 参考源
- **20ppm** 温漂系数
- **V_{REF}** 选项：
  外部参考源： SD16REFON = 0，SD16VMIDON = 0
  内部参考源： SD16REFON = 1，SD16VMIDON = 0
  内部参考源带缓冲器输出：SD16REFON = 1，SD16VMIDON = 1
- 针对温度 **(A6)**：采用内部参考源

内部参考源稳定时间





- C_{VREF} = 470nF
- 缓冲器打开比关闭时, 参考源稳定时间快 **100** 倍以上
- 参考源稳定后禁用缓冲器

调制器的时钟选择

- 用 **D4270** 进行 f_M 补偿：
  （连续转换模式，启用内部参考和缓冲器）

ACLK = 32.768kHz, 256 OSR, SD16INTDLY_0

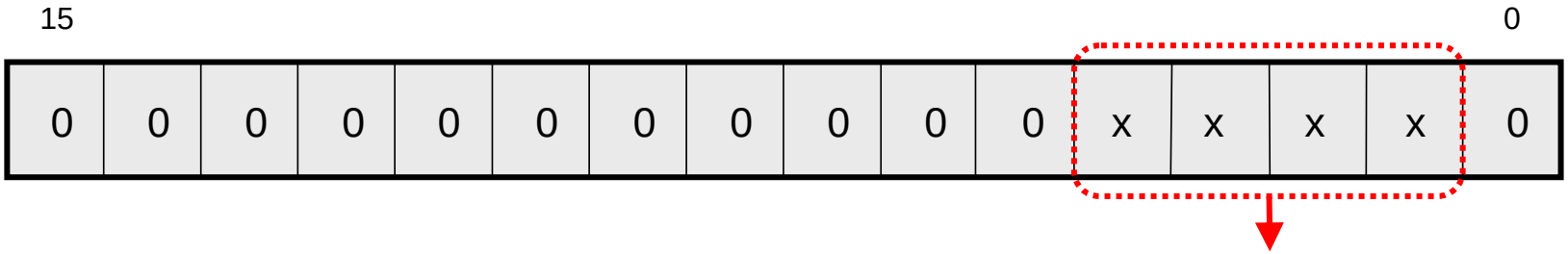
- 约 **7.8ms/**转换
- 约 **950uA**：**LPM3** 中的 **CPU**（约 **1009uA**：**LPM0** 中的 **CPU**）

SMCLK = 1.048MHz, 256 OSR, SD16INTDLY_0

- 约 **244us/**转换
- 约 **1095uA**：**LPM0** 中的 **CPU**

*转换速度约快 **32** 倍，而电流才增大 **15%***

中断向量发生器



信息源	SD16IV 目录
无中断阻碍	0
SD16MEMx 溢出	02h
SD16CCTL0 IFG	04h
保留	06h
保留	08h
保留	0Ah
保留	0Ch
保留	0Eh
保留	10h

- 共享一个中断向量：**SD16_A IFG** 和溢出标志
- 快速解码可显著精减代码尺寸并减少 **CPU** 负载

用汇编程序处理 SD16IV

```
SD16_ISR    add     &SD16IV,PC   ; Offset to Jump table
            reti                 ; SD16IV = 0, no int.
            jmp     Over_ISR     ; Overflow handler

CH1_ISR     ; Handle channel 1 interrupt

Over_ISR    ; Handle overflow interrupt
```

用 C 语言处理 SD16IV

```
// SD16_ISR
#pragma vector=SD16_VECTOR
__interrupt void SD16_ISR(void)
{
   switch  (__even_in_range(SD16IV, 16))
   {
      case 2: Handle overflow;
              break;
      case 4: Channel 1 IFG;
              break;
    }
}
```

使用“__even_in_range ()”

```
106 // Add SD16_A ISR Here
107 #pragma vector=SD16_VECTOR
108 __interrupt void SD16_isr(void)
109 {
110   switch (SD16IV)
111   {
112   case 2:
113     break;
114   case 4:
115     SD16MEM0 = SD16MEM0-offset;
116     break;
117   }
118 }
```

```
SD16_isr:
 008184    0F12           push.w   R15
  switch (SD16IV)
 008186    1F421001       mov.w    &SD16IV,R15
 00818A    2F82           sub.w    #0x4,R15
 00818C    0320           jne      0x8194
    SD16MEM0 = SD16MEM0-offset;
 00818E    928202021201   sub.w    &offset,&SD16MEM0
}
 008194    3F41           pop.w    R15
 008196    0013           reti
}
```

与

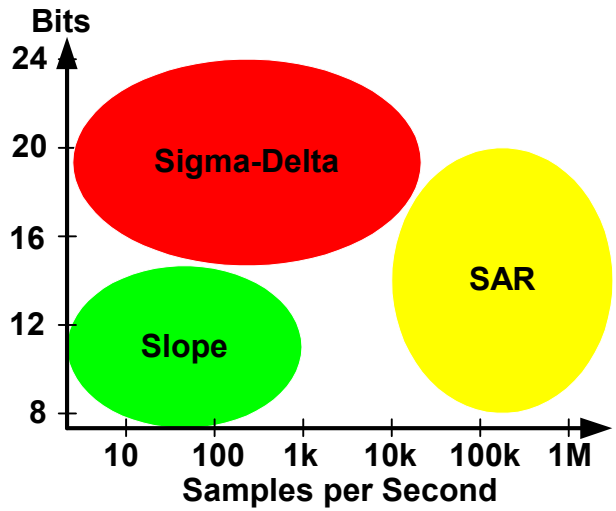
```
106 // Add SD16_A ISR Here
107 #pragma vector=SD16_VECTOR
108 __interrupt void SD16_isr(void)
109 {
110   switch (__even_in_range(SD16IV, 4))
111   {
112   case 2:
113     break;
114   case 4:
115     SD16MEM0 = SD16MEM0-offset;
116     break;
117   }
118 }
```

```
  switch (__even_in_range(SD16IV, 4))
SD16_isr:
 008184    10521001       add.w    &SD16IV,PC
 008188    0013           reti
 00818A    0013           reti
 00818C    003C           jmp      0x818E
    SD16MEM0 = SD16MEM0-offset;
 00818E    928202021201   sub.w    &offset,&SD16MEM0
}
 008194    0013           reti
}
```

选择 MSP430 ADC

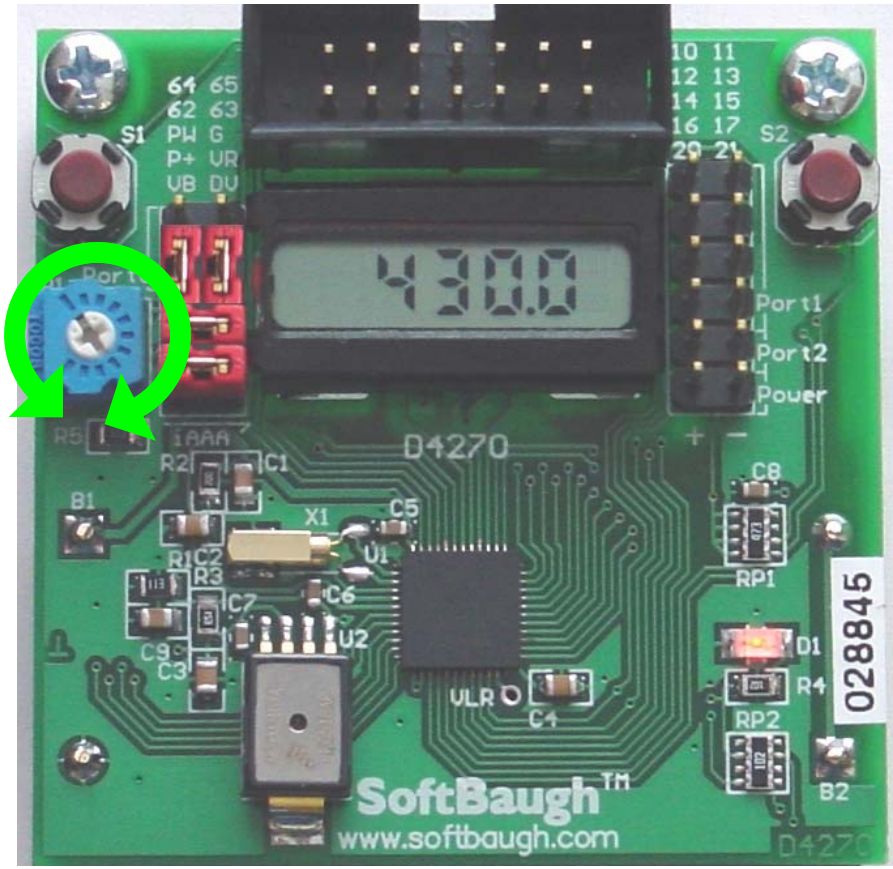
	channels	f_{SAMPLE} (ksps)		res	SINAD (typ)	A_{IN}	reference			triggering	gain	features
		min	max				Ref_{IN}	Ref_{OUT}	Ref_{I_OUT}			
ADC10	8	34	200+	10	57	Vss to Vref	1.4-3.6	1.5/2.5V	+/-1mA	SW/Timer/Cont	N/A	DTC
ADC12	12	34	200+	12	68	Vss to Vref	1.4-3.6	1.5/2.5V	+/-1mA	SW/Timer/Cont	N/A	Conv Mem
SD16	3 ind	~4		16	85	+/-600mV	1.0-1.5	1.2V	+/-1mA	SW/Cont	to 32x	Preload
SD16_A	4 mux'd	~0.03	~5	16	85	+/-600mV	1.0-1.5	1.2V	+/-1mA	SW/Cont	to 32x	Buffered input

- 需要测量电压范围吗？
- A_{IN} 的最大频率是多少？
- 分辨率多高？
- 需要差分输入吗？
- 参考源的范围是多少？
- 需要多个通道？



实验练习 1：测量电压

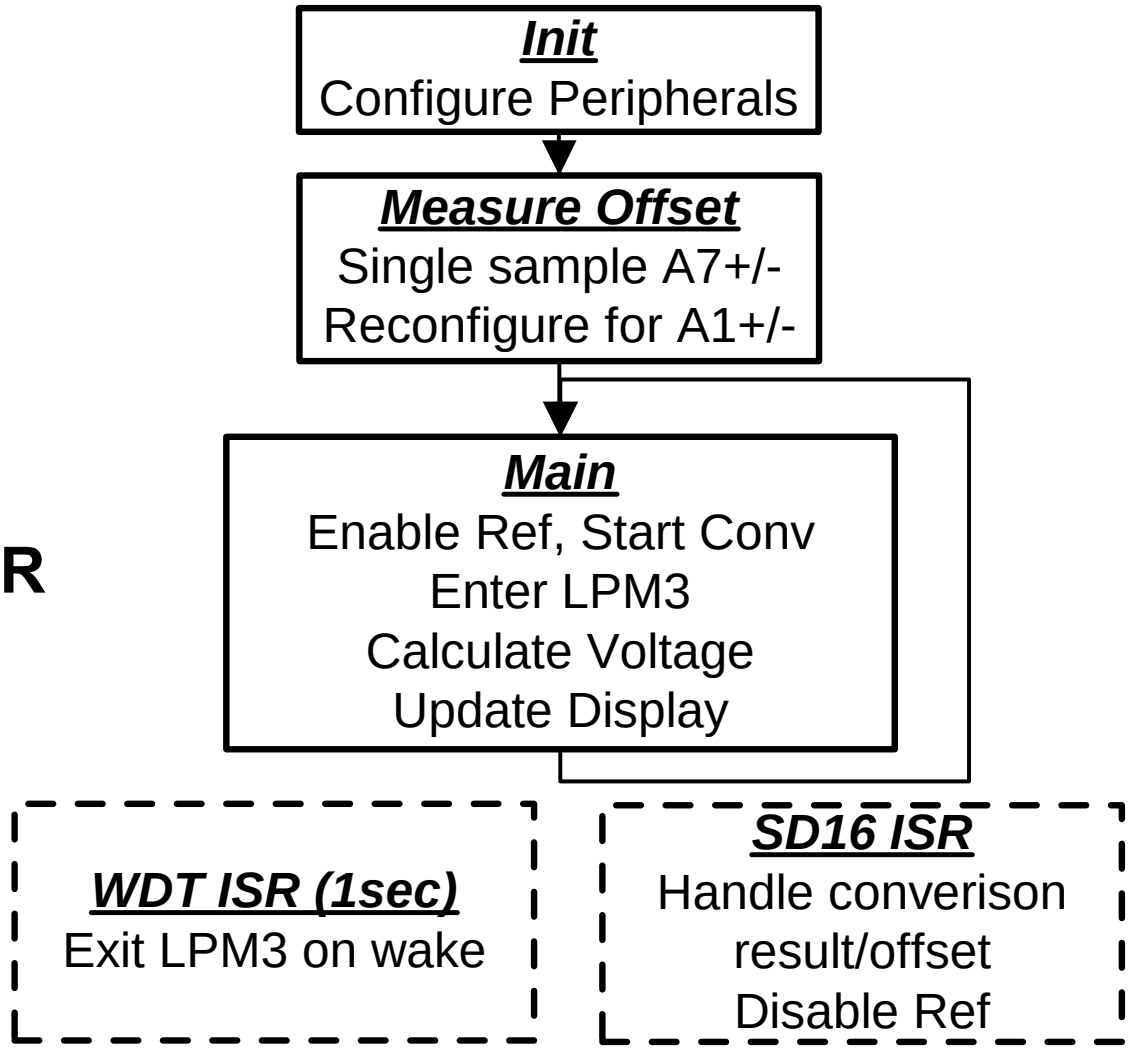
- 配置 **SD16_A**
- 测量可变电压
- 转换至 **mV**
- 每秒更新 **LCD**



实验讲解 1 软件流程

SD16_A 配置：

- **1x** 增益
- 慢速输入缓冲器
- 通道 **A1+/-**
- 内部 V_{REF}
- f_M **= ACLK，256 OSR**
- 单次转换
- **2's** 补码



实验讲解 1 SD16_A 设置

```
// Add SD16_A Configuration Here
  SD16CTL = ____________________________________;
  SD16CCTL0 |= ____________________________________;

// Get internal offset
  SD16INCTL0 = ____________________________________;
  SD16CCTL0 |= ____________________________________;
  while(!(SD16CCTL0 & _______));
  offset = SD16MEM0;

// Configure for external potentiometer
  SD16INCTL0 = ____________________________________;
  SD16CCTL0 = ____________________________________;
```

- 添加适当的 **SD16_A** 设置代码
- **A7** 设置 单次转换
- 启动转换并查询 **IFG**
- **A1** 再设置 单次转换

实验讲解 1 SD16_Main

```
// Main
while (1)
  {
    SD16CTL |= _____________________________;
    SD16CCTL0 |= ________;

    _BIS_SR(LPM3_bits + GIE);

    Disp_Value(2, (float)result * MV_PER_LSB);
  }
```

- 启用内部参考
- 启动转换

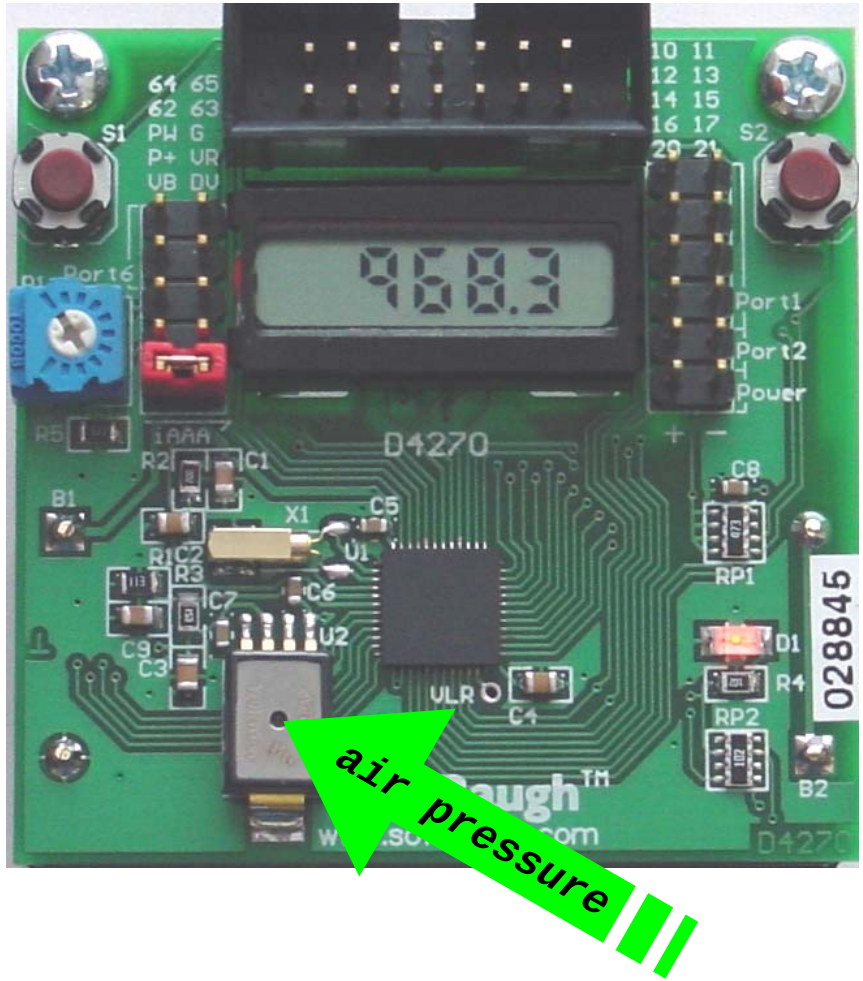
实验讲解 1 SD16_A ISR

```
// Add SD16_A ISR Here
#pragma vector=SD16_VECTOR
__interrupt void SD16_isr(void)
{
  switch (__even_in_range(SD16IV, 4))
  {
  case 2:
         break;
  case 4:
         result = ____________________;
         SD16CTL &= ~(_____________________);
         break;
  }
}
```

- 得到转换结果并减去测量到的偏移值
- 关闭内部参考源

实验练习 2：测量压力

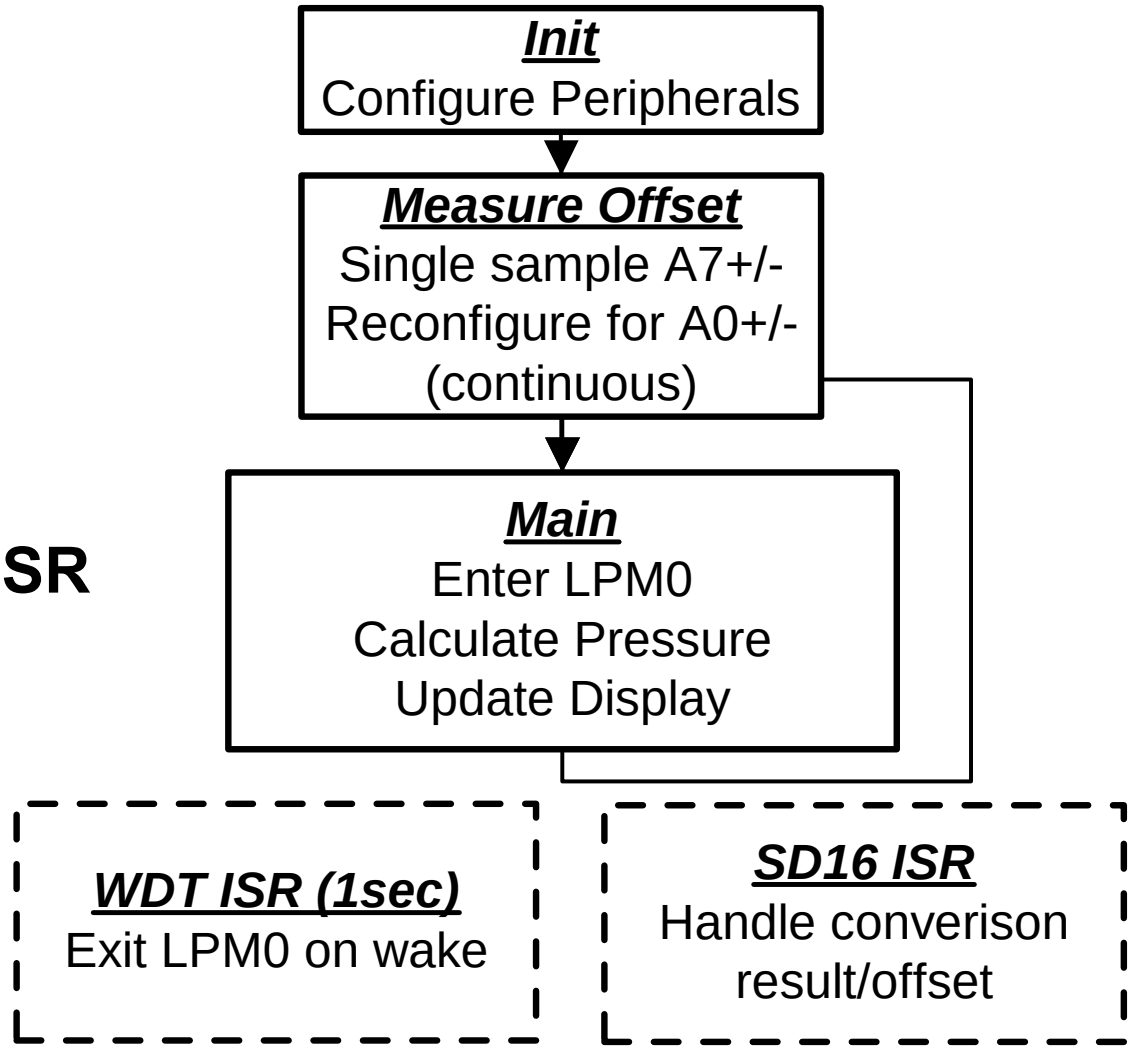
- 配置 **SD16_A**
- 测量压力传感器
- 转换至 **mbar**
- 每秒更新 **LCD**



实验讲解 2 软件流程

SD16_A 配置：

- **32x** 增益
- 中速输入缓冲器
- 通道 **A0+/-**
- 外部 V_{REF}
- f_M **= MCLK，1024 OSR**
- 连续转换
- **2's** 补码



实验讲解 2 SD16_A 设置

```
// Add SD16_A Configuration Here
  SD16CTL = ____________________________________;
  SD16CCTL0 |= ____________________________________;

// Get internal offset
  SD16INCTL0 = ____________________________________;
  SD16CCTL0 |= ____________________________________;
  while(!(SD16CCTL0 & _______));
  offset = SD16MEM0;

// Configure for external potentiometer
  SD16INCTL0 = ____________________________________;
  SD16CCTL0 = ____________________________________;
```

- 添加适当的 **SD16_A** 配置代码
- 配置 **A7** 为单次转换
- 启动转换并查询 **IFG**
- 再配置**A0** 为连续转换

SD16_A: 以一变应千变

- 具备灵活的 **16** 位输入架构
- 兼顾到了参考源稳定时间
- **SD16INTDLYx**：设置*是否足够长？*
- 匹配缓冲器设置，实现最佳功耗性能比
- 理解 f_M、**OSR** 与采样率的关系

根据应用要求匹配 **ADC**

不仅仅是分辨率的问题！

实验讲解 1设置SD16_A 的解决方案

```
// Add SD16_A Configuration Here
  SD16CTL = SD16VMIDON+SD16REFON+SD16SSEL1;
  SD16CCTL0 |= SD16BUF_1+SD16DF+SD16SNGL;

// Get internal offset
  SD16INCTL0 = SD16INCH_7;
  SD16CCTL0 |= SD16SC;
  while(!(SD16CCTL0 & SD16IFG));
  offset = SD16MEM0;

// Configure for external potentiometer
  SD16INCTL0 = SD16INCH_1;
  SD16CCTL0 = SD16BUF_1+SD16DF+SD16SNGL+SD16IE;
```

实验讲解 1 SD16_A Main 解决方案

```
// Main
while (1)
  {
    SD16CTL |= SD16VMIDON+SD16REFON;
    SD16CCTL0 |= SD16SC;

    _BIS_SR(LPM3_bits + GIE);

    Disp_Value(2, (float)result * MV_PER_LSB);
  }
```

实验讲解 1 SD16_A ISR 解决方案

```
// Add SD16_A ISR Here
#pragma vector=SD16_VECTOR
__interrupt void SD16_isr(void)
{
  switch (__even_in_range(SD16IV, 4))
  {
  case 2:
         break;
  case 4:
         result = SD16MEM0-offset;
         SD16CTL &= ~(SD16VMIDON+SD16REFON);
         break;
  }
}
```

实验讲解 2 SD16_A 设置解决方案

```
// Add SD16_A Configuration Here
  SD16CTL = 0;              // This is the default, delete line
  SD16CCTL0 |= SD16BUF_2+SD16OSR_1024+SD16DF+SD16SNGL;

// Get internal offset
  SD16INCTL0 = SD16GAIN_32+SD16INCH_7;
  SD16CCTL0 |= SD16SC;
  while(!(SD16CCTL0 & SD16IFG));
  offset = SD16MEM0;

// Configure for external potentiometer
  SD16INCTL0 = SD16GAIN_32+SD16INCH_0;
  SD16CCTL0 = SD16BUF_2+SD16OSR_1024+SD16DF+SD16SC+SD16IE;
```

- 在进行偏差测量前，确保外部参考源电压已经稳定！