

Periféricos de microcontroladores

Sistemas embebidos para tiempo real

Objetivos

- Utilizar e interpretar manuales de usuario y hojas de datos.
- Familiarizarse con la configuración de periféricos avanzados.

Agenda

- Repaso
- Periféricos del MSP430G2553
 - Reloj: Basic Clock Module+
 - Conversor A/D: ADC10

Repaso: contexto

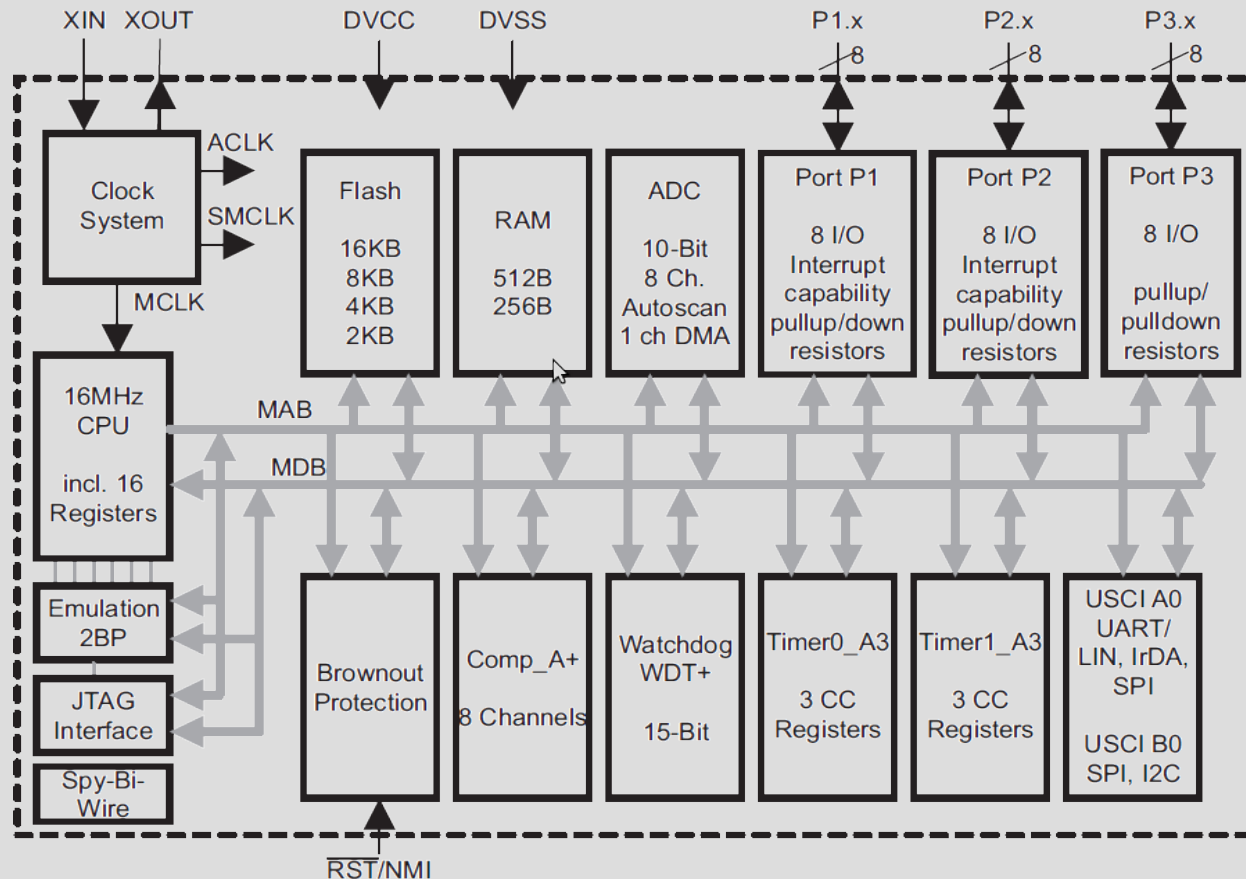
Cosas que importa saber sobre el microcontrolador para empezar un proyecto de sistemas embebidos:

- Tipo procesador, tamaño palabra, frecuencia máxima
- Arquitectura y mapa de memoria
- Registros (cantidad, uso, etc.)
- Características del PC, SR y SP
- Set de instrucciones, modos de direccionamiento.
- Modos de bajo consumo (LPM)
- Interrupciones: ISR, vector de interrupciones, prioridades (última clase)
- Reloj y Periféricos (clase de hoy)

Repaso: periféricos

- Watchdog timer (WDT)
- Puertos digitales de E/S (I/O pin)
- Temporizadores (Timers)
- Conversores A/D y D/A (ADC / DAC)
- Interfaz/protocolos de comunicación
 - E/S de datos digitales (UART, SPI, I2C)
- DMA
- Especializados (no siempre disponibles)
 - MPU / MMU (Memory Protection Unit / Memory Management Unit)
 - SVS

MSP430G2x53



Fuente: "MSP430G2x53 MSP430G2x13 MIXED SIGNAL MICROC." (file: SLAS735J.pdf) Functional Block Diagram, MSP430G2x53, page 5.

Actividad en grupo

- Estudiar funcionamiento del Reloj (Basic Clock Module+)
 - Responder:
 - ¿Cuántos relojes hay? ¿por qué hay más de uno?
 - ¿Cómo se generan?
 - ¿Qué características tienen? Rangos de frecuencia, necesidad de componentes externos, etc.
 - ¿Están prendidos cuando el uC operan en bajo consumo (LPM)?
 - Grupos:
 - 2 a 4 participantes
 - Tiempo:
 - 10 minutos
 - Material:
 - Manual familia MSP430x2xx y hoja de datos MSP430G2553

Reloj

- 3 relojes
 - ACLK
 - Aux clock
 - MCLK
 - Master clock
 - SMCLK
 - Sub-main clock
- 4 fuentes
 - VLOCLK
 - LFXT1CLK
 - XT2CLK
 - DCOCLK

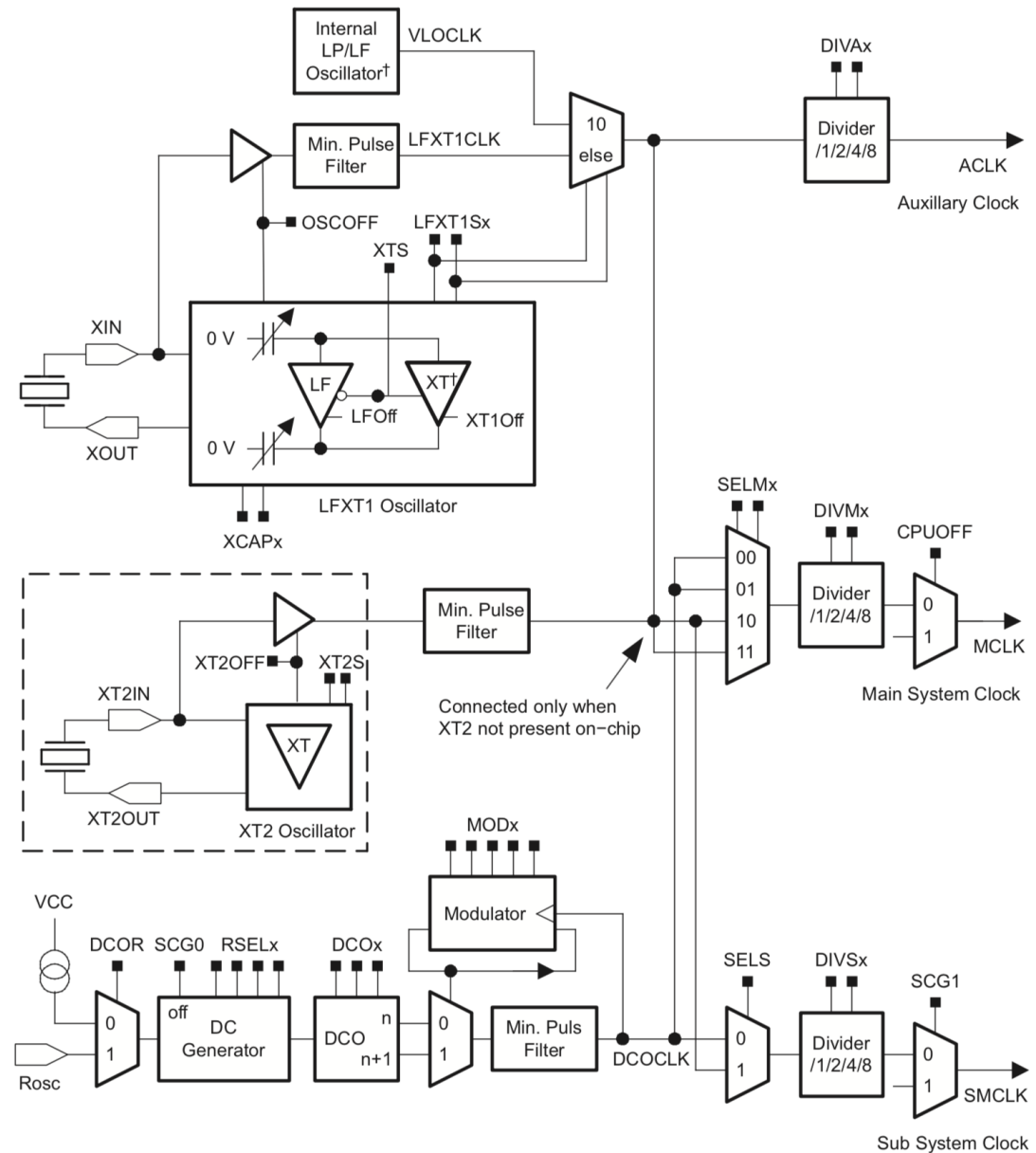


Figure 5-1. Basic Clock Module+ Block Diagram – MSP430F2xx

Reloj (Basic Clock Module+)

- Sin componentes externos:
 - VLOCLK: baja frecuencia (12 kHz)
 - DCOCLK: frecuencia controlada digitalmente
- Requiere componentes externos (por ejemplo un cristal):
 - LFXT1CLK: relojes de 32768 Hz (baja frecuencia) y alta frecuencia
 - XT2CLK: alta frecuencia
- La variedad de relojes permite balancear el trade-off entre consumo y velocidad

Nota: alta frecuencia = 400kHz a 16MHz

Reloj (Basic Clock Module+)

Table 2-2. Operating Modes For Basic Clock System

SCG1	SCG0	OSCOFF	CPUOFF	Mode	CPU and Clocks Status
0	0	0	0	Active	CPU is active, all enabled clocks are active
0	0	0	1	LPM0	CPU, MCLK are disabled, SMCLK, ACLK are active
0	1	0	1	LPM1	CPU, MCLK are disabled. DCO and DC generator are disabled if the DCO is not used for SMCLK. ACLK is active.
1	0	0	1	LPM2	CPU, MCLK, SMCLK, DCO are disabled. DC generator remains enabled. ACLK is active.
1	1	0	1	LPM3	CPU, MCLK, SMCLK, DCO are disabled. DC generator disabled. ACLK is active.
1	1	1	1	LPM4	CPU and all clocks disabled

Configuración y uso de registros en C (y CCS)

- **Seteo x BIT**
 - **P1SEL |= BIT4; // BIT4=0x0010**
 - **P1SEL = P1SEL OR BIT4**
 - **P1SEL = P1SEL OR 00010000**
- **Clear x Bit**
 - **P3DIR &= ~BIT0; // BIT0=0x0001**
 - **P3DIR = P3DIR AND (~BIT0)**
 - **P3DIR = P3DIR AND 11111110**
- **Por registro:**
 - **TACTL = TASSEL_1 + MC_1 + TACLRL + ID_3;**
- **Uso: test BIT**
 - **if (ADC10CTL1 & ADC10BUSY) {}**

Actividad en grupo

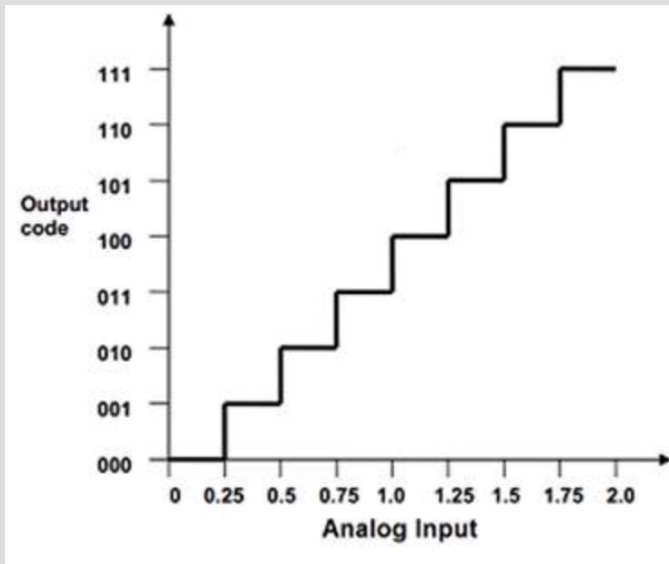
- Configuración de registros del Basic Clock Module+
 - Especificaciones:
 - Configurar el VLOCLK (oscilador interno de bajo consumo de 12 kHz) como ACLK.
 - Dejar incambiados los registros que no se necesitan.
 - Grupos:
 - 2 a 4 participantes
 - Tiempo:
 - 10 minutos
 - Material:
 - Manual familia MSP430x2xx y hoja de datos MSP430G2553

Configuración registros Reloj

- BCSCTL1: no hace falta modificarlo (por defecto está ok) pero podría configurarse:
 - Para hacerlo “evidente” en el código
 - Para evitar bugs (que alguien más lo modifique)
- BCSCTL3: solo hace falta modificar:
 - ```
BCSCTL3 |= LFXT1S_2; // selecciona VLOCLK
// vale comentario sobre BCSCTL1, asume que el uC viene de un reset
// (BCSCTL3 está inicializado con sus valores x defecto)
```
- DCOCTL y BCSCTL2:
  - no hace falta tocarlos (no juegan con VLOCLK)

# Conversor A/D (ADC)

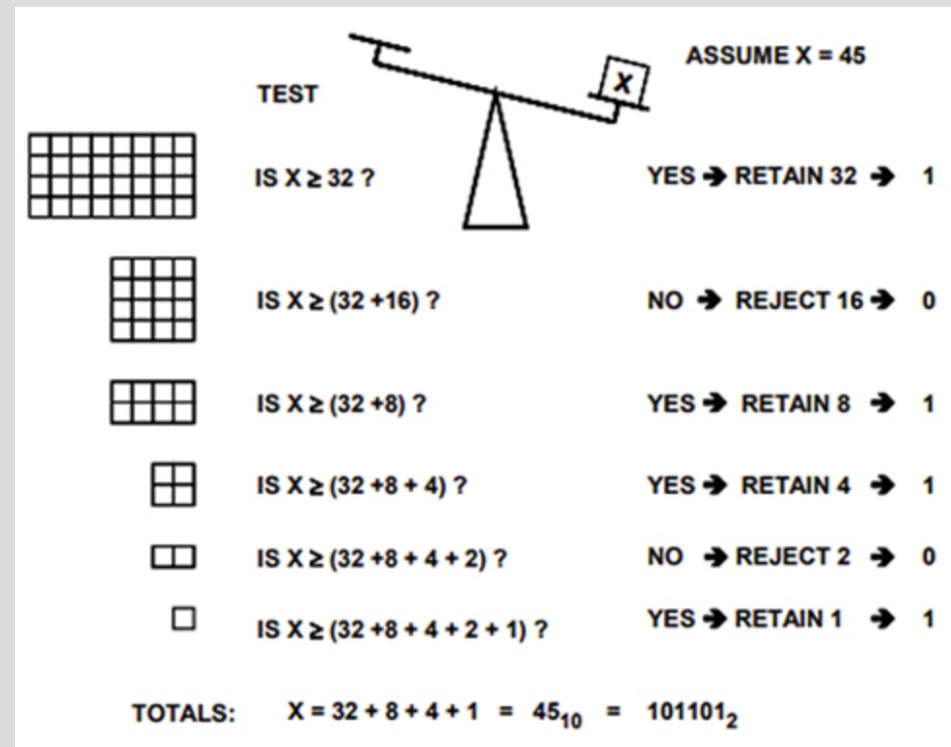
Figuras tomadas de <https://www.digikey.com/en/articles/techzone/2017/sep/adc-dac-tutorial>



3-bit ADC con  $V_{REF}=1.75V$  (FS)

$$V_{IN} = N_{ADC} * (1.75V / 7)$$

$$V_{IN} = N_{ADC} * [V_{REF} / (2^N - 1)]$$



6-bit SAR ADC

# Conversor A/D (ADC)

- Conversor ADC10 (periférico)
  - SAR 10 bits
  - Hasta 8 entradas (externas)
  - Frecuencia de muestro  $\leq 200$ -ksps
  - $V_{REF}$ : on-chip (1.5 V or 2.5 V) o externa

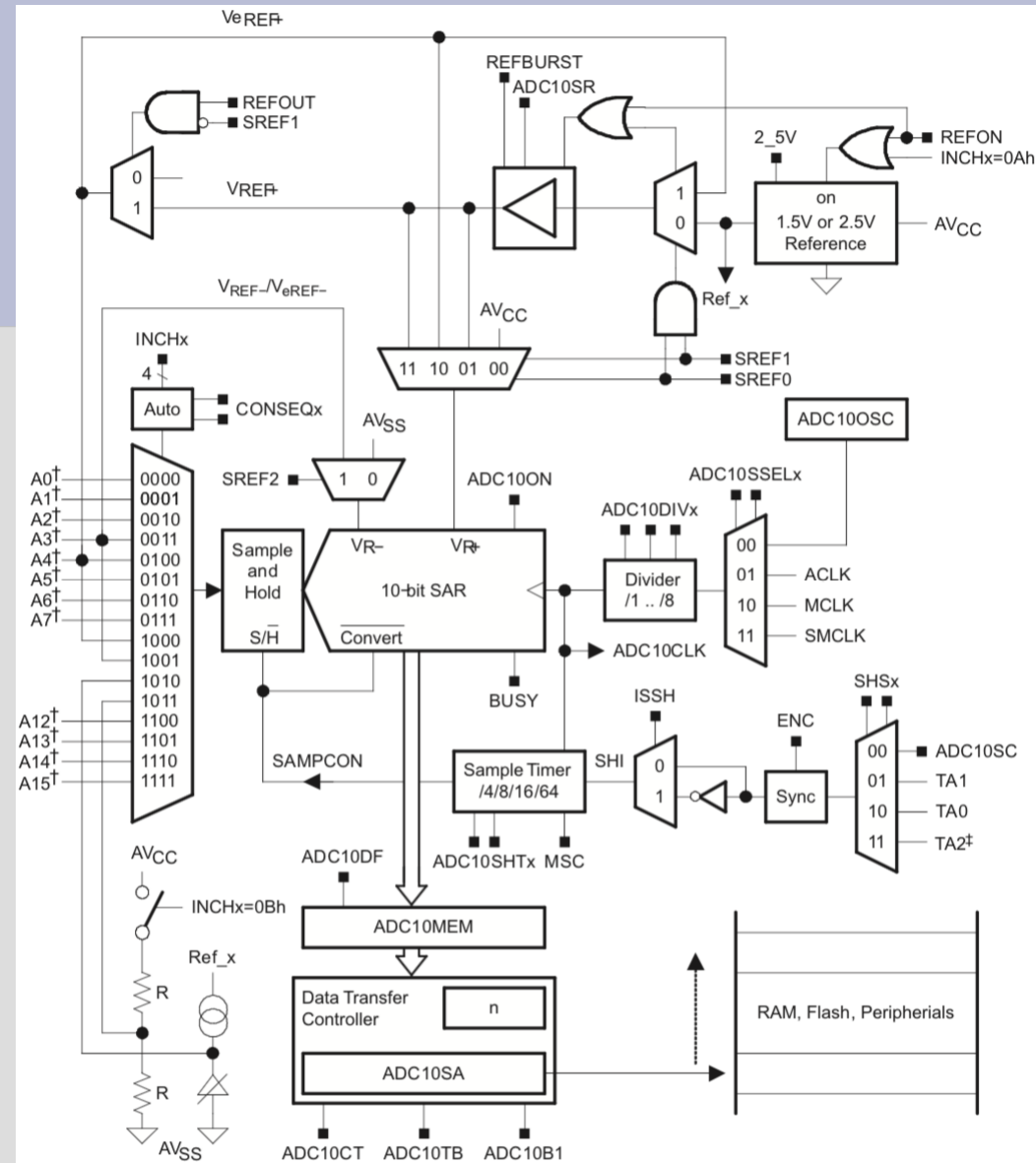
# Actividad en grupo

- Estudiar funcionamiento del ADC10
  - Responder:
    - Identificar bloques en Fig. 22-1
  - Grupos:
    - 2 a 4 participantes
  - Tiempo:
    - 5 minutos
  - Material:
    - Manual familia MSP430x2xx y hoja de datos MSP430G2553



# ADC10

- Bloques:
  - Gen Ref
  - Reloj
  - Fmuestreo
  - Entrada data
  - Salida data



†Channels A12-A15 are available in MSP430F22xx devices only. Channels A12-A15 tied to channel A11 in other devices. Not all channels are available in all devices.  
 ‡TA1 on MSP430F20x2, MSP430G2x31, and MSP430G2x30 devices

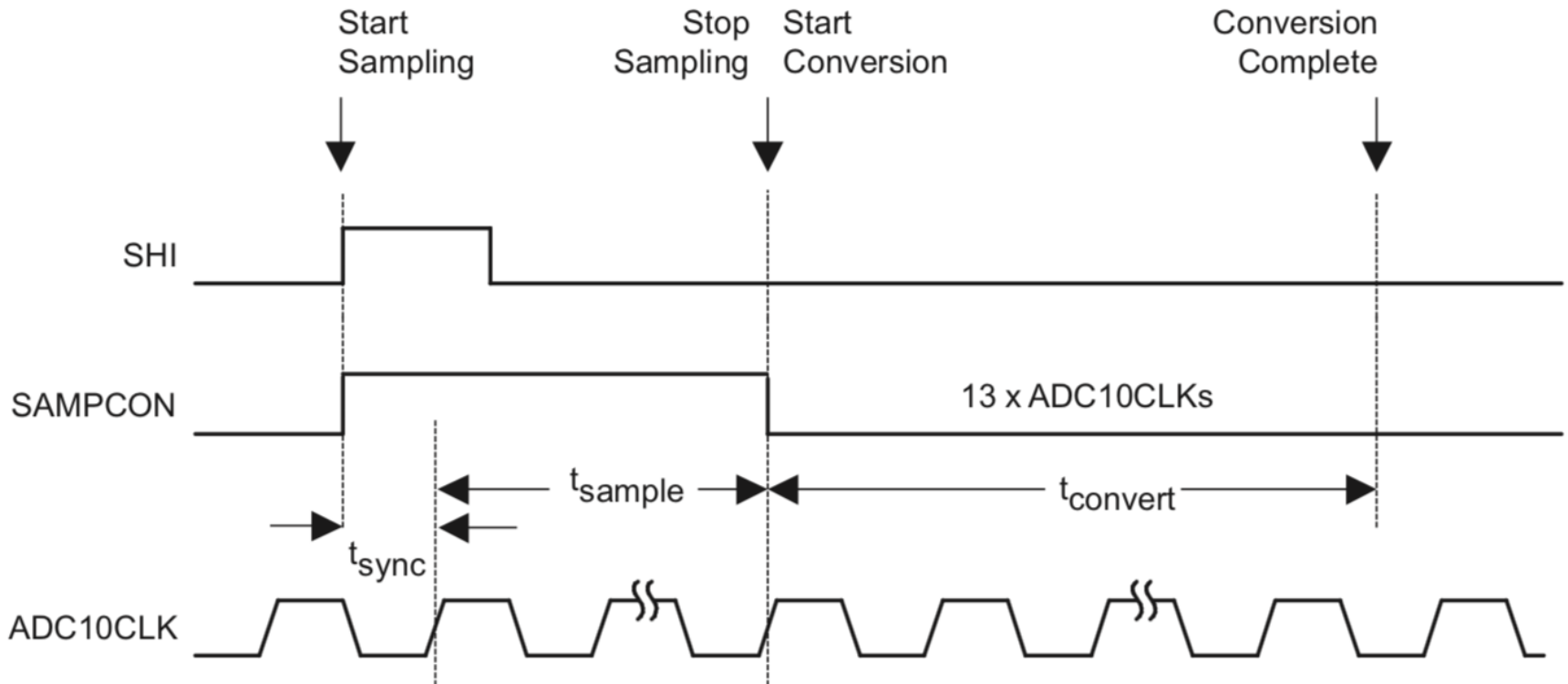
Figure 22-1. ADC10 Block Diagram

# ADC10

$$N_{\text{ADC}} = 1023 \times \frac{V_{\text{IN}} - V_{\text{R-}}}{V_{\text{R+}} - V_{\text{R-}}}$$

- $N = 10 \text{ bits} \Rightarrow 2^N - 1 = 1023$
- $V_{\text{R+}} = \text{REF positiva}$
- $V_{\text{R-}} = \text{REF negativa}$
- $V_{\text{REF}} = \text{REF}$
- $V_{\text{IN}} = \text{entrada analógica}$

# ADC10



**Figure 22-3. Sample Timing**

# ADC10: modo de operación

**Table 22-1. Conversion Mode Summary**

| CONSEQx | Mode                             | Operation                                       |
|---------|----------------------------------|-------------------------------------------------|
| 00      | Single channel single-conversion | A single channel is converted once.             |
| 01      | Sequence-of-channels             | A sequence of channels is converted once.       |
| 10      | Repeat single channel            | A single channel is converted repeatedly.       |
| 11      | Repeat sequence-of-channels      | A sequence of channels is converted repeatedly. |

# ADC10

- Single-channel single-conversion
- Arranque (SHS=0):
  - ENC=1 y ADC10SC=1
- Fin:
  - ADC10IF=1
  - Resultado en ADC10MEM

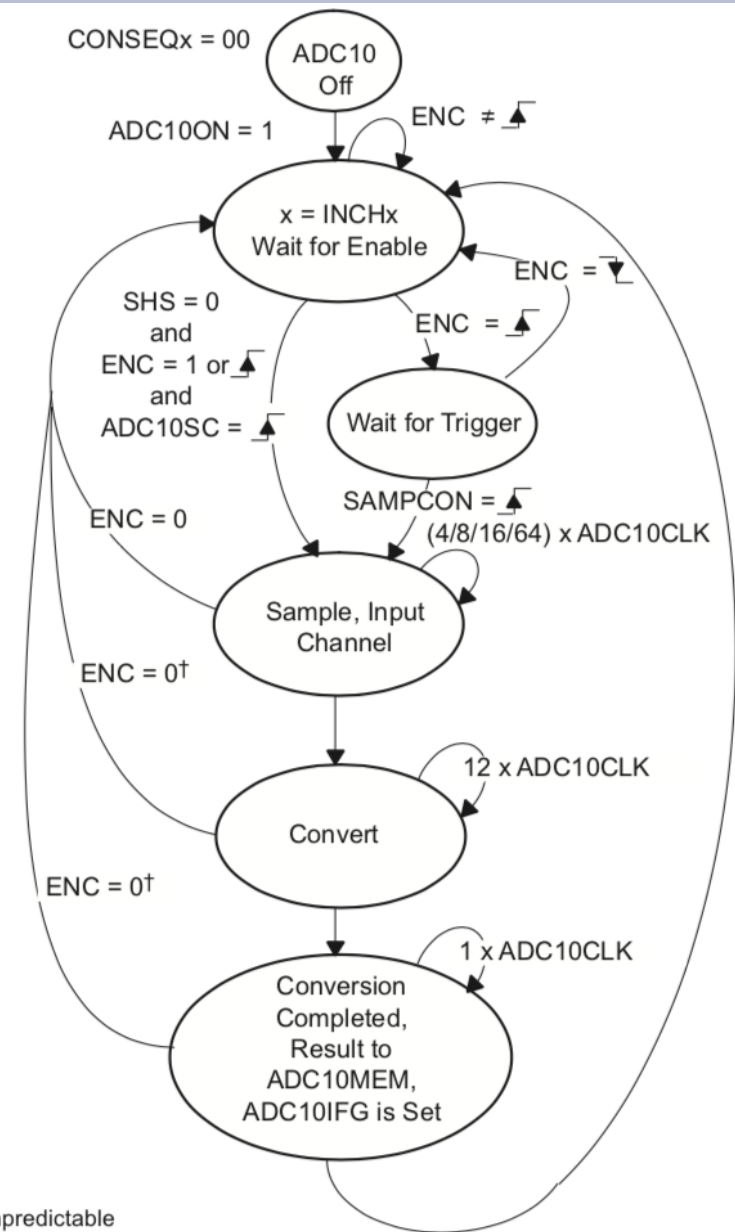
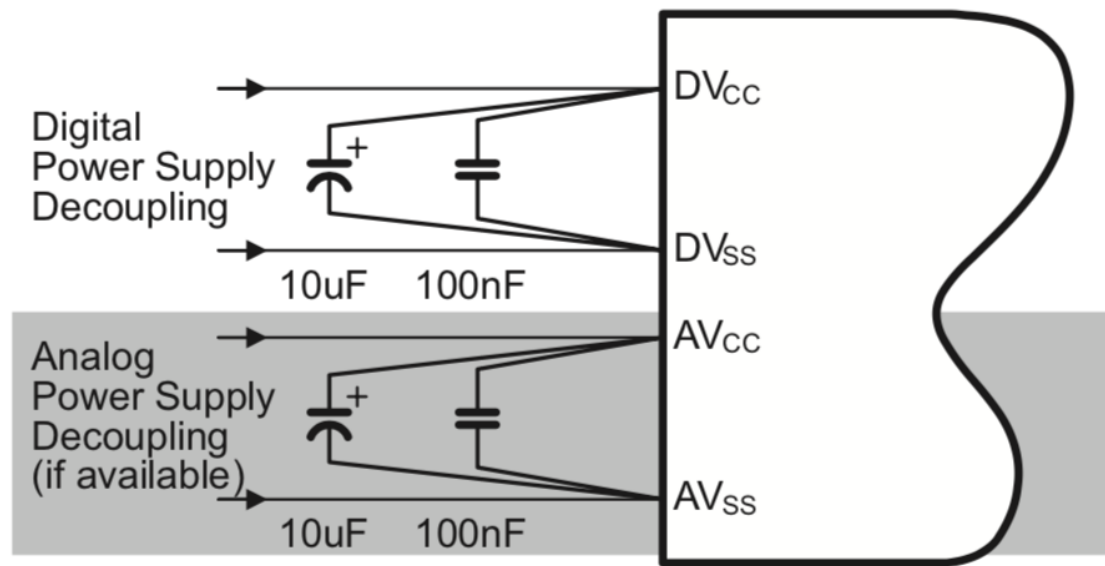


Figure 22-5. Single-Channel Single-Conversion Mode

# ADC10



**Figure 22-14. ADC10 Grounding and Noise Considerations (Internal  $V_{REF}$ )**

# Actividad en grupo

- Configuración de registros del ADC10
  - Especificaciones:
    - 1 sola entrada analógica en pin A5
    - Período de muestreo = 250ms (usando interrupción del TIMER\_A cada 250 ms: similar LAB2 pero usando VLOCLK como ACLK)
    - Modo: Single-channel-single-conversion
    - Reloj: interno del ADC (ADC10OSC)
    - Funcionamiento lo más rápido que se pueda
    - Maximizar rango dinámico:  $V_{R+}=V_{CC}$  and  $V_{R-}=V_{SS}$
    - Referencia interna = 1.5 V, disponible en pin A4 (para test)
  - Grupos:
    - 2 a 4 participantes
  - Tiempo:
    - 15 minutos
  - Material:
    - Manual familia MSP430x2xx y hoja de datos MSP430G2553

# ADC10CTL0

// asume que el uC viene de un reset y los registros están inicializados con sus valores por defecto

```
ADC10CTL0 |= SREF_0; // (VR+=VCC and VR-=VSS)
ADC10CTL0 |= ADC10SHT_0; // ADC10SHTx=00 (4x ADC10CLKs) queremos que
demore lo menos posible en el muestreo
ADC10CTL0 |= ADC10SR; // ADC10SR= 1, ADC10SR=0 → fs<200ksps /
 ADC10SR=1 → fs<50ksps (ADC10SR=1 reduce consumo)
ADC10CTL0 |= REFOUT; // REFOUT=1, REF disponible en pin VREF+
ADC10CTL0 &= ~REFBURST; // REFBURST=0 prende el buffer de Vref
siempre, Si REFBURST=1 el buffer está ON solo durante sample-and-
conversion (ver Sección 22.2.3.1)
ADC10CTL0 &= ~REF2_5V; // REF2_5V=0, Vref = 1.5 V.
ADC10CTL0 |= REFON; // REFON=1 habilita Vref.
ADC10CTL0 |= ADC10ON; // prende ADC
ADC10CTL0 |= ADC10IE; // ADC10IE=1 habilita interrupción de ADC10
// ADC10CTL0 &= ~MSC; // MSC=0, The sampling requires a rising
edge of the SHI signal to trigger each sample-and-conversion.
```



# Otros registros

```
// ADC10 control register 0: ADC10CTL0
ADC10CTL1 |= INCH_5; // entrada por pin A5
ADC10CTL1 |= SHS_0; // SHSx=00 la conversión se dispara
// seteando el bit ADC10SC
ADC10CTL1 &= ~ADC10DF; // ADC10DF=0 (binario)
ADC10CTL1 |= ADC10DIV_0; // sin prescaler, tan rápido como sea
// posible
ADC10CTL1 |= ADC10SSEL_0; // ADC100SC
ADC10CTL1 |= CONSEQ_0; // single-channel-single-conversion

// ADC10 input enable register 0
ADC10AE0 |= BIT5; // input = A5
```

# Deberes

- Implementar la digitalización de una señal analógica en un MSP430G2553
  - Especificaciones:
    - Utilizar el ADC10 y el Basic Module Clock+ según se configuraron durante esta clase
    - Definir adc10.c y adc10.h
    - Mostrar el valor digitalizado en una variable en el CCS
  - Grupos:
    - 2 a 4 participantes
  - Tiempo:
    - 15 minutos
  - Material:
    - Manual familia MSP430x2xx y hoja de datos MSP430G2553

# Pistas para la implementación

```
// En adc10.c
void adc_start(){
 while (ADC10CTL1 & ADC10BUSY){}; //if ADC10BUSY=1 then ADC busy
 ADC10CTL0 |= ADC10SC + ENC;
 //ADC10SC = 1 start conversion, ENC = 1 enable conversion
 return;
}

#pragma vector=ADC10_VECTOR // ADC10IFG is automatically reset when IRQ is accepted
__interrupt void ADC10_ISR(void){
 static const int ADC_FSR = 1023;
 static const float Vcc = 3.3; // Vss=0
 int Nadc=ADC10MEM;
 Vin = Vcc*Nadc/ADC_FSR; // Vss=0
}

// En timerA_hw.c
#pragma vector=TIMER0_A0_VECTOR
__interrupt void tic250 (void){
 timer_inc();
 P1OUT ^= LED1; // Conmuta LED1 (P1.0) usando XOR
 adc_start();
}
```

# Optimizaciones

- Bajar consumo:
  - Dormir uC cuando no se hace nada
  - Configurar puertos I/O no usados
  - ¿Apagar ADC entre conversiones?
  - Inhabilitar Vref externa (era solo para test)
  - Ver sugerencias de CCS

# Bibliografía

- “MSP430x2xx Family User's Guide”
- Hoja de datos MSP430G2553