Electrónica Digital II

Santiago Rúa Pérez, PhD.

18 de septiembre de 2022

INTERFAZ ANÁLOGA

Objetivos

- Entender el funcionamiento de una interfaz análoga en un sistema embebido.
- Entender el funcionamiento de un conversor SAR.
- Comprender la configuración de registros para la lectura de un puerto convertidor.

Porque es necesario?

- Los sistemas embebidos a menudo requieren medir parámetros físicos.
- Usualmente estos parámetros físicos son continuos (análogos) y no se encuentran en forma digital las cuales pueden ser procesados por las computadores
- Algunas de estas variables son:
  - **Temperatura**: termómetros, controlador de motores en un carro, monitoreo en un reactor químico, seguridad en los microcontroladores.
  - **Luz o intensidad de luz**: cámara digital, receptor IR, monitor UV.
  - **Posición angular**: medidor de viento o nudos.
  - **Presión**: monitores de la presión sanguínea, altímetro, controlador en carro, detector de tsunami.
  - **Aceleración**: estabilidad del vehículo, controles de video juegos.
  - **Otros**: controlador de touchscreen, EKG, EEG, etc.

Panorama general - Ejemplo

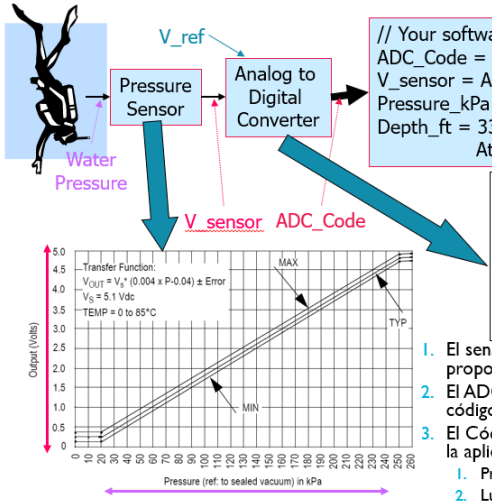


Figure 4. Output vs. Absolute Pressure

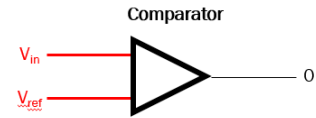
```
// Your software
ADC_Code = ADC0->R[0];
V_sensor = ADC_code*V_ref/1023;
Pressure_kPa = 250 * (V_sensor/V_supply+0.04);
Depth_ft = 33 * (Pressure_kPa –
                 Atmos_Press_kPa)/101.3;
```

Voltages	ADC Output Codes
V_ref	111..111 111..110 111..101 111..100
V_sensor	ADC_Code
Ground	000..001 000..000

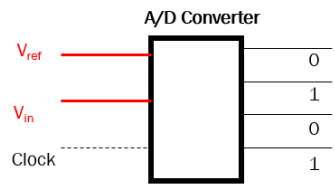
1. El sensor detecta presión del agua y genera una salida proporcional de voltaje V_sensor.
2. El ADC genera una salida proporcional digital en código entero basado en V_sensor y V_ref
3. El Código convierte ese entero un un valor útil para la aplicación
   1. Primero un flotante representando el voltaje
   2. Luego otro flotante representando la presión
   3. Finalmente, otro flotante representando la profundidad.

Obteniendo de análogo a digital

- Un comparador nos dice si $V_{in} > V_{ref}$.
  - Se compara un **voltaje análogo de entrada** con un **voltaje de referencia** para determinar cual es mas grande, retornando un bit 1.
  - Indicar si la profundidad es mayor a 100 m.
  - Poner V_{ref} al voltaje que marca dicha presión.

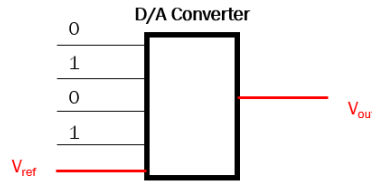


- Un convertidor de análogo a digital (ADC) nos dice que tan grande es V_{in} en comparación de V_{ref}.
  - Lee una señal análoga y produce el correspondiente numero bit binario a la salida.

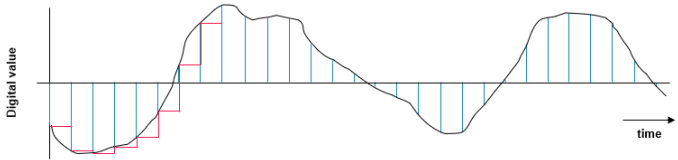


Conversión de digital a análogo

- Se necesita generar un voltaje análogo o corriente como una señal de salida.
  - Ejemplo: señal de audio, señal de intensidad de brillo.
- DAC: Generar un voltaje análogo como una fracción del voltaje de referencia.
- Ecuación de conversión entre digital a análogo.
  - n código de entrada.
  - N número de bits de resolución.
  - V_{ref} voltaje de referencia.
  - V_{out} voltaje de salida. Esta dado por $V_{out} = V_{ref} * n/2^N$



Muestreo y cuantización de una señal



- Una señal es muestreada a un frecuencia constante cada δt.
  - Cada muestra representa un valor instanteneo de amplitud en dicho instante.
  - A los 37 ms, la entrada tiene el valor $1.913\,419\,214\,124$ V
  - Muestrear convierte una señal en tiempo continuo a valores discretos.
- La muestra puede ahora ser cuantizada en un valor digital.
  - La cuantización consiste en representar un valor continuo (análogo) con el valor más cercano discreto.
  - El valor de voltaje de entrada $1.913\,419\,214\,124$ V se puede representar con el código 0×018, ya que se encuentra entre el rango 1.901 V a 1.998 V el cual corresponde a dicho código.

Como se obtiene el valor o código de n para representar el voltaje de entrada en el ADC?

La ecuación general está dada por

$$n = \left\lfloor \frac{(V_{in} - V_{-ref})2^N}{V_{+ref} - V_{-ref}} + 1/2 \right\rfloor$$

donde n es el código, V_{in} el voltaje de entrada, V_{+ref} voltaje de referencia superior, V_{-ref} voltaje de referencia inferior, y N el número de bits. Por lo general el voltaje de referencia inferior es $0\,\text{V}$, entonces

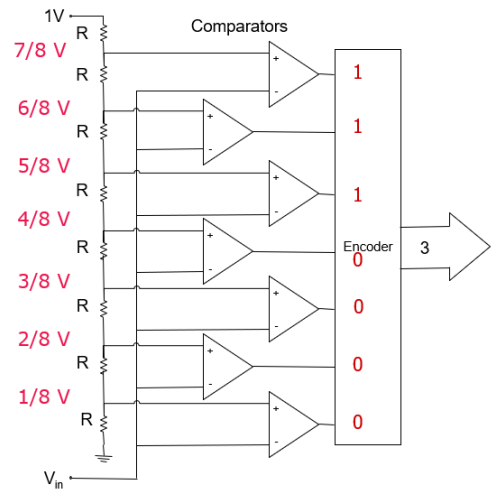
$$n = \left\lfloor \frac{V_{in}2^N}{V_{+ref}} + 1/2 \right\rfloor$$

Ejemplo: suponga que se tiene un convertidor ADC de 10 bits, el voltaje de referencia es de $5\,\text{V}$, y el de entrada es de $3.3\,\text{V}$, obtenga el código de salida. Aplicando la formula se tiene que

$$n = \left\lfloor \frac{3.3\,\text{V}\ 2^{10}}{5\,\text{V}} + 1/2 \right\rfloor = 676$$

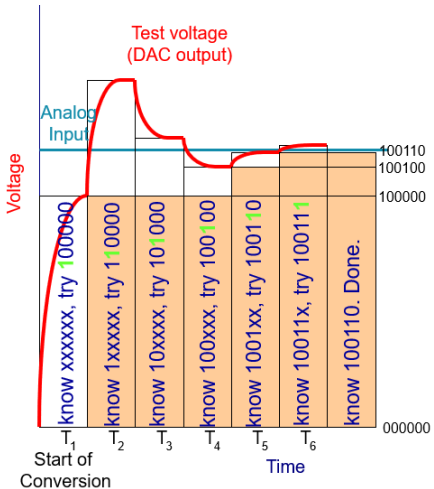
Conceptos del ADC - Convertidor Flash

- Un divisor multinivel de voltaje es usado para poner la salida sobre todo el rango de conversión.
- Un comparador es usado en cada nivel para determinar si el voltaje es inferior o superior en dicho nivel.
- La salida de los comparadores son codificadas en número binarios con prioridad.
- Componentes usados: 2^N resistencia, y $2^N - 1$ comparadores.
- Nota:
  - El divisor de resistencias genera un voltaje el cual no satisface $1/2$ bit, lo que ocasiona un error de un bit.
  - Se puede cambiar el offset con los valores de las resistencias.

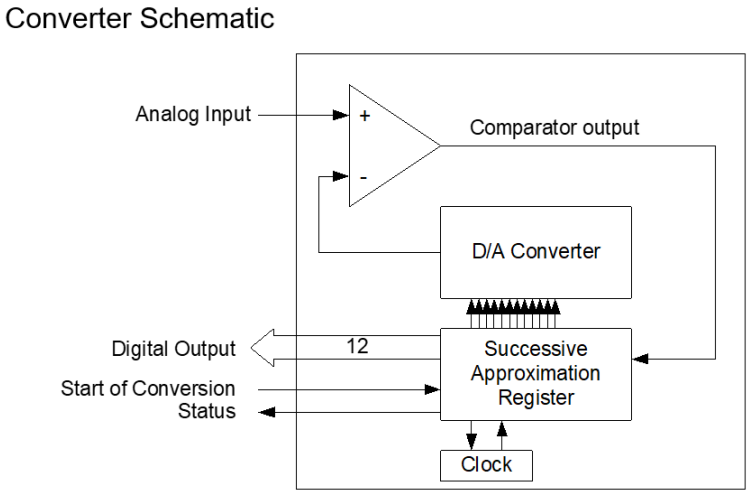


Conceptos del ADC - Convertidor de aproximaciones sucesivas (SAR)

- Se aproxima sucesivamente el voltaje de entrada realizando una búsqueda binaria y un DAC.
- El registro SA almacena el valor actual de la aproximación.
- Se pone todas las entradas del DAC en cero.
- Pone un uno en el bit más significativo del DAC.
- Se repite
  - Poner el siguiente bit del DAC en 1.
  - Esperar a que se estabilice y comparar el DAC con la salida.
  - Si la salida del DAC es menor que la entrada entonces se pone el siguiente bit en 1, sino se el bit actual en cero.



Esquemático del SAR



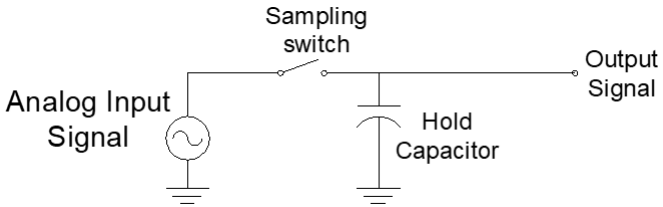
Conceptos del ADC - Métrica y problemas

- La **linealidad** mide que tanto la conversión se mantiene en una linea recta.
- **Linealidad diferencial** mide la igualdad entre pasos.
- **Tiempo de conversión**: desde que se comienza la conversión hasta que está listo el resultado.
- **Tasa de conversión**: inverso del tiempo.

Cuando se muestrea una señal analógica se debe tener en cuenta los siguiente problemas.

- **Criterio de Nyquist**: muestrear al menos 2 veces más rápido que la señal para poder reconstruirla.
- Este criterio asume que se tienen filtro perfecto.
- Se debe elegir una frecuencia de muestreo mayor para atenuar efectos de aliasing.

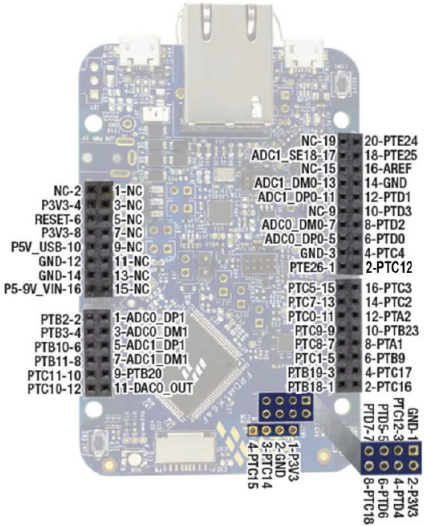
Dispositivos de muestreo y retención



- Algunos convertidores ADC requieren que la entrada se mantenga constante durante la conversión (ej: SAR).
- En otros casos, la captura del pico o el muestreo a un punto específico requiere de un circuito adicional.
- Muchos de los ADC contienen dicho circuito.

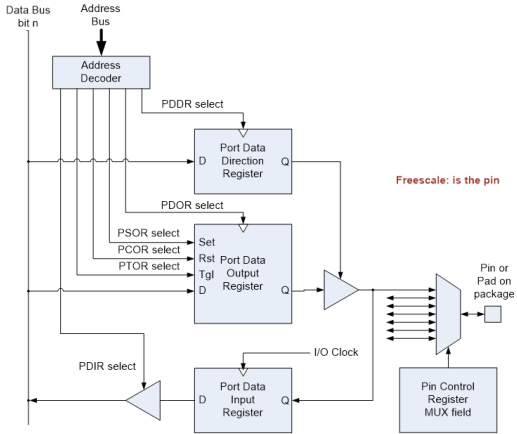
Pines del K64

- 100 LQFP
- 2 ADC de 16-bit ADC con hasta 14 canales.
- 2 comparadores.
- 1 DAC de 12-bits

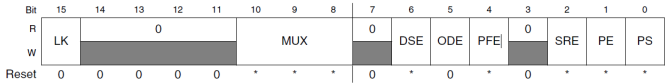


Usar el pin como entrada o salida análoga

- Recuerde configurar con el registro PCR, tanto dirección como MUX.
- Dato: la salida hay diferentes formas para acceder, y la entrada existe el registro.



Registro de control de pin PCRn



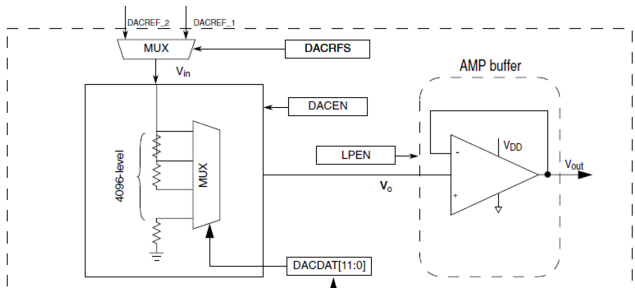
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LK	0				MUX			0	DSE	ODE	PFE	0	SRE	PE	PS
W																
Reset	0	0	0	0	0	*	*	*	0	*	0	*	0	*	*	*

144 LQFP	144 MAP BGA	121 XFB GA	100 LQFP	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	EzPort
23	J1	H1	14	ADC0_DP1	ADC0_DP1	ADC0_DP1								
24	J2	H2	15	ADC0_DM1	ADC0_DM1	ADC0_DM1								
25	K1	J1	16	ADC1_DP1	ADC1_DP1	ADC1_DP1								

El campo del MUX define las conexiones internas de la señal.

MUX (bit 10-8)	Configuración
000	Pin deshabilitado
001	ALT 1 - GPIO
010	ALT 2
011	ALT 3
100	ALT 4
101	ALT 5
110	ALT 6
111	ALT 7

Conversor digital a análogo (DAC)



- Cargar en el registro DACDAT con un dato de 12-bit.
- El MUX selecciona un nodo de la red de resistencia para crear la salida dada por $V_o = (N + 1)V_{in}/2^{12}$.
- V_o es acoplado a la salida por el amplificador operacional.

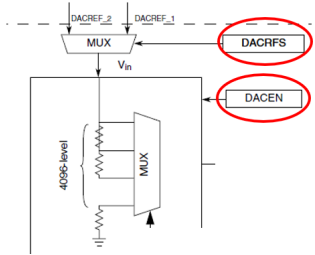
Modos de operación del DAC

- Normal
  - DAT0 es convertido a salida de voltaje inmediatamente.
- Buffered
  - Los datos a la salida son almacenados en un buffer de 16 bits.
  - El siguiente dato es enviado al DAC cuando ocurre un evento por disparo.
    - Evento por software - escribir al campo DACSWTRG en el DACx_C0
    - Evento por hardware - del timer.
  - Modo normal - buffer circular
  - Modo de escaneo una única vez
  - Bandera de estatus en DACx_SR

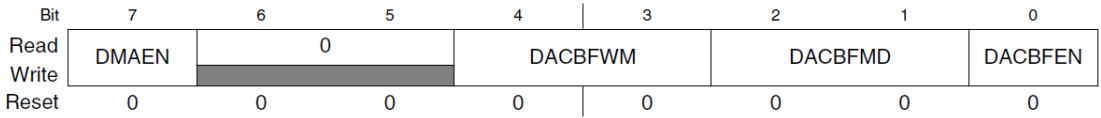
Registro de control 0 para el DAC (DACx_C0)

Bit	7	6	5	4	3	2	1	0
Read	DACEN	DACRFS	DACTRGSEL	0	LPEN	DACBWIEN	DACBTIEN	DACBBIEN
Write				DACSWTRG				
Reset	0	0	0	0	0	0	0	0

- DACEN - habilita el DAC cuando es un 1.
- DACRFS - seleccionar la referencia de voltaje para el DAC.
  - 0: DACREF_1 conectado a VREFH
  - 1: DACREF_2 conectado al VDDA
- LPEN - modo de bajo consumo.
  - 0: modo de alta velocidad. Rápido ($15\,\mu s$ tiempo de asentamiento), pero usa mayor potencia.
  - 1: modo bajo consumo. Lento ($100\,\mu s$) pero menos potencia.
- Registros adicional para el control en modo buffered.



Registro de control 1 para el DAC (DACx_C1)



Bit	7	6	5	4	3	2	1	0
Read	DMAEN	0		DACBFWM		DACBFMD		DACBFEN
Write								
Reset	0	0	0	0	0	0	0	0

- DACBFEN:
  - 0: Deshabilita el modo buffer.
  - 1: Habilita el modo buffer.
- DACBFMD - selecciona el modo buffer.
  - 0: Modo normal
  - 1: Modo de un solo escaneo.

Registro de datos para el DAC

- Estos registros son solo de 8 bits.
- DATA[11:0] se almacena en dos registros.
  - DATA0: byte bajo en DACx_DATnL
  - DATA1: nibble alto en DACx_DATnH

Pasos para trabajar con el DAC

- Habilite el clock del DAC
- Ponga el MUX del PCR para trabajar análogo.
- Habilite el DAC
- Configure el DAC: voltaje de referencia, bajo consumo, modo normal.
- Escriba el registro del DAC

Ejemplo - DAC

Genere una señal de salida triangular utilizando el DAC.

```
1  void Init_DAC(void) {
2    // Enable clock to DAC
3    //SIM->SCGC6 |= SIM_SCGC6_DAC0_MASK;
4    SIM->SCGC2 |= SIM_SCGC2_DAC0_MASK;
5
6    // Disable buffer mode
7    DAC0->C1 = (uint8_t)0;
8
9    // Enable DAC, select VDDA as reference voltage
10   DAC0->C0 = (uint8_t)(DAC_C0_DACEN_MASK | DAC_C0_DACRFS_MASK);
11 }
12
```

Ejemplo - DAC

Genere una señal de salida triangular utilizando el DAC.

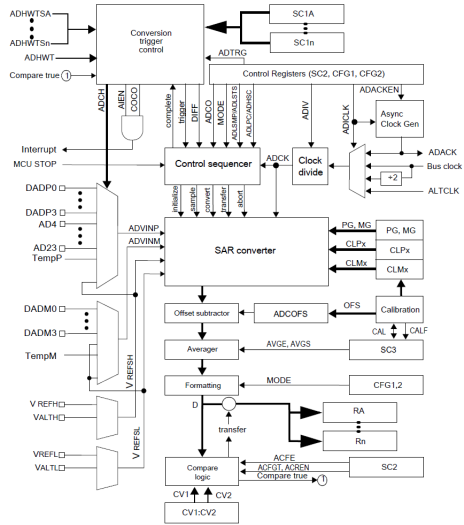
```
1  void Triangle_Output(void) {
2    int i=0, change=1;
3
4    while (1) {
5      DAC0->DAT[0].DATL = DAC_DATL_DATA0(i);
6      DAC0->DAT[0].DATH = DAC_DATH_DATA1(i >> 8);
7
8      i += change;
9      if (i == 0) {
10       change = 1;
11     } else if (i == DAC_RESOLUTION-1) {
12       change = -1;
13     }
14   }
15 }
16
```

Panorama general del ADC

- Usa aproximaciones sucesivas para la conversión.
- Soporta múltiples resoluciones: 16, 12, 8 bit.
- Soporta conversiones diferenciales o de un solo canal.
- Comparación automática e interrupción por nivel.
- Promedio de datos por hardware.
- Sensor de temperatura.

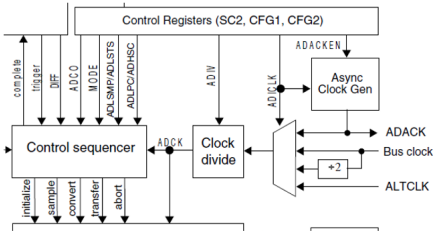
Sistema general del ADC

- Inicialización del ADC
  - Configurar el clock
  - Seleccionar el voltaje de referencia a trabajar.
  - Seleccionar la fuente de trigger.
  - Seleccionar el canal.
  - Seleccionar otros parámetros.
- Disparar la conversión.
- Leer resultados.



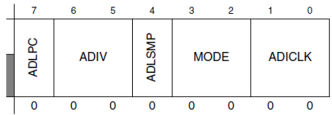
Configuración del clock

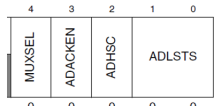
- Seleccione la fuente del clock ADICLK
  - Bus clock.
  - ADACK: clock local, permite operaciones del ADC mientras la CPU está parada.
  - ALTCLK: clock alternativo
- Divide el clock seleccionado por un factor de ADIV, creando asi ADCK.
- El ADCK resultante debe estar en un rango valido, que depende de las resolución.



Registros para configuración del clock

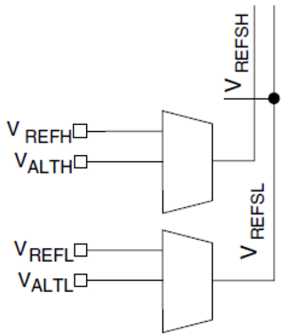
- ADCx_CGF1
  - ADIV: divide el clock por 2^{ADIV}
  - ADICLK: selección del bus de entrada.
    - 00: bus clock
    - 01: bus clock/2
    - 10: ALTCLK
    - 11: ADACK
- ADCx_CGF2
  - ADACKEN: habilita clock asíncrono.





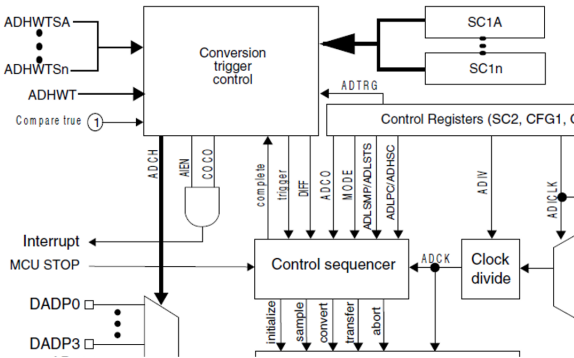
Selección del voltaje de referencia

- Dos pares de voltaje de referencia estan disponibles
  - V_{REFH}, V_{REFL}
  - V_{ALTH}, V_{ALTL}
- Dentro del registro SC2 el campo de bits REFSEL
  - 00: V_{REFH}, V_{REFL}
  - 01: V_{ALTH}, V_{ALTL}
  - 10, 11: reservado
- K64
  - V_{ALTH} conectado a V_{DDA}



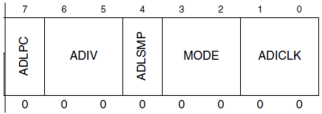
Selección del trigger para la conversión

- ADTRG en SC2
  - 0: trigger por software
  - 1: trigger por hardware
- Trigger por software
  - Escribir al SCIA
- Buffering ping-pong
  - SCIA vs SCIn
- Trigger por hardware
  - Flanco de subida para la señal ADHWT
  - Fuente para el ADHWT: TPM, LPTMR, PIT, RTC, EXTRG_IN



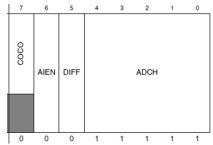
Selección del modo de conversión

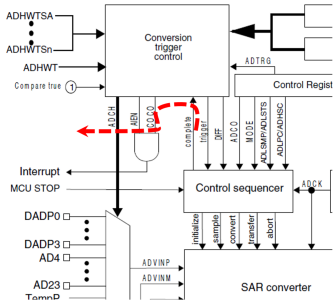
- Bajo consumo
  - Poner ADLPC a 1 (en el ADCx_CFG1)
  - Velocidad máxima del reloj mas lenta.
- Largo tiempo de muestreo
  - Poner ADLSMP en 1 (en el ADCx_CFG1)
  - Se puede seleccionar tiempo de muestreo mas largos con ADLSTS
- Modo de conversión
  - Modo (en ADCx_CFG1)
  - Configurar la precisión del resultado (8 a 16 bits)
- Conversión sencilla vs continua
  - Poner en 1 el ADCO para conversión continua.



	DIFF	
MODE	0	1
0	Single ended 8-bit	Differential 9-bit 2's complement
1	Single ended 12-bit	Differential 13-bit 2's complement
2	Single ended 10-bit	Differential 11-bit 2's complement
3	Single ended 16-bit	Differential 16-bit 2's complement

Conversión completa

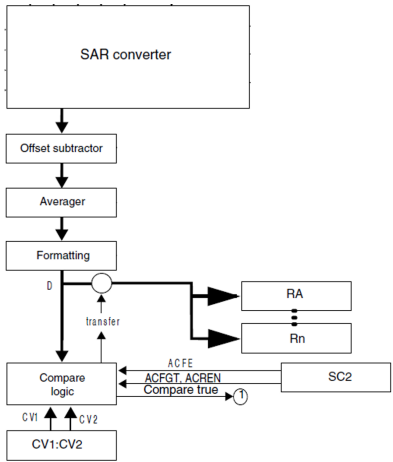




- Señalada en el bit COCO del SCIn
- Puede generar una interrupción por conversión completa si AIEN en el SCI esta configurado.
  - El handle del CMSIS para la interrupción se llama ADC0_IRQHandler

Registros del resultado

- Procesamiento adicional antes de ser almacenados en el registro de resultados
  - Resta de offset de la calibración.
  - Promediado: 1, 4, 8, 16 y 32 muestras.
  - Formateo: justificación a la derecha, extensión de signo de 16 bits
  - Comparación de salida
- Dos registros de resultados RA y Rn
  - Resultado de la conversión para al registro correspondiente del SCI usado para iniciar la conversión



Ejemplo - Lectura de un potenciometro

EL objetivo es utilizar un pin conversor de análogo a digital para hacer lectura del valor de voltaje de un potenciómetro.

Solución: primero se inicializa el ADC

```
1  #define ADC_CHANNEL (1)
2
3  void Init_ADC(void) {
4
5    SIM->SCGC6 |= SIM_SCGC6_ADC0_MASK;
6
7    // Low power configuration, long sample time, 16 bit single-ended
     conversion, bus clock input
8    ADC0->CFG1 = ADC_CFG1_ADLPC_MASK | ADC_CFG1_ADLSMP_MASK | ADC_CFG1_MODE(3)
      | ADC_CFG1_ADICLK(0);
9
10   // Software trigger, compare function disabled, DMA disabled, voltage
     references VREFH and VREFL
11   ADC0->SC2 = ADC_SC2_REFSEL(0);
12 }
```

Ejemplo - Lectura de un potenciometro

El main

```
1  int main(void) {
2
3    volatile static int i = 0 ;
4    float result;
5
6    Init_ADC();
7
8    while(1) {
9      ADC0->SC1[0] = ADC_CHANNEL;            /* start conversion on channel 0 */
10     while(!(ADC0->SC1[0] & ADC_SC1_COCO_MASK)) { } /* wait for conversion
       complete */
11     result = (float)ADC0->R[0];          /* read conversion result and clear
       COCO flag */
12     result = 3.3*(result/65536);
13
14     i++ ;
15     /* 'Dummy' NOP to allow source level single stepping of
16     tight while() loop */
17     __asm volatile ("nop");
18   }
19   return 0 ;
20 }
21
```

INTERFAZ ANÁLOGA

GRACIAS