Peripherals of Freescale Kinetis microcontrolers

Part 2

Analog Interfacing

Outline

- **Basics of analog to digital conversion**
- **KinetisL25 analog peripherals**
  - Digital to Analog Converter (DAC)
  - Comparator
  - Analog to Digital Converter (ADC)

CONVERTING BETWEEN ANALOG AND DIGITAL VALUES

The Big Picture – A Depth Gauge

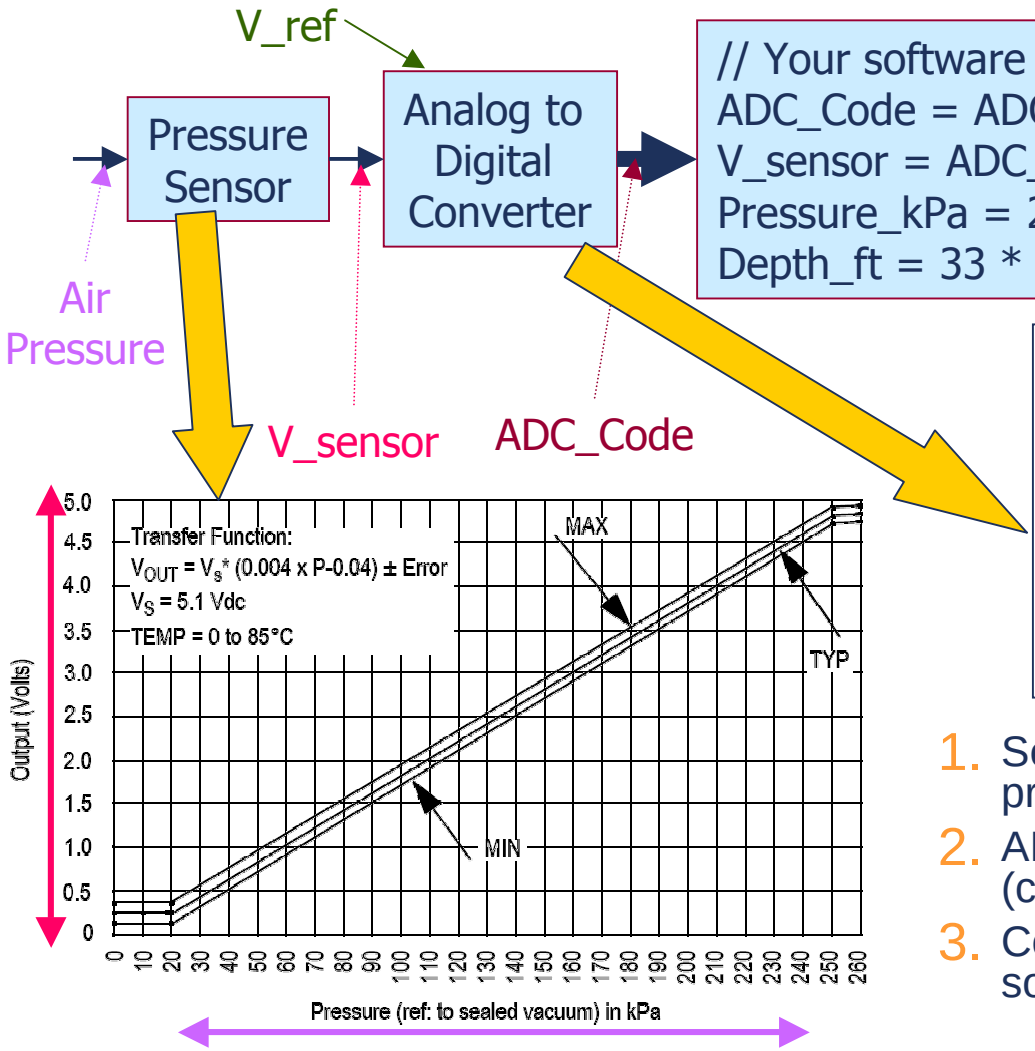


Figure 4. Output vs. Absolute Pressure

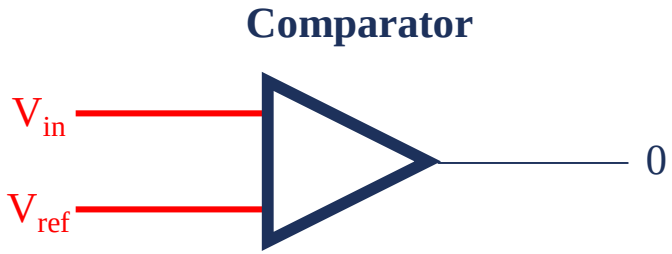
```
// Your software
ADC_Code = ADC0->R[0];
V_sensor = ADC_code * V_ref / 1023;
Pressure_kPa = 250 * (V_sensor/V_supply+0.04);
Depth_ft = 33 * (Pressure_kPa – Atmos_Press_kPa)/101.3;
```

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

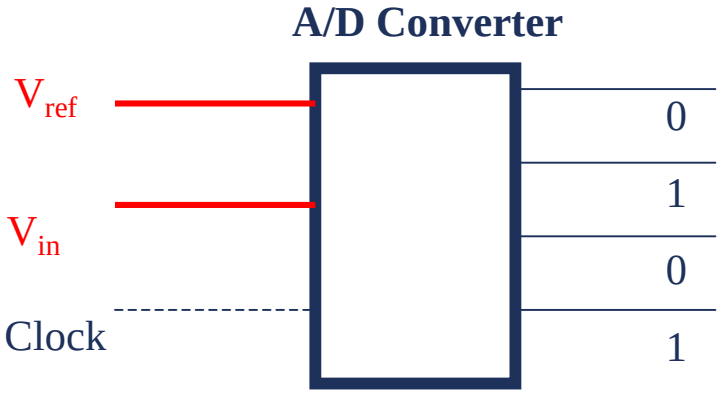
1. Sensor detects *air pressure* and generates a proportional *output voltage* V_sensor
2. ADC generates a proportional digital *integer* (code) based on V_sensor and V_ref
3. Code can convert that integer to a something more useful
    1. first a float representing the *voltage*,
    2. then another float representing *pressure*,
    3. finally another float representing *depth*

Getting From Analog to Digital

- **A Comparator tells us "Is $V_{in} > V_{ref}$?"**
  - Compares an **analog input voltage** with an **analog reference voltage** and determines which is larger, returning a 1-bit number
  - E.g. Indicate if depth > 100 ft
  - Set V_{ref} to voltage pressure sensor returns with 100 ft depth.

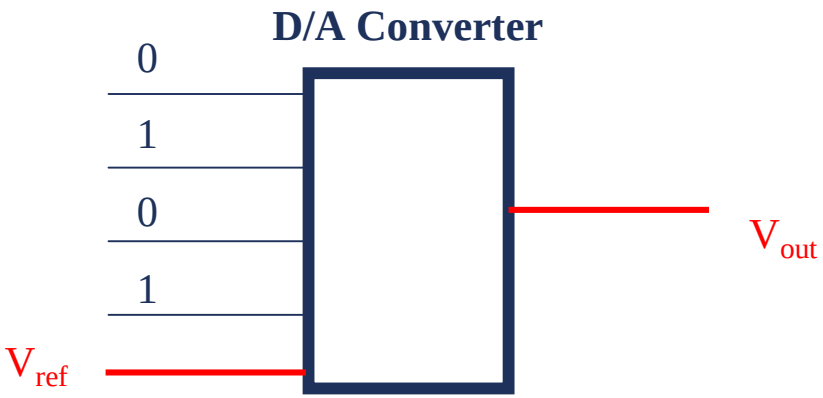


- **An Analog to Digital converter [AD or ADC] tells us how large V_{in} is as a fraction of V_{ref}.**
  - Reads an analog input signal (usually a voltage) and produces a corresponding multi-bit number at the output.
  - E.g. calculate the depth

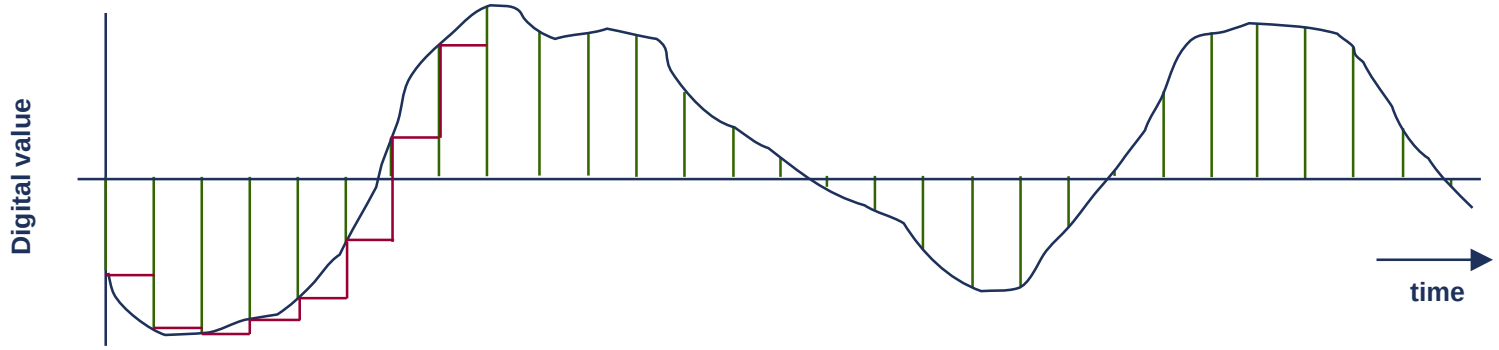


Digital to Analog Conversion

- **May need to generate an analog voltage or current as an output signal**
  - E.g. audio signal, video signal brightness.
- **DAC: "Generate the analog voltage which is this fraction of V_{ref}"**
- **Digital to Analog Converter equation**
  - *n* = input code
  - *N* = number of bits of resolution of converter
  - V_{ref} = reference voltage
  - V_{out} = output voltage. Either:
    - $V_{out} = V_{ref} * n/(2^N)$ *or*
    - $V_{out} = V_{ref} * (n+1)/(2^N)$
    - *The offset +1 term depends on the internal tap configuration of the DAC – check the datasheet to be sure*



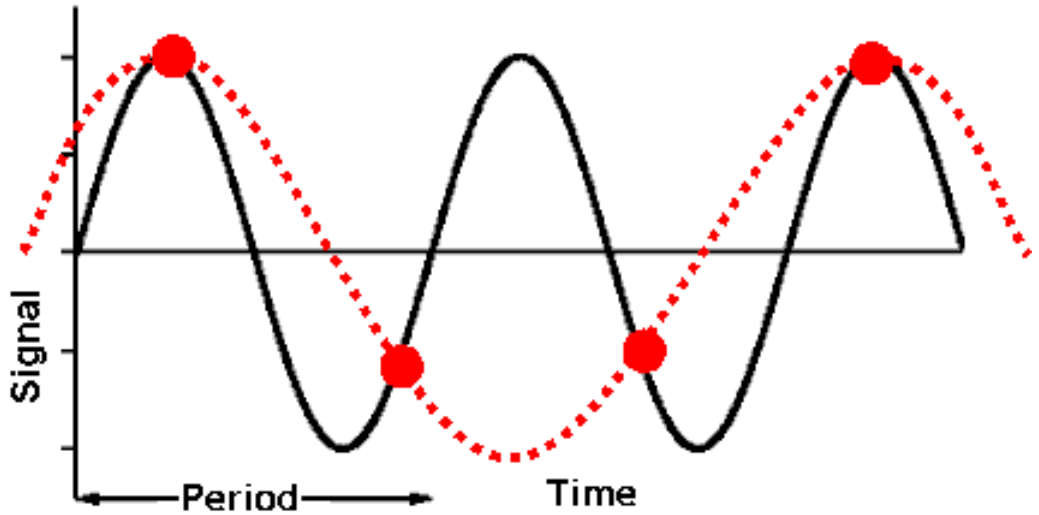
Waveform Sampling and Quantization



- **A waveform is sampled at a constant rate – every Δt**
  - Sampling converts a **continuous time** signal to a **discrete time** signal
- **The sample can now be quantized (converted) into a digital value**
  - Quantization represents a **continuous** (analog) value with the closest **discrete** (digital) value

Sampling problem example

- Consider if we sample a signal at 1.5 times it's period.
- In the graph below the black line shows the actual signal fluctuation, while the red dots show our samples.
- Observing our samples, we would infer that the signal was half it's actual frequency (dotted red line)



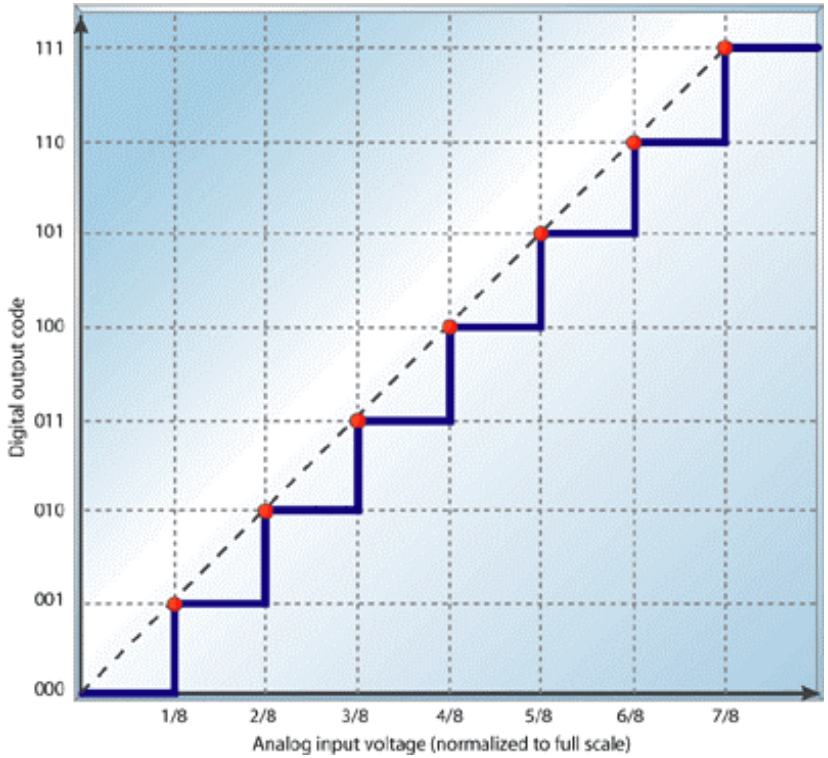
Nyquist criterion

- $F_{sample} >= 2 * F_{max\ frequency\ component}$
- **Frequency components above ½ F_{sample} are aliased, distort measured signal**

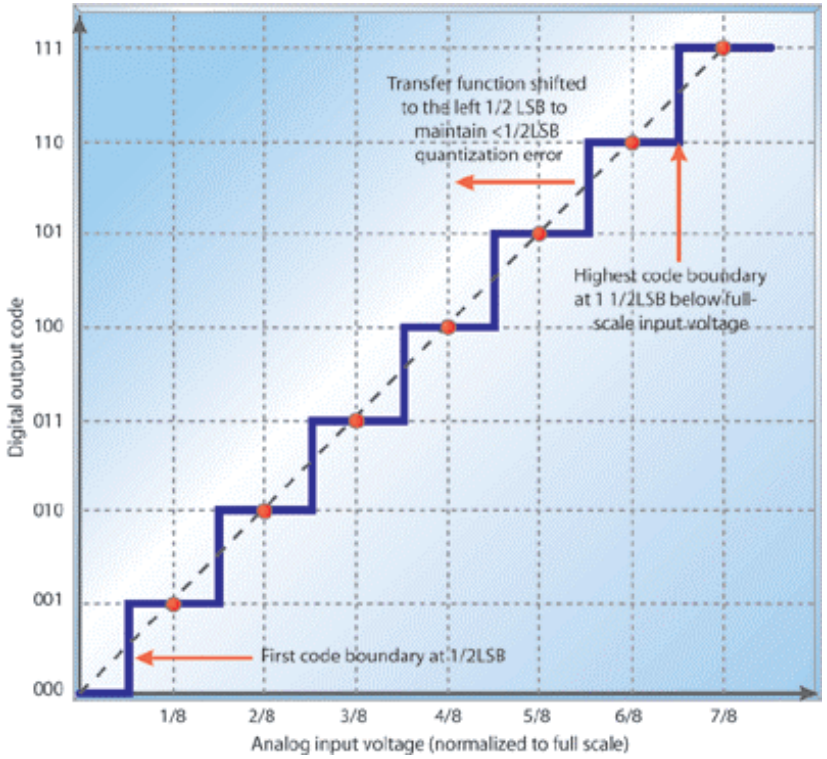
- **Nyquist and the real world**
  - This theorem assumes we have a perfect analog filter with “brick wall” roll-off
  - Real world filters have more gentle roll-off
  - Inexpensive filters are even worse (e.g. first order filter is 20 dB/decade, aka 6 dB/octave)
  - **So we have to choose a sampling frequency high enough that our filter attenuates aliasing components adequately**

Analog to Digital Transfer functions

- Ideal transfer function for a 3-bit ADC with reference points at code transition boundaries.
- Maximum quantization error is 1 LSB (Least Significant Bit) i.e. 1/8 V_{ref} for 3-bit ADC



- The transfer function implemented with an offset of - 1/2 LSB
- This shift of the transfer function to the left shifts the quantization error from a range of (- 1 to 0 LSB) to (- 1/2 to +1/2 LSB).



Forward Transfer Function Equations

What code n will the ADC use to represent voltage V_{in}?

n = converted code

V_{in} = sampled input voltage

V_{ref} = voltage reference

N = number of bits of resolution in ADC

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

$\lfloor X \rfloor = I$ *floor function: nearest integer I such that I <= X*

floor (x+0.5) rounds x to the nearest integer

Inverse Transfer Function

What range of voltages V_{in_min} to V_{in_max} does code n represent?

$$V_{in\_min} = \frac{n - \frac{1}{2}}{2^N} V_{ref}$$

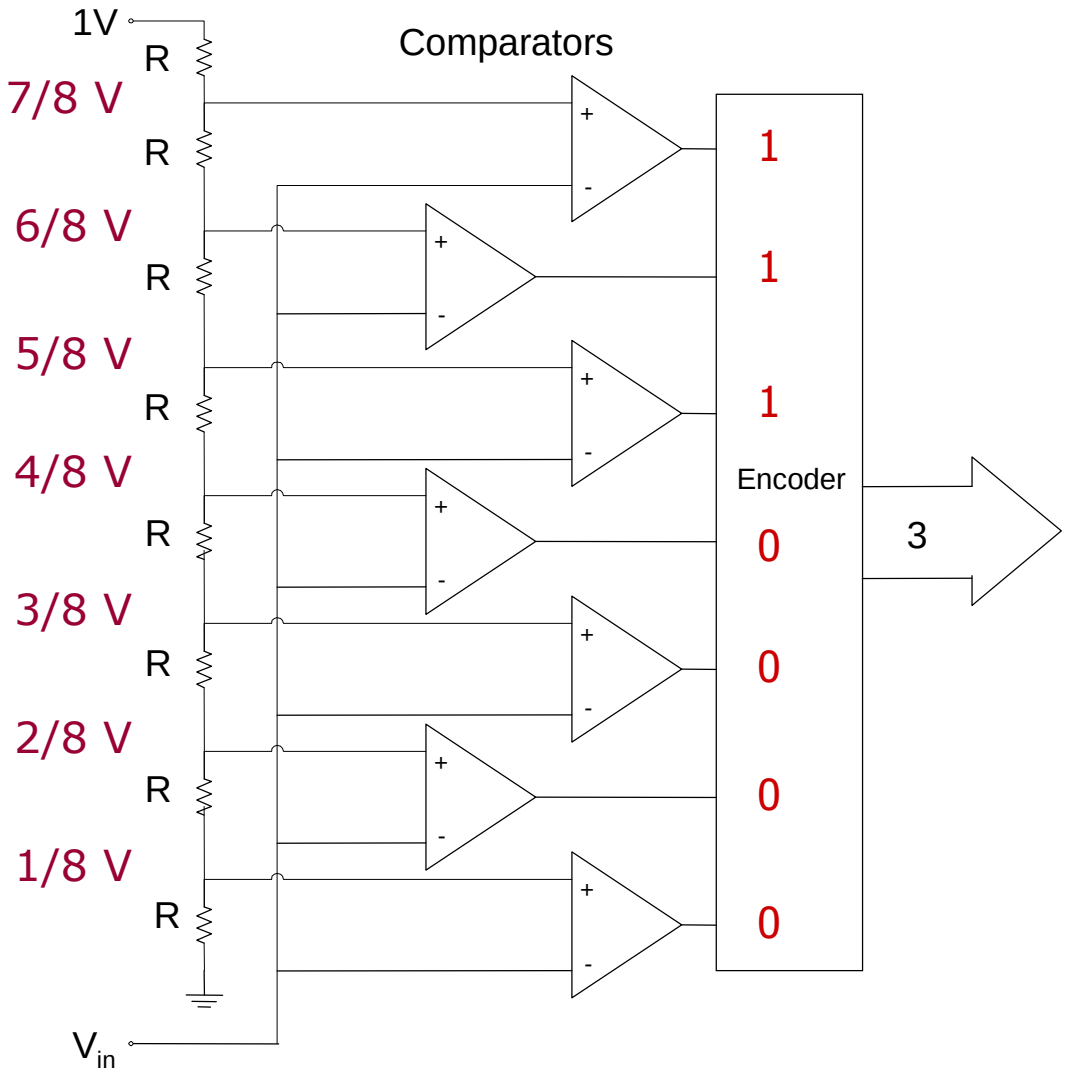
$$V_{in\_max} = \frac{n + \frac{1}{2}}{2^N} V_{ref}$$

$$V_{in\_avg} = \frac{n}{2^N} V_{ref}$$

ANALOG TO DIGITAL CONVERSION CONCEPTS

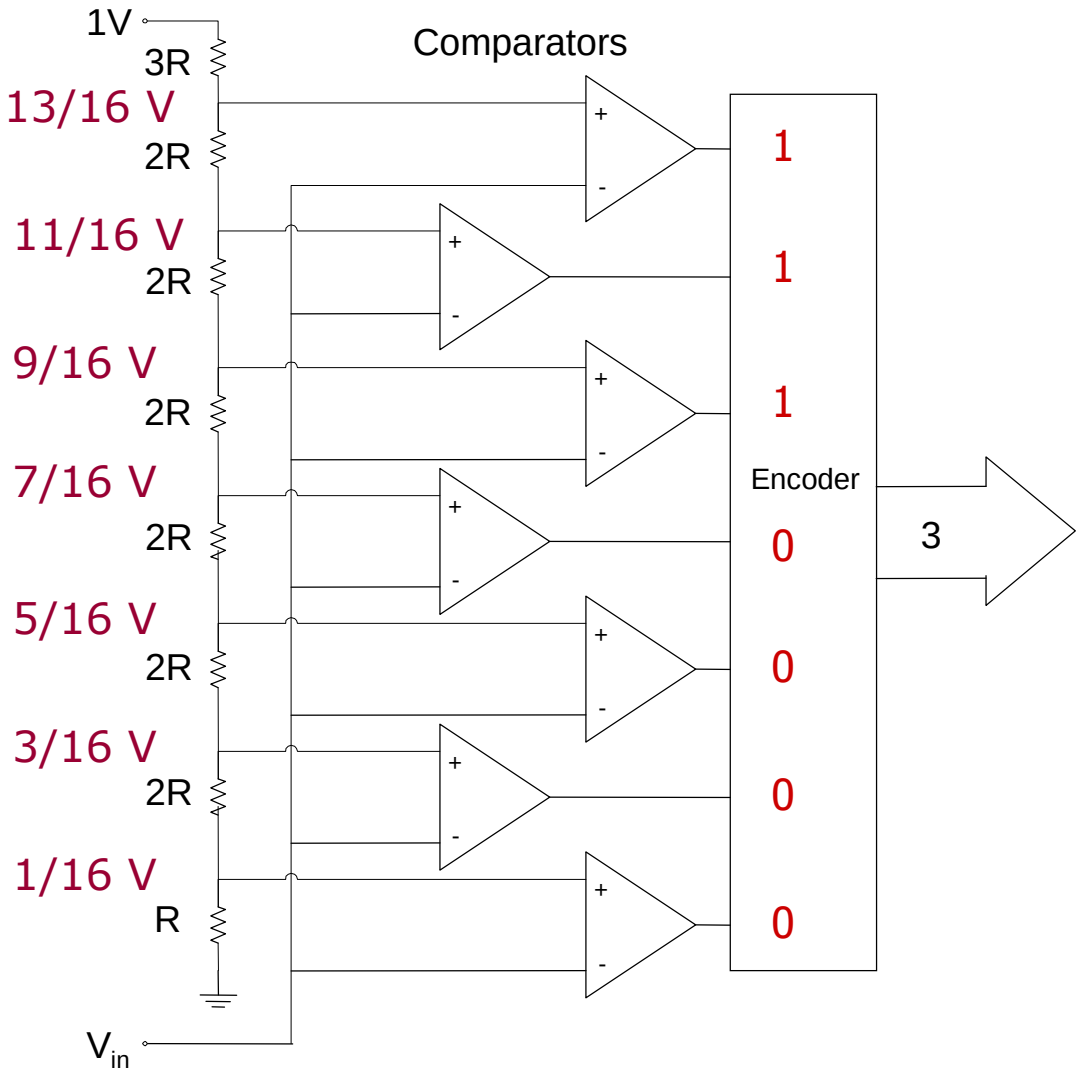
A/D – Flash Conversion

- **A multi-level voltage divider**
- **A comparator is used at each level**
- **The series of comparator outputs are encoded to a binary number in a priority encoder**
- **Components used**
  - 2^N resistors
  - 2^N-1 comparators
- **Note**
  - This particular resistor divider generates voltages which are *not* offset by ½ bit, so maximum error is 1 bit



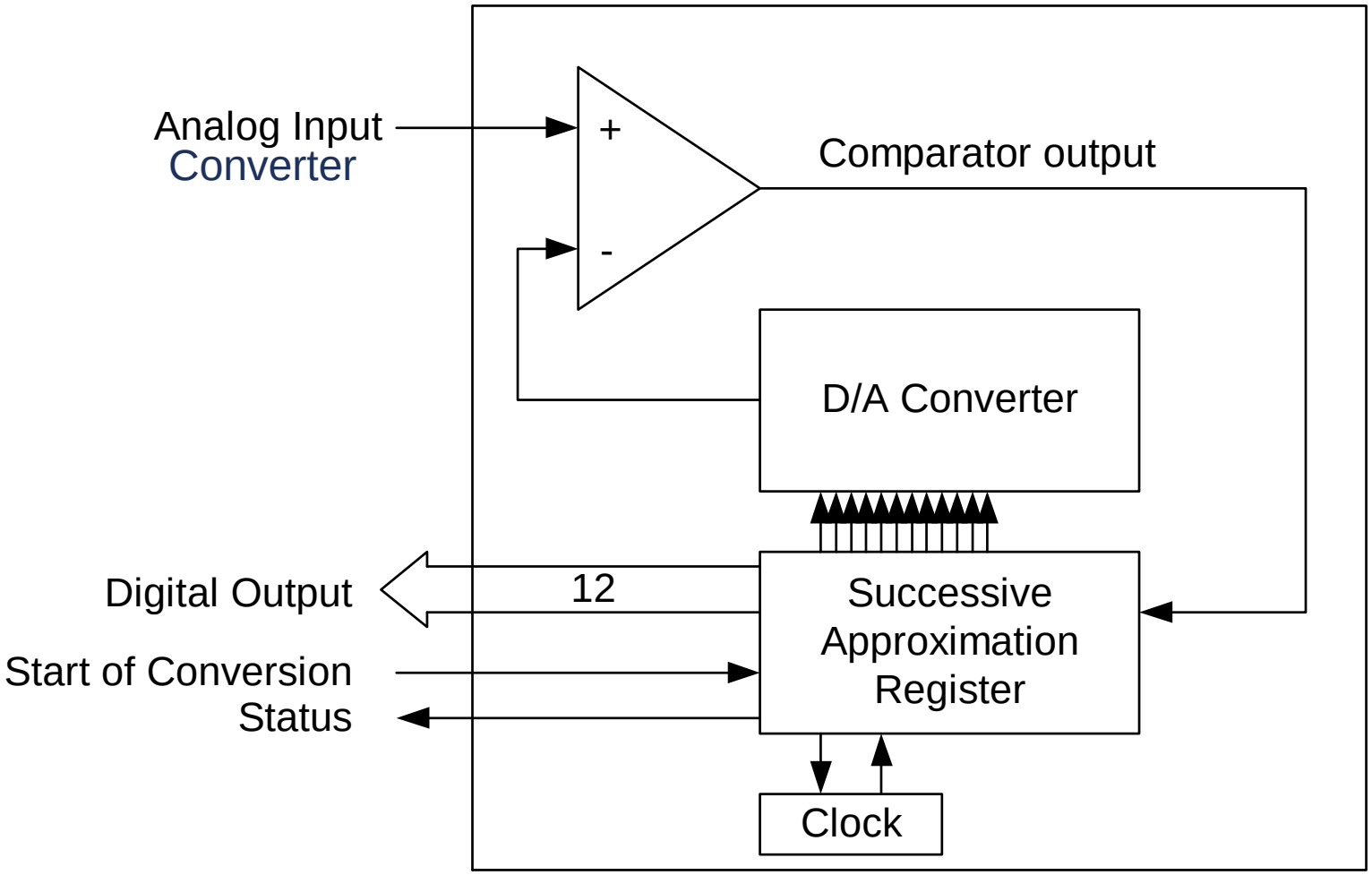
A/D – Flash Conversion

- **A multi-level voltage divider**
- **A comparator is used at each level**
- **The series of comparator outputs are encoded to a binary number in a priority encoder**
- **Components used**
  - 2^N resistors
  - 2^N-1 comparators
- **Note**
  - This particular resistor divider generates voltages which are *not* offset by ½ bit, so maximum error is 1 bit
  - We could change this offset voltage by using resistors of values R, 2R, 2R ... 2R, 3R (starting at bottom)



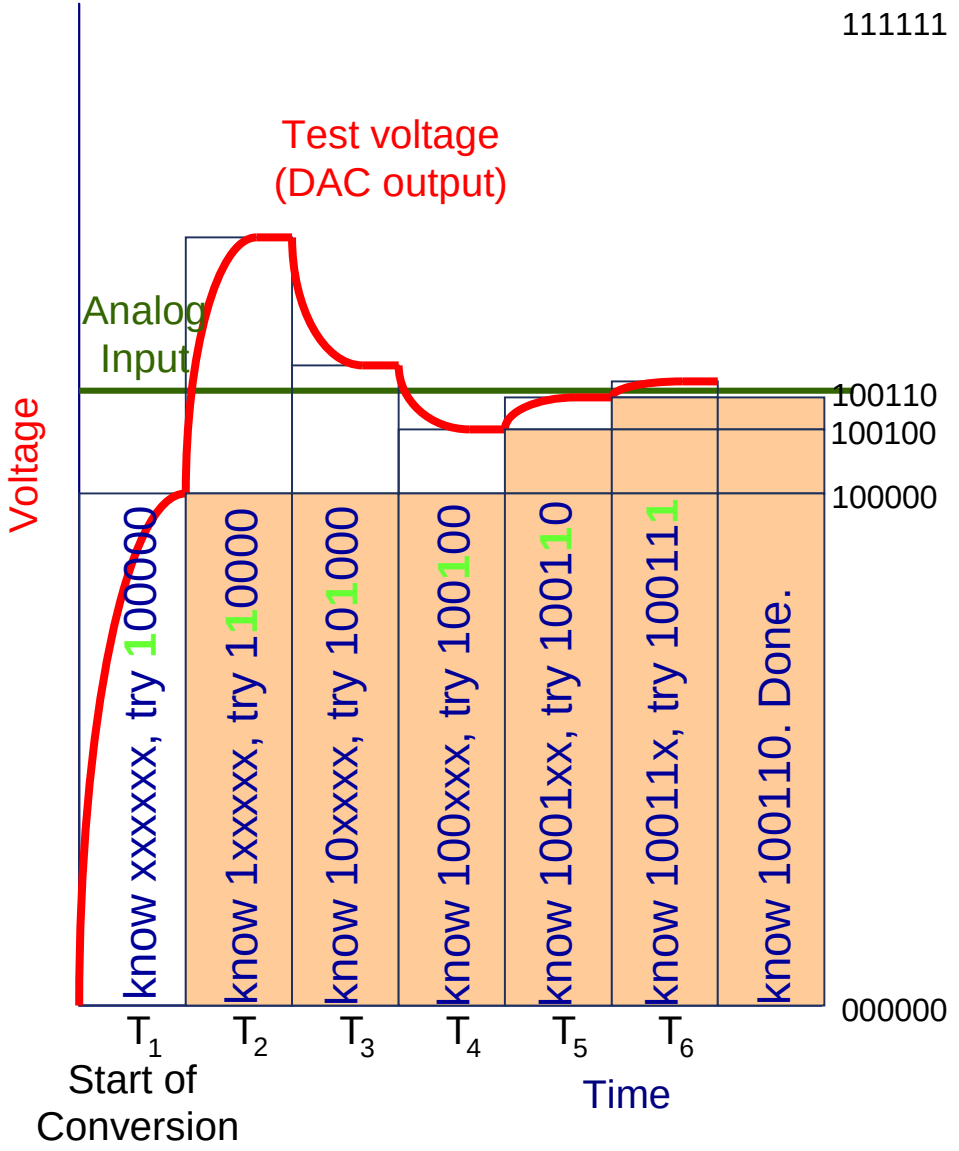
A/D - Successive Approximation

Converter Schematic



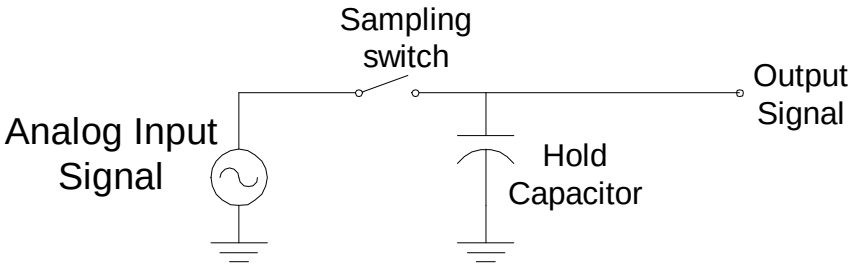
ADC - Successive Approximation Conversion

- **Successively approximate input voltage by using a binary search and a DAC**
- **SA Register holds current approximation of result**
- **Set all DAC input bits to 0**
- **Start with DAC's most significant bit**
- **Repeat**
  - Set next input bit for DAC to 1
  - Wait for DAC and comparator to stabilize
  - If the DAC output (test voltage) is **smaller** than the input then set the current bit to 1, else clear the current bit to 0



Sample and Hold Devices

- AD conversion takes time (e.g. successive approximation devices)
- So A/D converters require the input analog signal to be held stable during conversion,
- In other cases, peak capture or sampling at a specific point in time necessitates a **sampling device**.
- This function is accomplished by a sample and hold (S&H) device as shown to the right:
- These devices are incorporated into some A/D converters



KL25 ANALOG INTERFACING PERIPHERALS

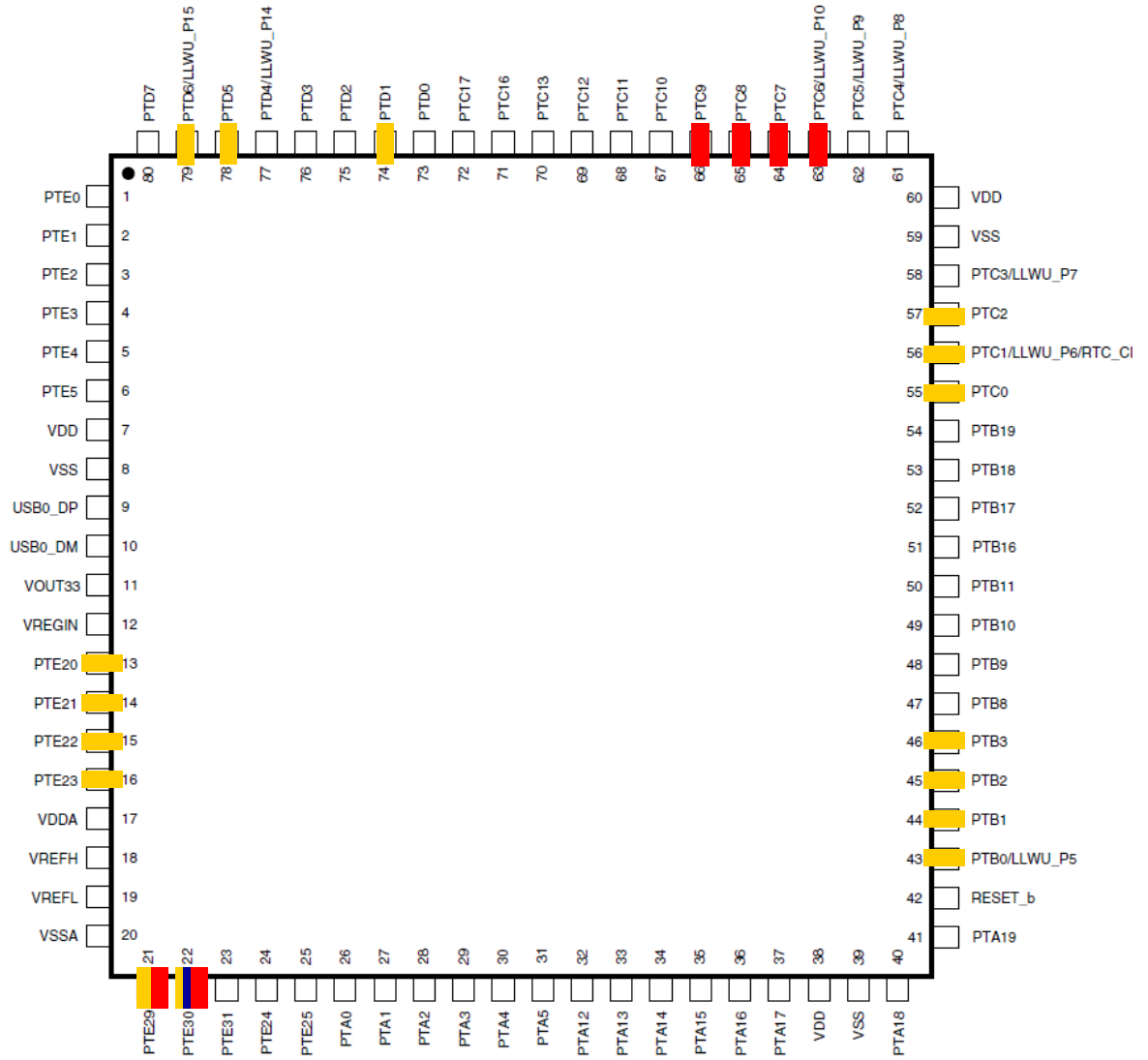
Sources of Information

- **KL25 Subfamily Reference Manual (Rev. 1, June 2012)**
  - Describes architecture of peripherals and their control registers
  - Digital to Analog Converter
    - Chapter 30 of KL25 Subfamily Reference Manual
  - Analog Comparator
    - Chapter 29 of KL25 Subfamily Reference Manual
  - Analog to Digital Converter
    - Chapter 28 of KL25 Subfamily Reference Manual

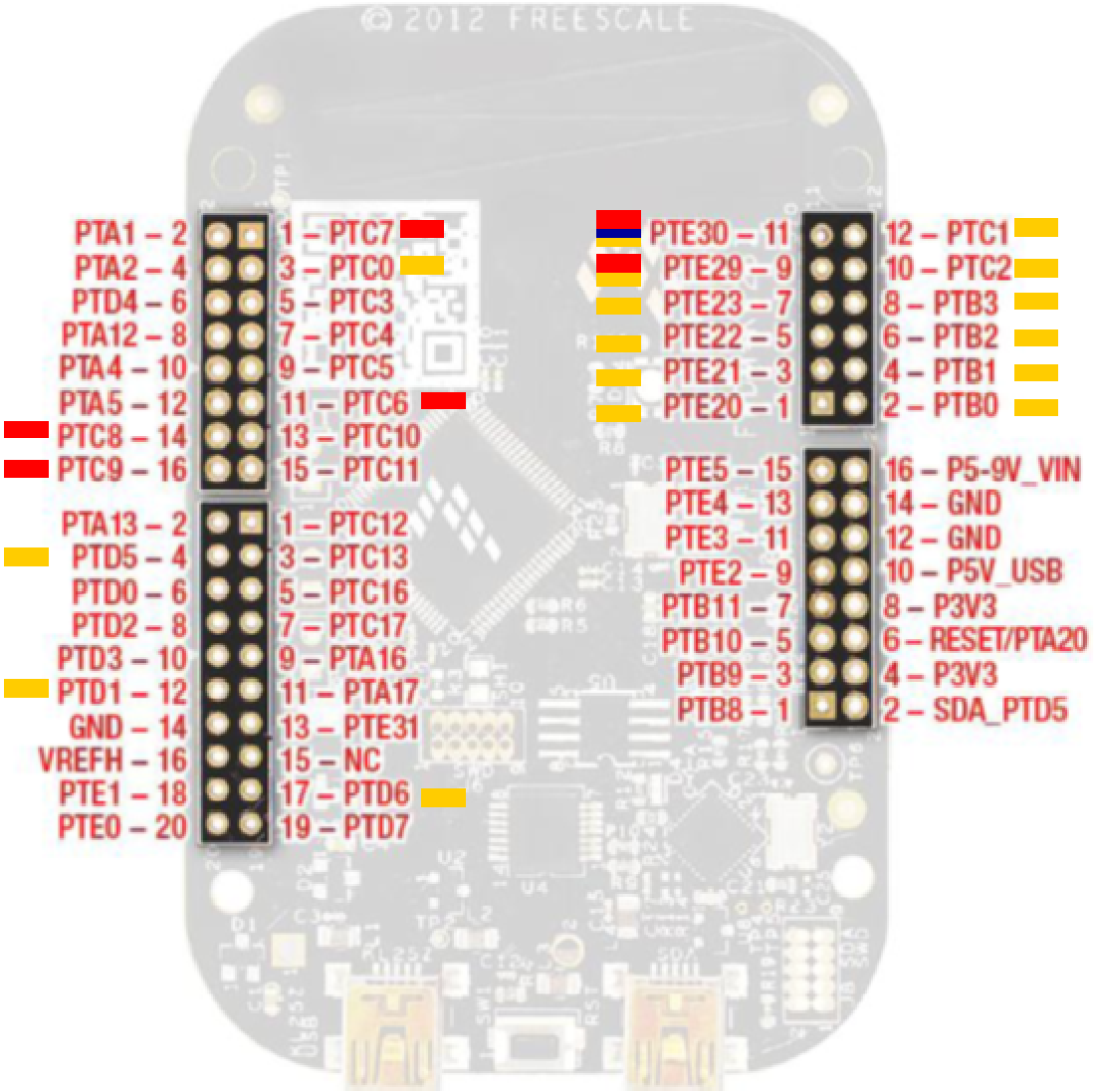
- **KL25 Sub-family Data Sheet (Rev. 3, 9/19/2012)**
  - Describes circuit-specific performance parameters: operating voltages, min/max speeds, cycle times, delays, power and energy use
- **Kinetis L Peripheral Module Quick Reference, Freescale Semiconductor**
  - Gives the valuable examples of programme codes

KL25Z Analog Interface Pins

- **80-pin QFP**
- **Inputs**
  - One 16-bit ADC with 14 input channels
  - One comparator with 6 external inputs and one 6-bit DAC
- **Output**
  - One 12-bit DAC



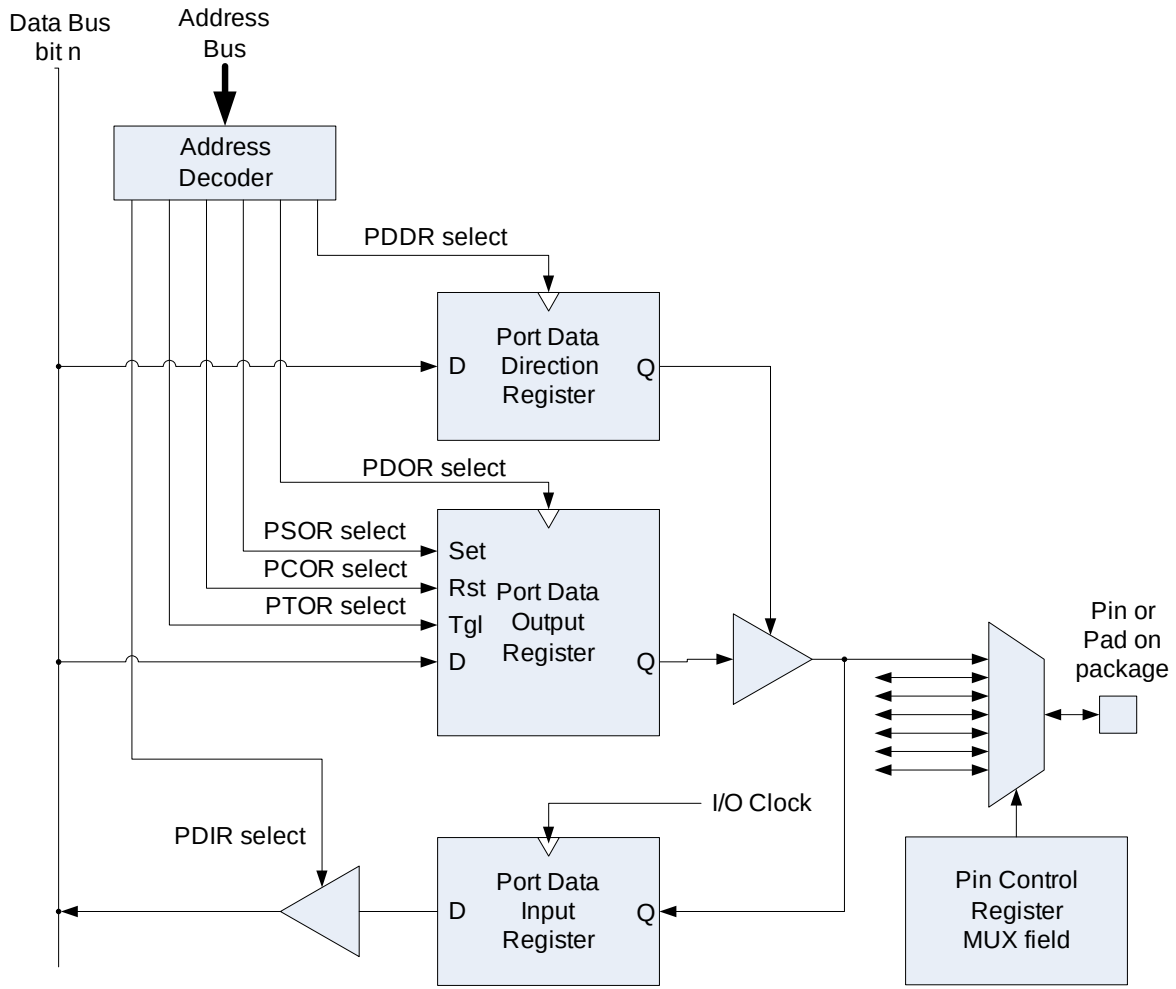
Freedom KL25Z Analog I/O



- Inputs
  - 14 external ADC channels
  - 6 external comparator channels
- Output
  - 1 12-bit DAC

GPIO Port Bit Circuitry in MCU

- **Configuration**
  - Direction
  - **MUX**
- **Data**
  - Output (different ways to access it)
  - Input



Pin Control Register to Select MUX Channel

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

80 LQFP	64 LQFP	48 QFN	32 QFN	Pin Name	Default	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7
64	52	40	28	PTC7	CMP0_IN1	CMP0_IN1	PTC7	SPI0_MISO			SPI0_MOSI		
65	53	—	—	PTC8	CMP0_IN2	CMP0_IN2	PTC8	I2C0_SCL	TPM0_CH4				

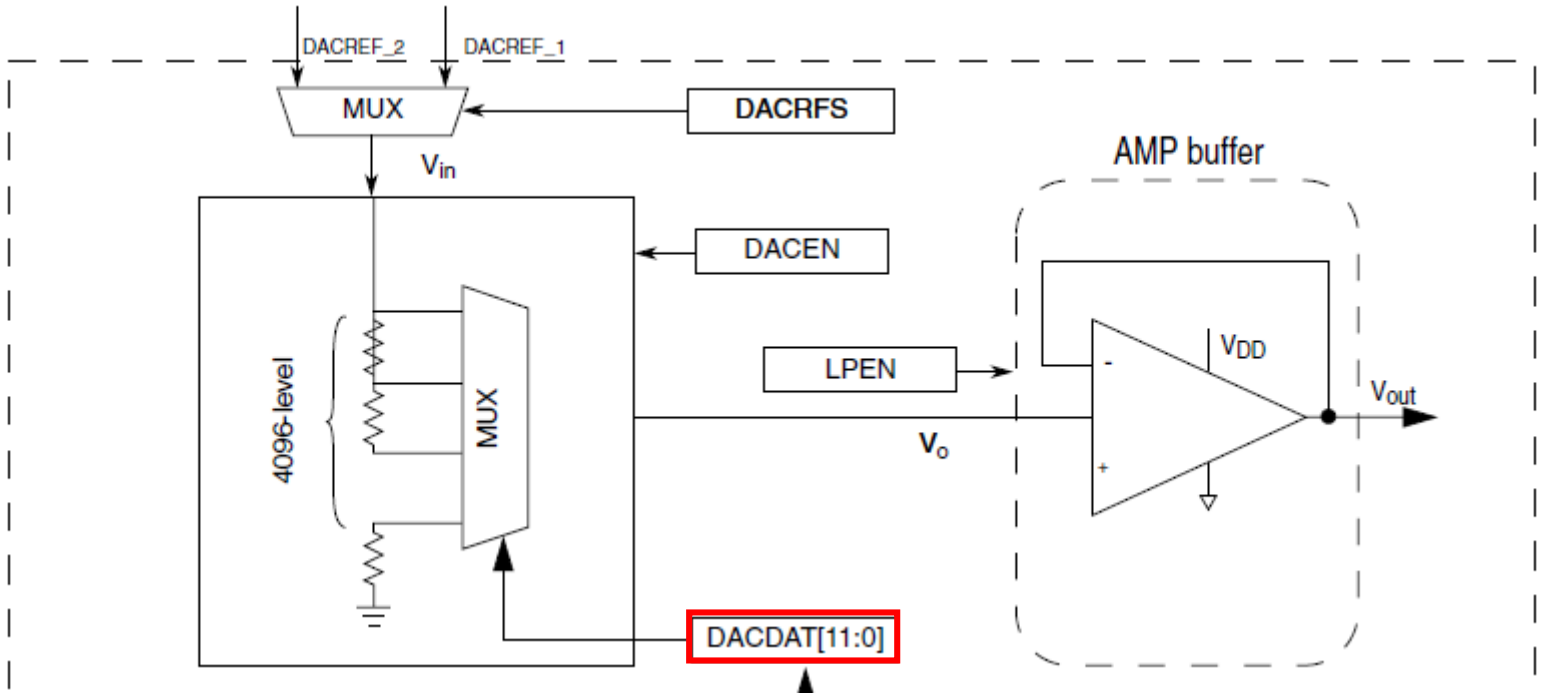
- **MUX field of PCR defines connections**

MUX (bits 10-8)	Configuration
000	Pin disabled (analog)
001	Alternative 1 – GPIO
010	Alternative 2
011	Alternative 3
100	Alternative 4
101	Alternative 5
110	Alternative 6
111	Alternative 7

```
PORTC->PCR[7] &= ~PORT_PCR_MUX_MASK;
PORTC->PCR[7] |= PORT_PCR_MUX(0);
```

DIGITAL TO ANALOG CONVERTER

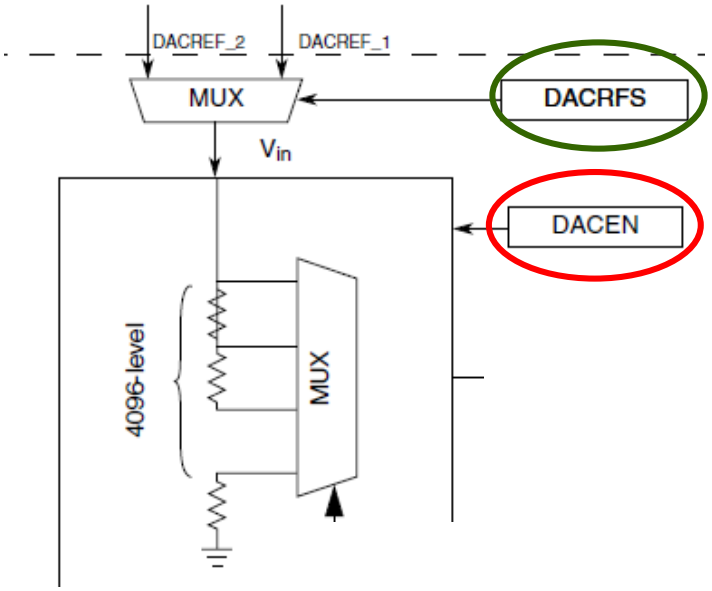
DAC Overview



- **Load DACDAT with 12-bit data N**
- **MUX selects a node from resistor divider network to create $V_o = (N+1)*V_{in}/2^{12}$**
- **V_o is buffered by output amplifier to create V_{out}**
  - $V_o = V_{out}$ but V_o is high impedance - can't drive much of a load, so need to buffer it

DAC Control Register 0: DACx_C0

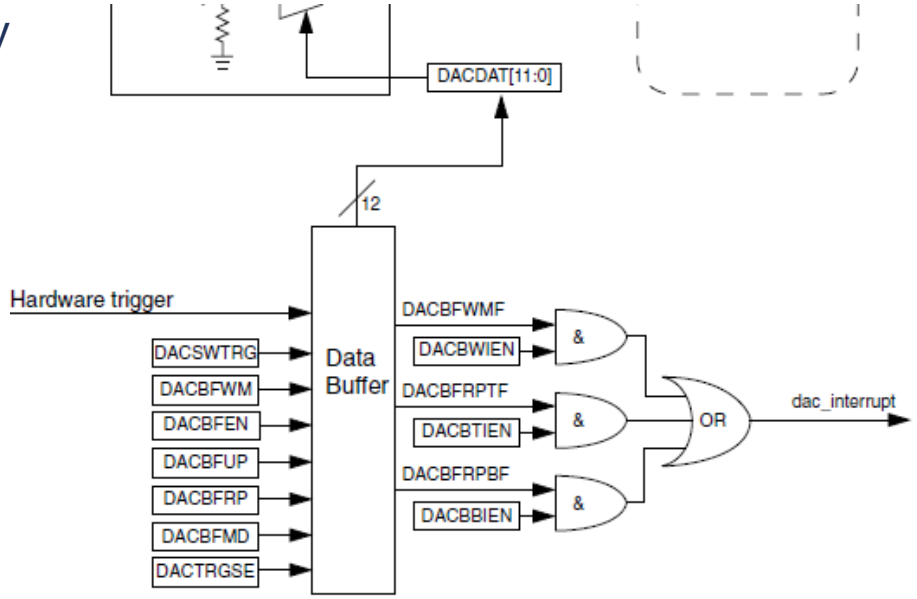
Bit	7	6	5	4	3	2	1	0
Read	DACEN	DACRFS	DACTRGSEL	0	LPEN	0	DACBTIEN	DACBBIEN
Write				DACSWTRG				
Reset	0	0	0	0	0	0	0	0



- **DACEN** - DAC Enabled when 1
- **DACRFS** - DAC reference voltage select
  - 0: DACREF_1. Connected to VREFH
  - 1: DACREF_2. Connected to VDDA
- **LPEN** - low-power mode
  - 0: High-speed mode. Fast (15 us settling time) but uses more power (up to 900 uA supply current)
  - 1: Low-power mode. Slow (100 us settling time) but more power-efficient (up to 250 uA supply current)
- Additional control registers used for buffered mode

DAC Operating Modes

- **Normal**
  - DAT0 is converted to voltage immediately
- **Buffered**
  - Data to output is stored in 16-word buffer
  - Next data item is sent to DAC when a selectable trigger event occurs
    - Software Trigger - write to DACSWTRG field in DACx_C0
    - Hardware Trigger - from PIT timer peripheral



- **DACBFEN -** DAC Buffer enable
- **DACSWTRG -** DAC Software Trigger
- **DACTRGSE -** DAC Trigger Select (hw/sw)

DAC Data Registers

- DAC registers are only eight bits long

- DATA[11:0] stored in two registers
  - DATA0: Low byte [7:0] in DACx_DATnL
  - DATA1: High nibble [11:0] in DACx_DATnH
    - Nibble is a half of byte

Example: Waveform Generator

Pseudo algorithm:

- **Supply clock to DAC0 module**
  - Set bit 31 of SIM SCGC6 register
- **Set Pin Mux to Analog (0)**
- **Enable DAC**
- **Configure DAC**
  - Reference voltage
  - Low power mode if necessary
  - Normal mode (not buffered)
- **Write to DAC data register**

Init DAC. Example Code

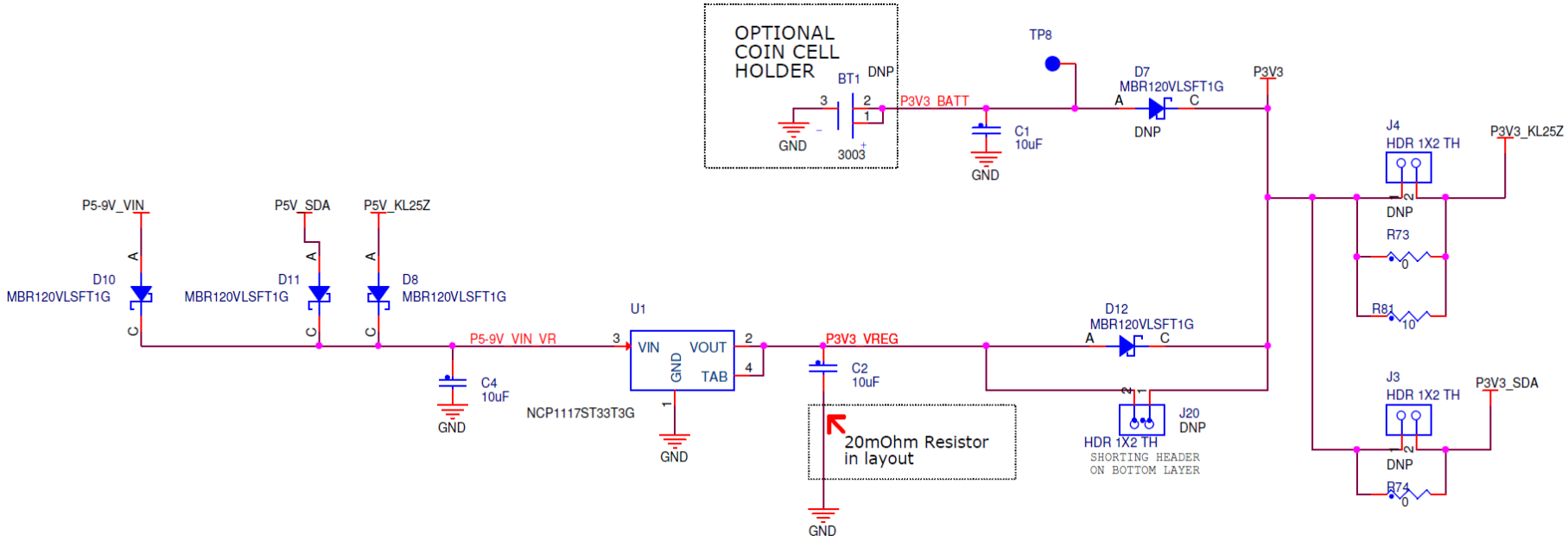
```
void Init_DAC(void) {

  /* Init DAC output */
  SIM->SCGC6 |= (1UL << SIM_SCGC6_DAC0_SHIFT);
  SIM->SCGC5 |= (1UL << SIM_SCGC5_PORTE_SHIFT);

  /* Select analog */
  PORTE->PCR[DAC_POS] &= ~(PORT_PCR_MUX(7))

  /* Disable buffer mode */
  DAC0->C1 = 0;
  DAC0->C2 = 0;

  /* Enable DAC, select VDDA as reference voltage */
  DAC0->C0 = (1 << DAC_C0_DACEN_SHIFT) |
             (1 << DAC_C0_DACRFS_SHIFT);

}
```

Sinus Waveform Generator Example

```
extern unsigned SineTable[]; /* Sinus values look-up table */

void SinGen(unsigned int period, unsigned int num_cycles ) {
  unsigned step, out_data;

  while (num_cycles>0) {
      num_cycles--;
      for (step = 0; step < NUM_STEPS; step++) {
            out_data = SineTable[step];
            /* Put next sin() value in DATA register */
            DAC0->DAT[0].DATH = DAC_DATH_DATA1(out_data >> 8);
            DAC0->DAT[0].DATL = DAC_DATL_DATA0(out_data);

            Delay_us(period/NUM_STEPS);
      }
  }
}
```

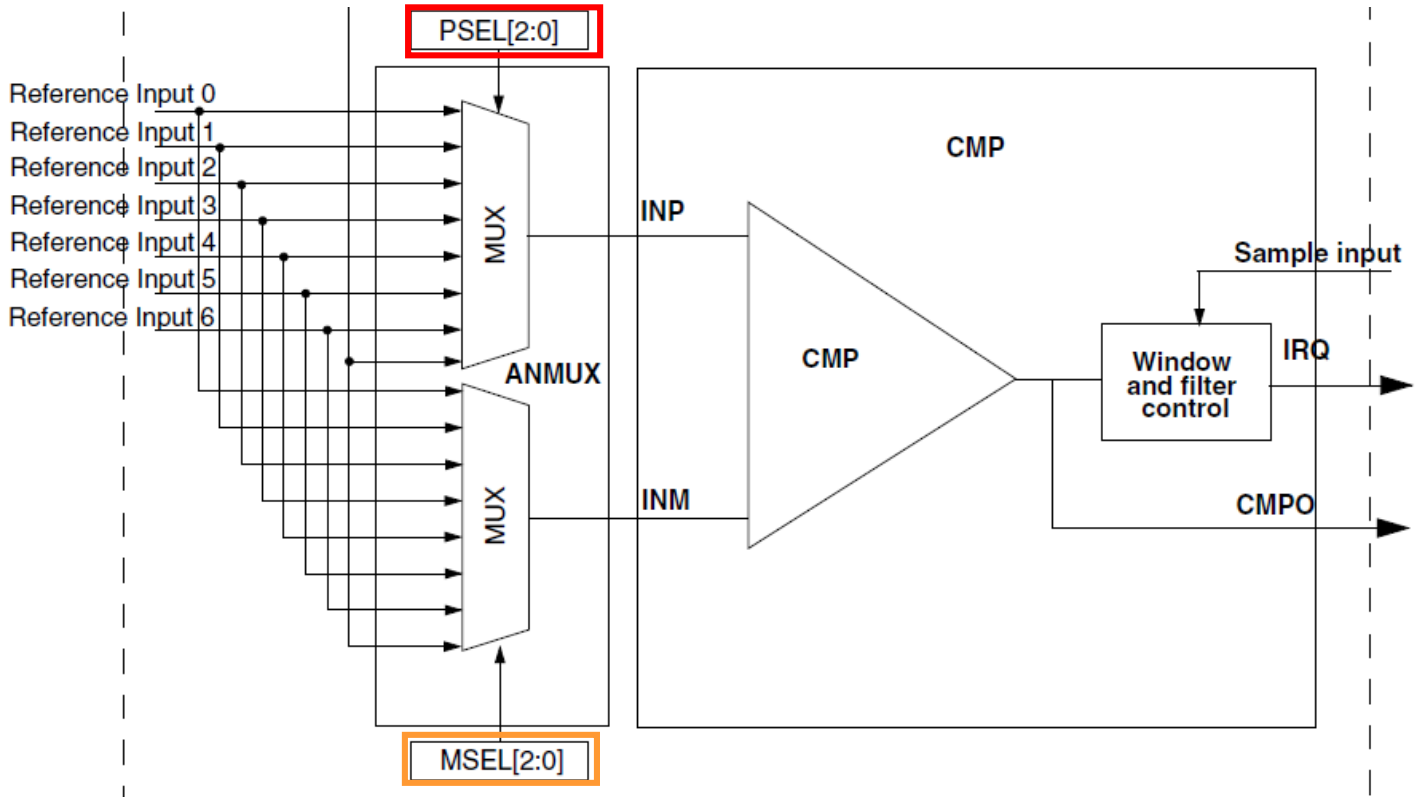
ANALOG COMPARATOR

Example: Power Failure Detection



- Need warning of when power has failed
  - Very limited amount of time before capacitor C2 discharges
  - Save critical information
  - Turn off output devices
  - Put system into safe mode
- Can use a comparator to compare V_{in} against a fixed reference voltage V_{Ref}

Comparator Overview



- Comparator compares INP and INM
- CMPO Output indicates if INP>INM (1) or INP<INM (0)
- Can generate an interrupt request (+, -, or +- edges)
- ANMUX selection of one of multiple reference inputs, using **PSEL** and **MSEL** fields

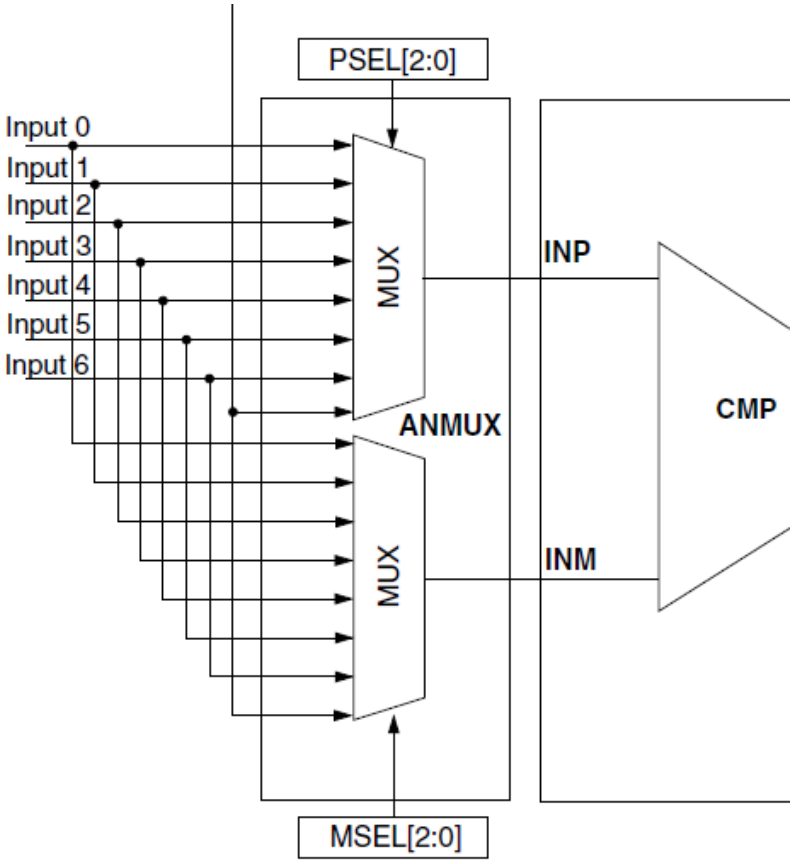
CMP Control Register 1 CMPx_CR1

Bit	7	6	5	4	3	2	1	0
Read Write	SE	WE	TRIGM	PMODE	INV	COS	OPE	EN
Reset	0	0	0	0	0	0	0	0

- **EN: Module enable (1)**
- **OPE: Output Pin Enable**
  - 1: connects comparator output signal CMPO to output pin
- **PMODE: Power Mode Select**
  - 0: Low speed
  - 1: High speed

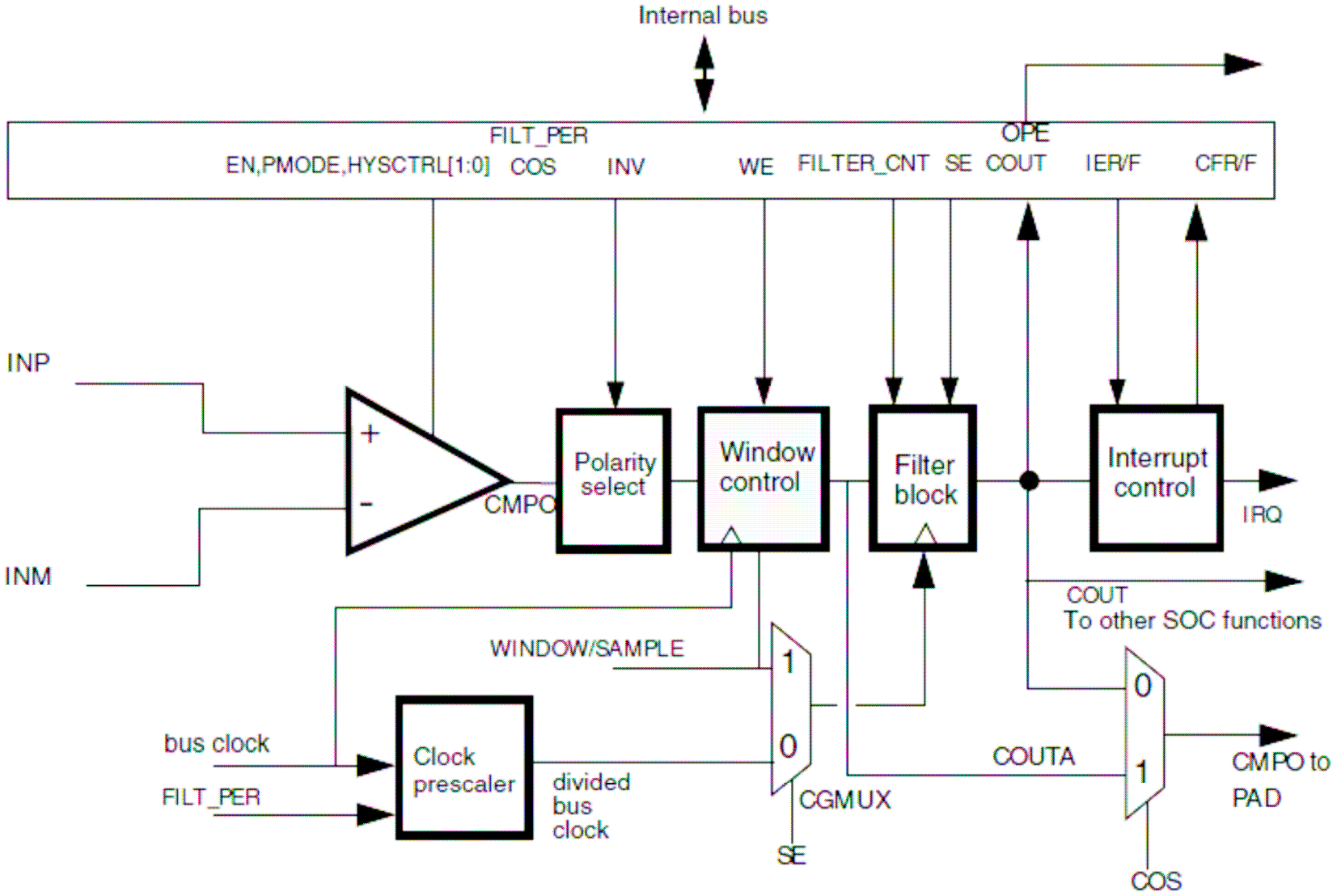
MUX Control Register CMPx_MUXCR

Bit	7	6	5	4	3	2	1	0
Read	0	PSTM	PSEL			MSEL		
Write								
Reset	0	0	0	0	0	0	0	0

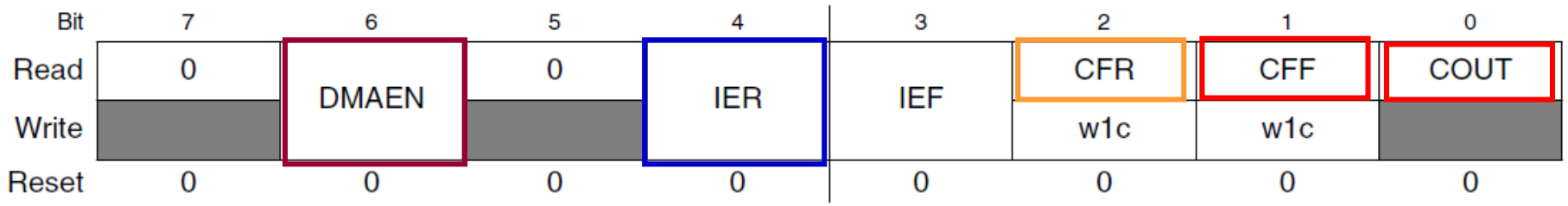


- **PSEL: Plus Input Mux Control**
  - Selects which input (IN0-7) goes to the comparator's + input
- **MSEL: Minus Input Mux Control**
  - Selects which input (IN0-7) goes to the comparator's - input

Comparator Output Processing



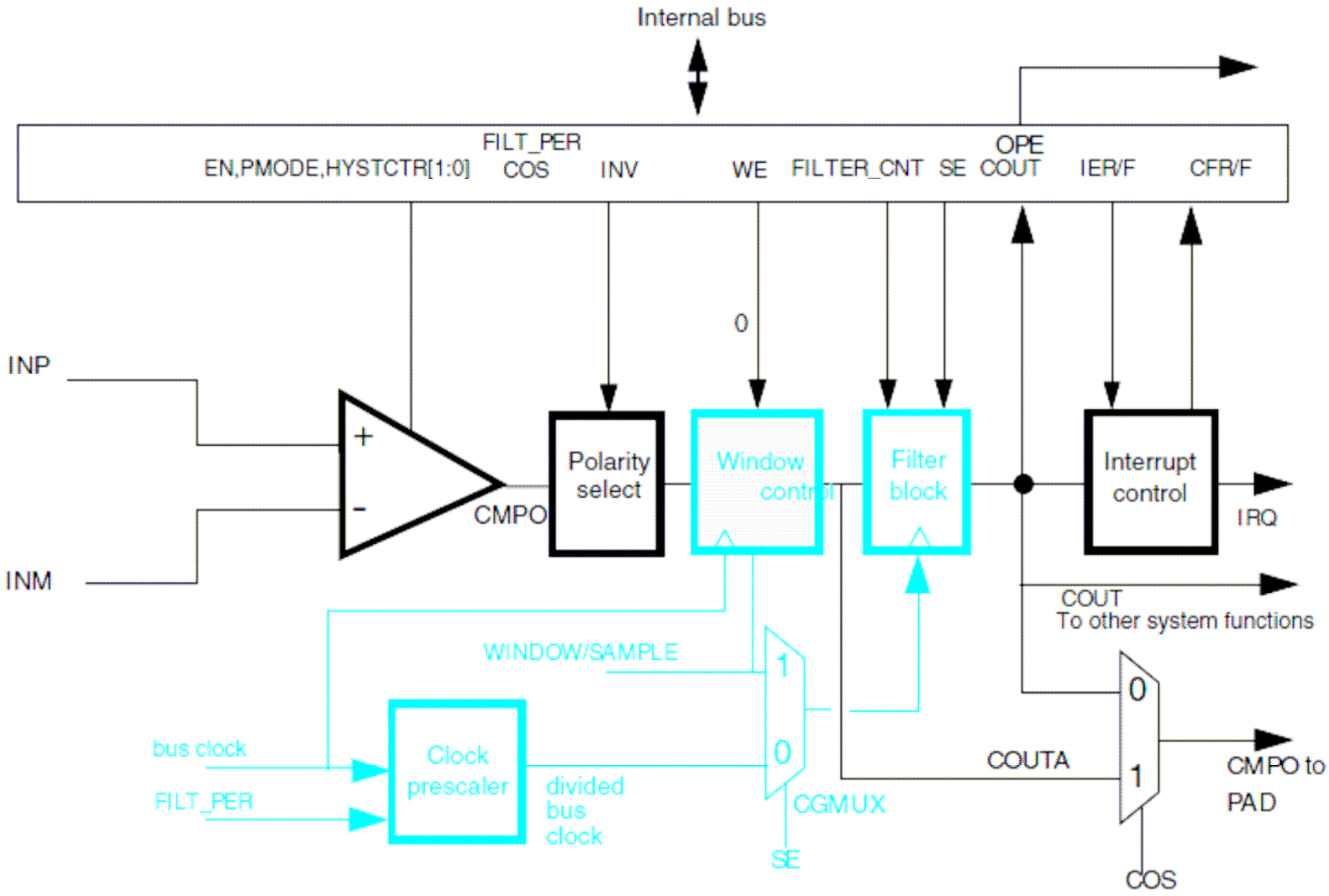
Comparator Interrupt



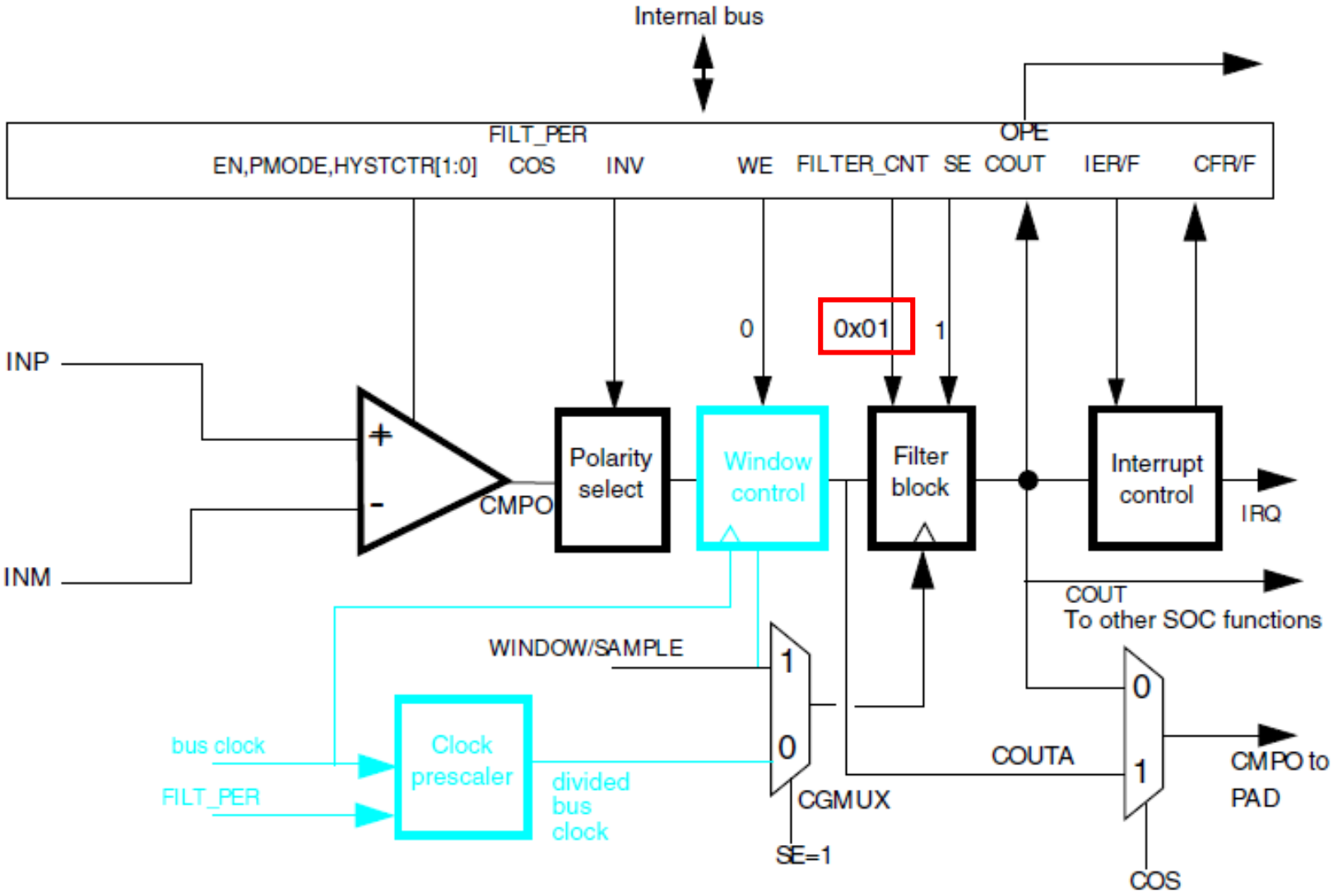
Bit	7	6	5	4	3	2	1	0
Read	0	DMAEN	0	IER	IEF	CFR	CFF	COUT
Write						w1c	w1c	
Reset	0	0	0	0	0	0	0	0

- **CMPx_SCR: Status and control register**
  - **COUT**: output of comparator
  - **CFR**: Comparator flag rising. Rising edge detected on comparator output COUT. Clear flag by writing with a 1.
  - **CFF**: Comparator flag falling. Falling edge detected on comparator output COUT. Clear flag by writing with a 1.
  - **IER**: 1 enables interrupt when CFR is set.
  - **IEF**: 1 enables interrupt when CFF is set.
- **Can generate interrupt on matching edge**
  - CMSIS-defined ISR name for Comparator Interrupt is **CMP0_IRQHandler**

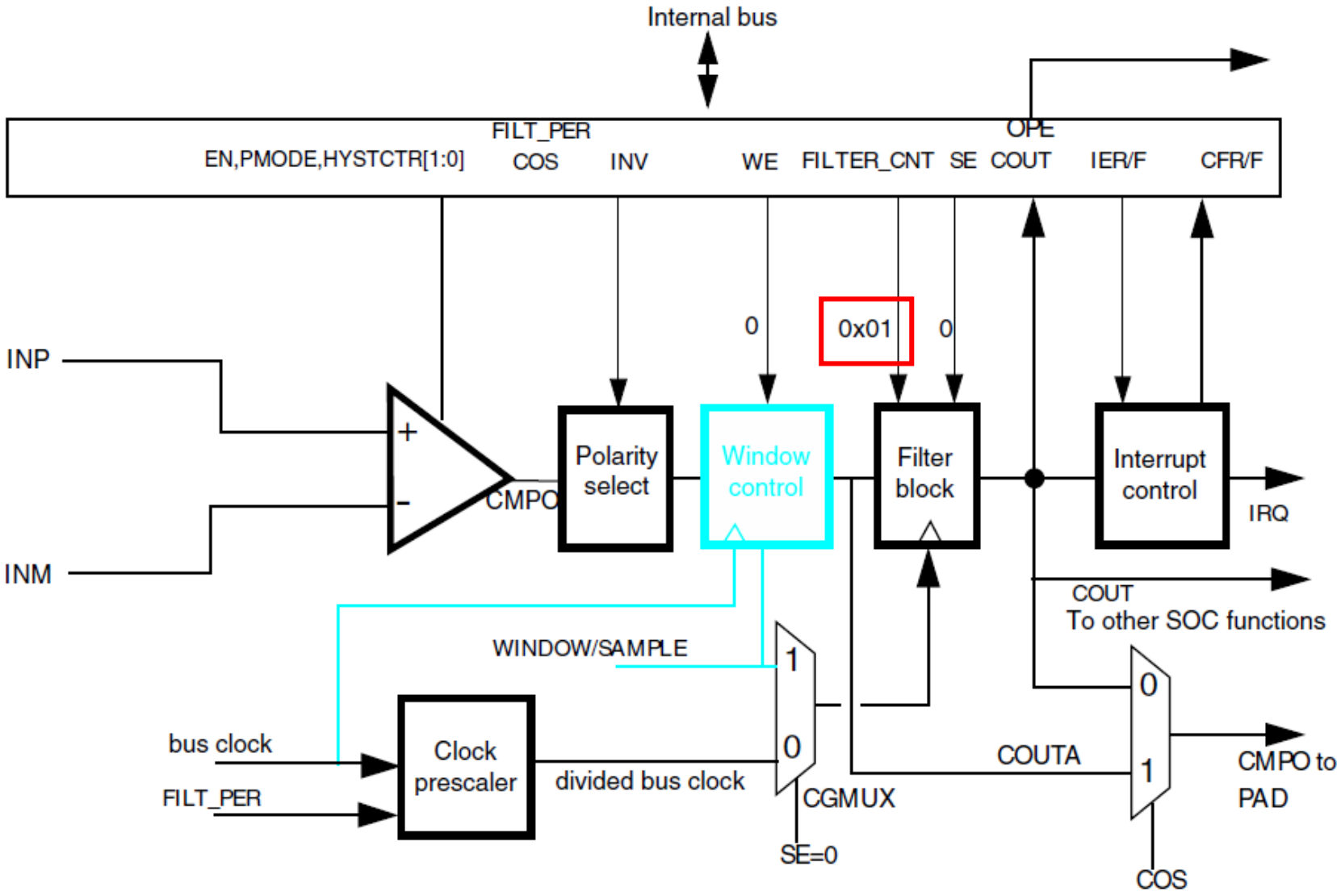
Continous Mode. SE=0 WE=0 F_CNT=0



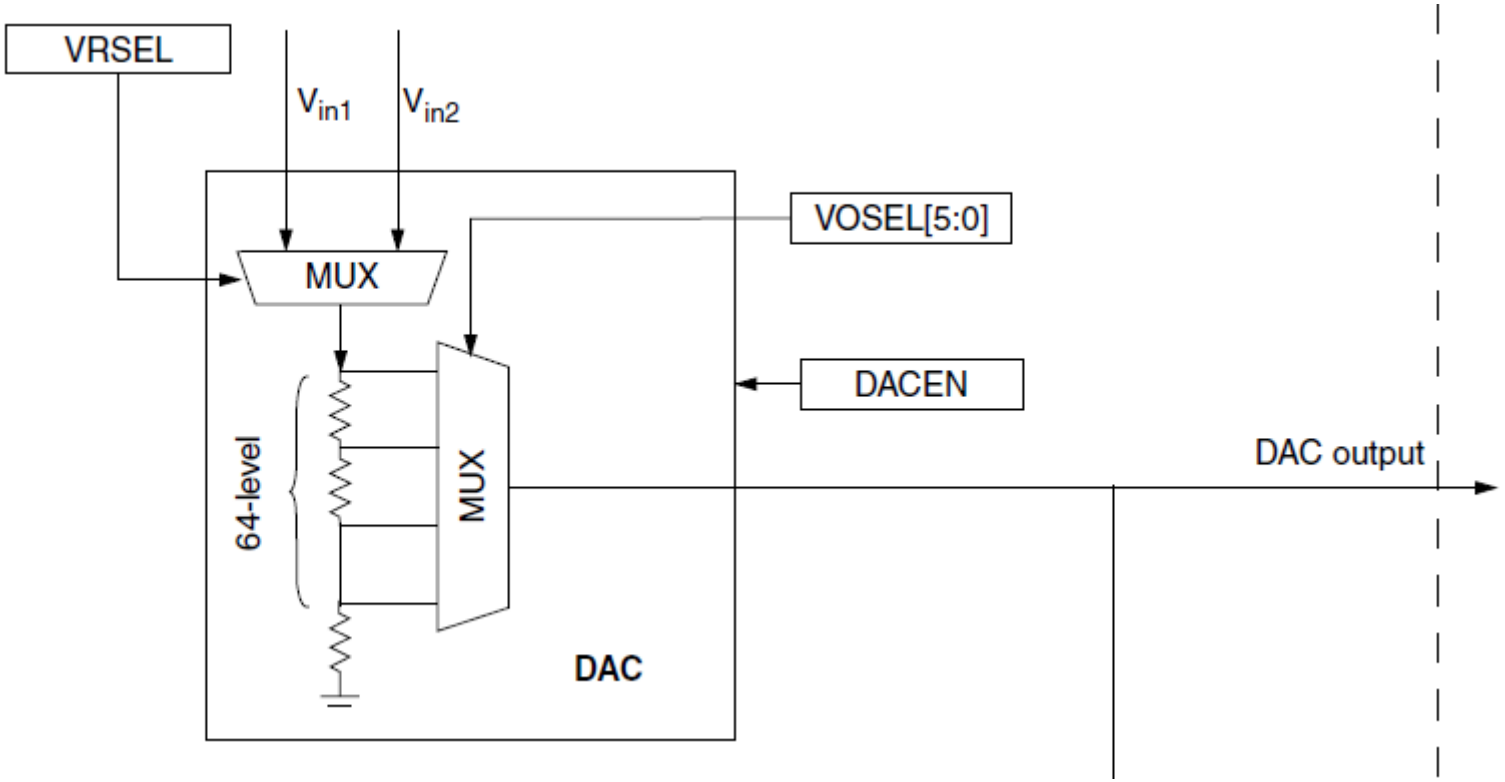
Sampled Mode. SE=1, WE=0, F_CNT>0



Sampled Mode. SE=0, WE=0, F_CNT>0

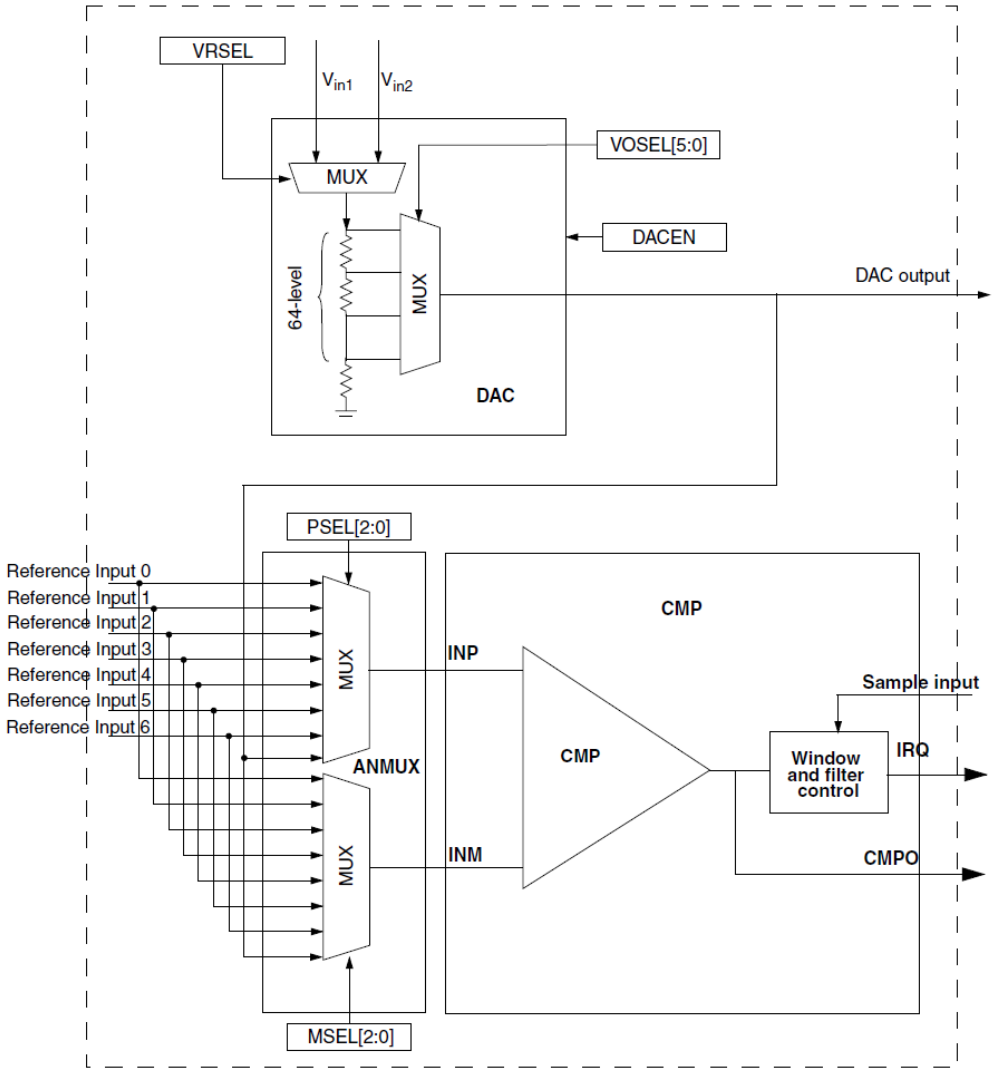


Comparator DAC



- Can set own reference voltage for comparator
- 6-bit DAC

Programmable Threshold for Comparator

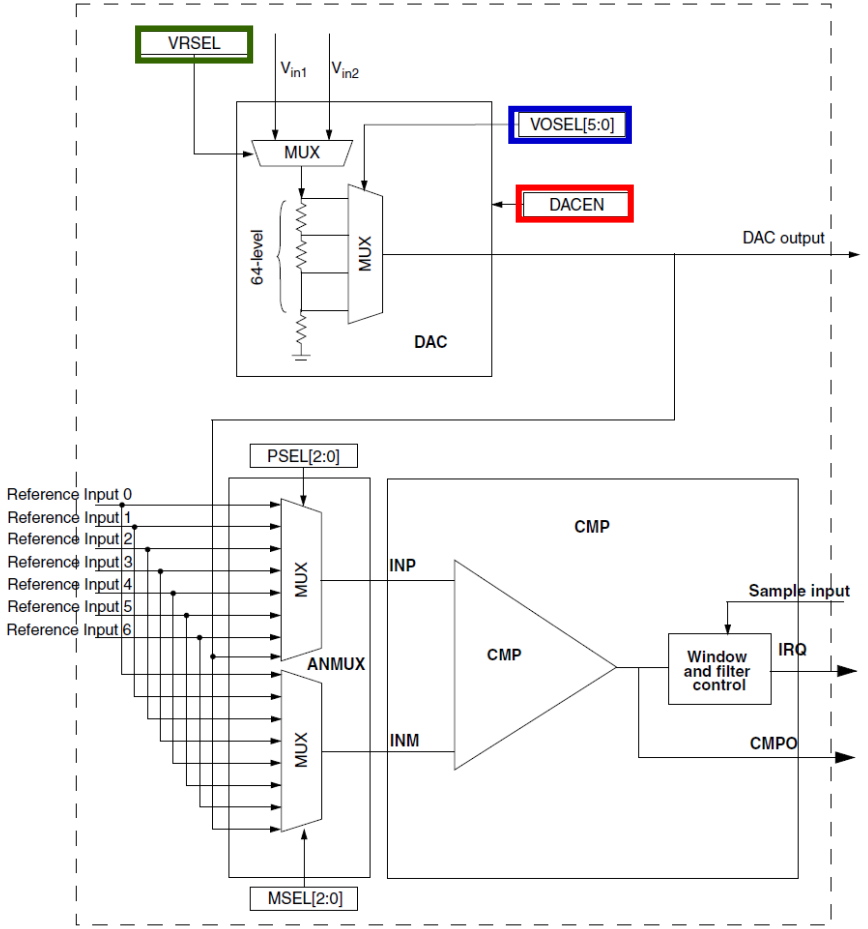


- Can use DAC to set threshold voltage for comparator

DAC Control Register CMPx_DACCR

Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	VRSEL	VOSEL					
Reset	0	0	0	0	0	0	0	0

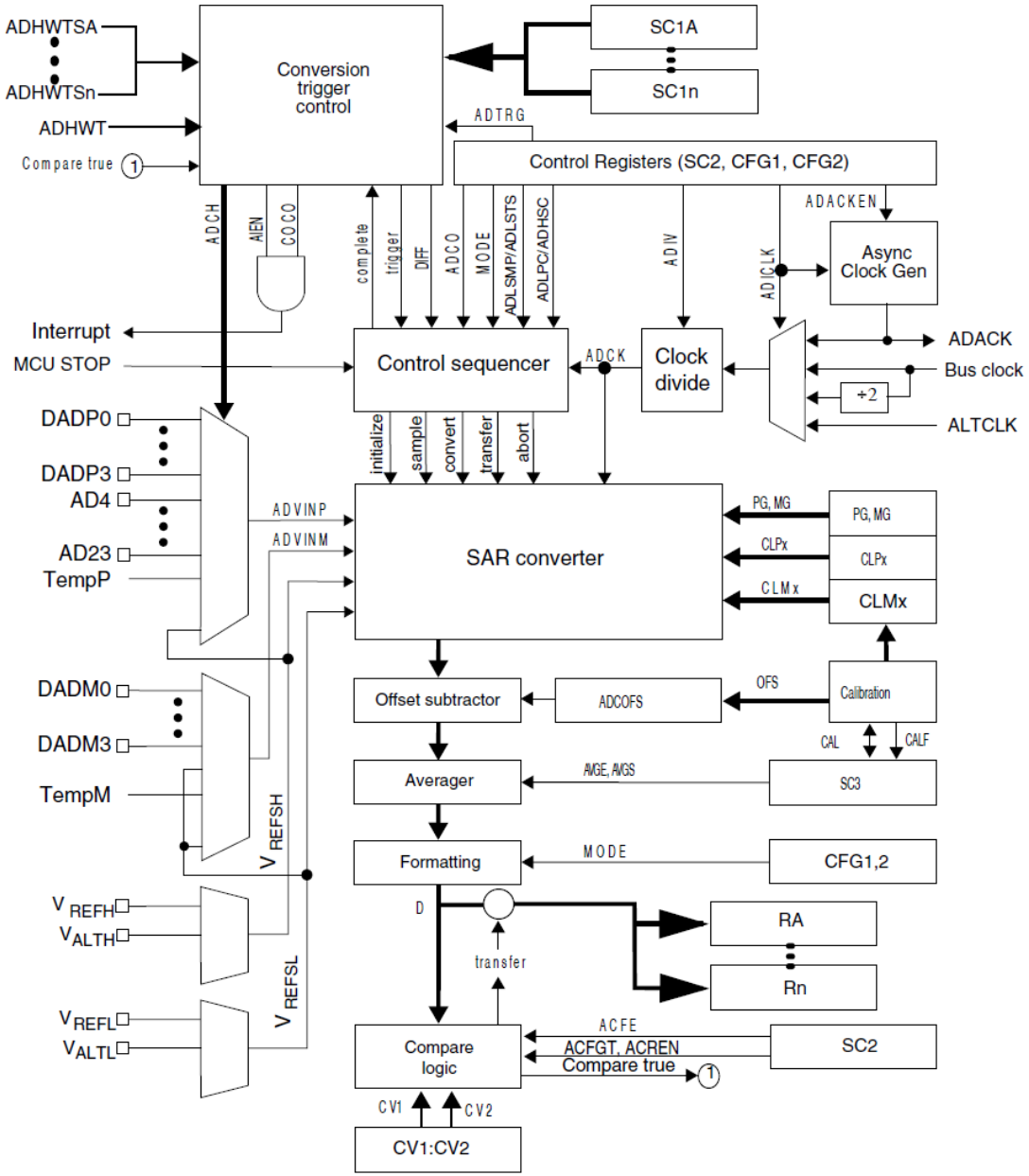
- **DACEN: Enable CMP DAC (1)**
- **VRSEL: DAC reference voltage select**
  - 0: Connected to VREFH
  - 1: Connected to VDD
- **VOSEL: Output voltage select**
  - $V_{DACO} = (VOSEL+1)*(V_{in}/64)$
  - $VOSEL = 64*(V_{DACO}/V_{in}) - 1$



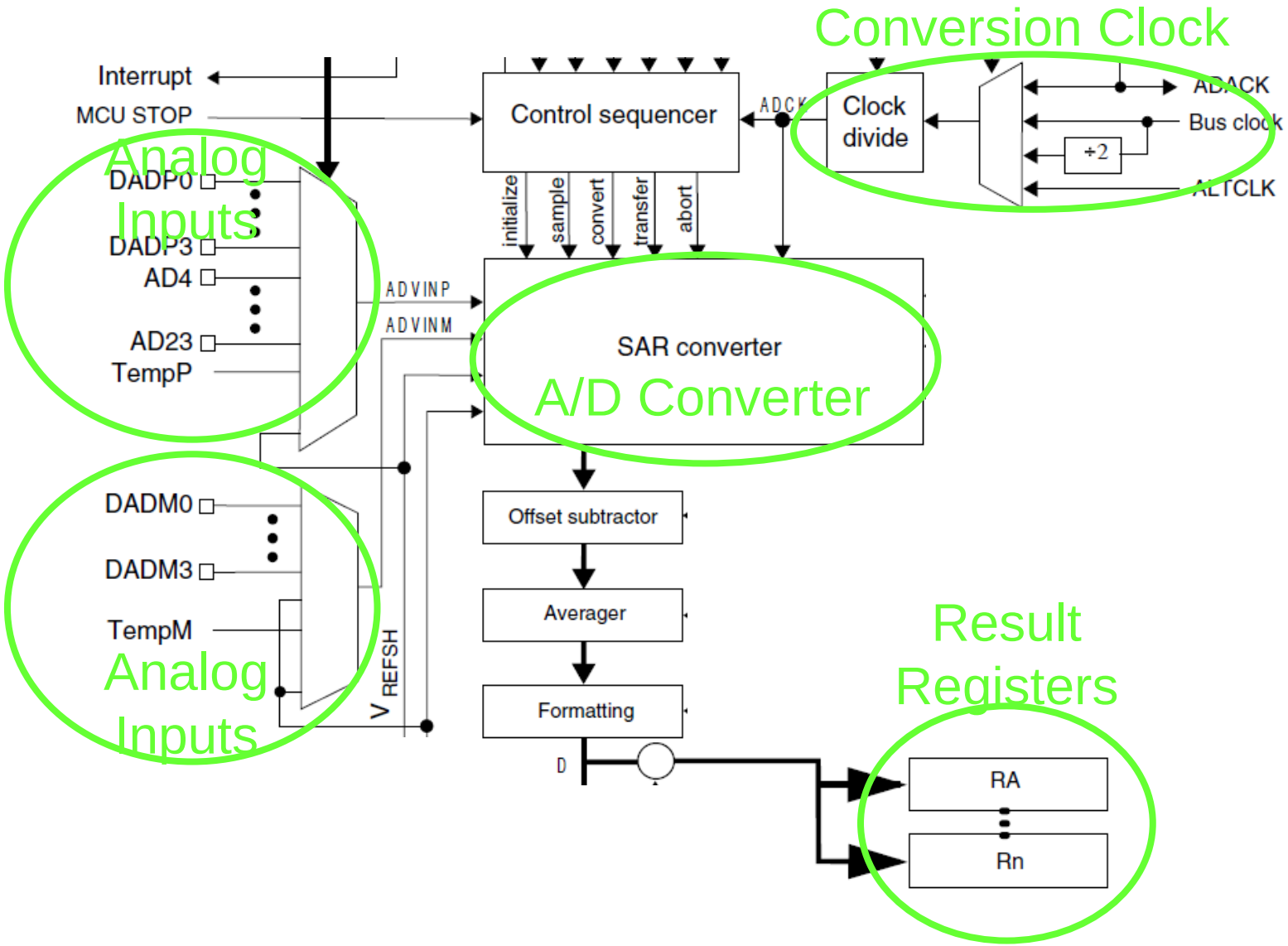
ANALOG TO DIGITAL CONVERTER

ADC Overview

- Uses **SAR** for conversion
- Supports multiple resolutions: 16, 13, 12, 11, 10, 9, and 8 bits
- Up to 24 analog inputs supported (single-ended), 4 pairs of differential inputs
- Signed or unsigned results available
- Automatic compare and interrupt for level and range comparisons
- Hardware data averaging
- Built-in Temperature sensor



ADC System Fundamentals

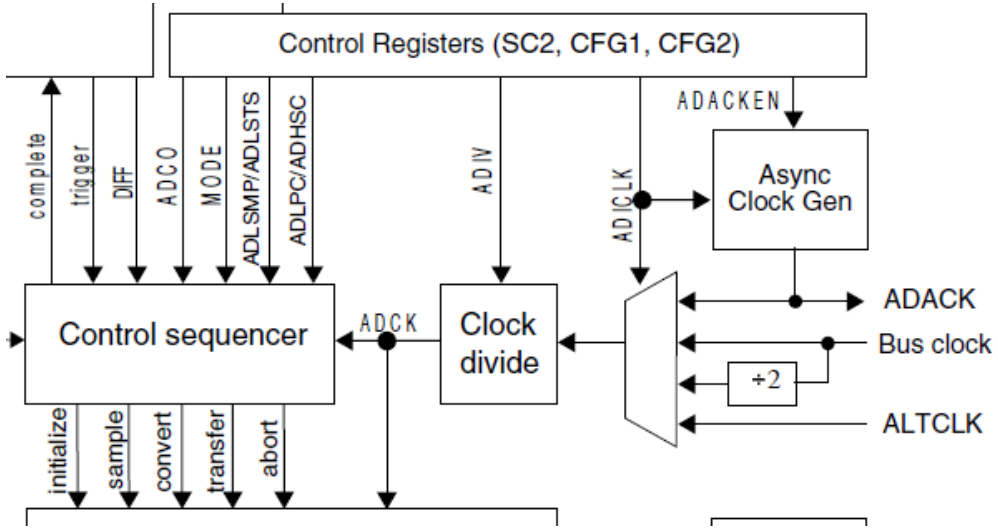


Using the ADC

- **ADC initialization**
  - Configure clock
  - Select voltage reference
  - Select trigger source (hardware or software)
  - Select input channel
- **Trigger conversion**
- **Read results**

Clock Configuration

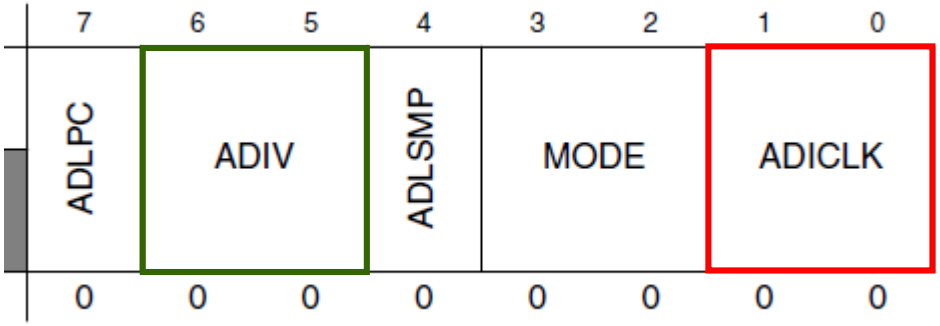
- Clock sources
  - Bus Clock (default)
  - ADACK: Local clock, allows ADC operation while rest of CPU is in stop mode
  - ALTCLK: alternate clock (MCU-specific)



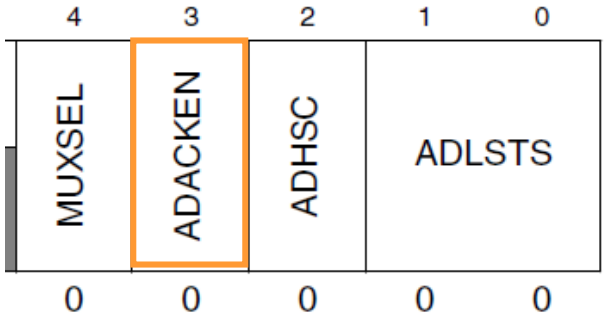
- Clock divider by a selected factor
- Resulting ADCK must be within valid range to ensure accuracy (See KL25 Subfamily datasheet)
  - 1 to 18 MHz (<= 13-bit mode)
  - 2 to 12 MHz (16-bit mode)

Clock Configuration Registers

- **ADCx_CFG1**
  - **ADICLK**: Input clock select
    - 00: Bus clock
    - 01: Bus clock/2
    - 10: ALTCLK
    - 11: ADACK
  - **ADIV**: divide clock by 2^{ADIV}
    - 00: 1
    - 01: 2
    - 10: 4
    - 11: 8

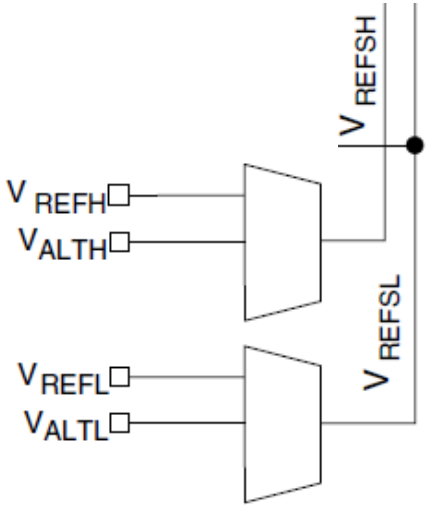


- **ADCx_CFG2**
  - **ADACKEN**: Enable asynchronous clock



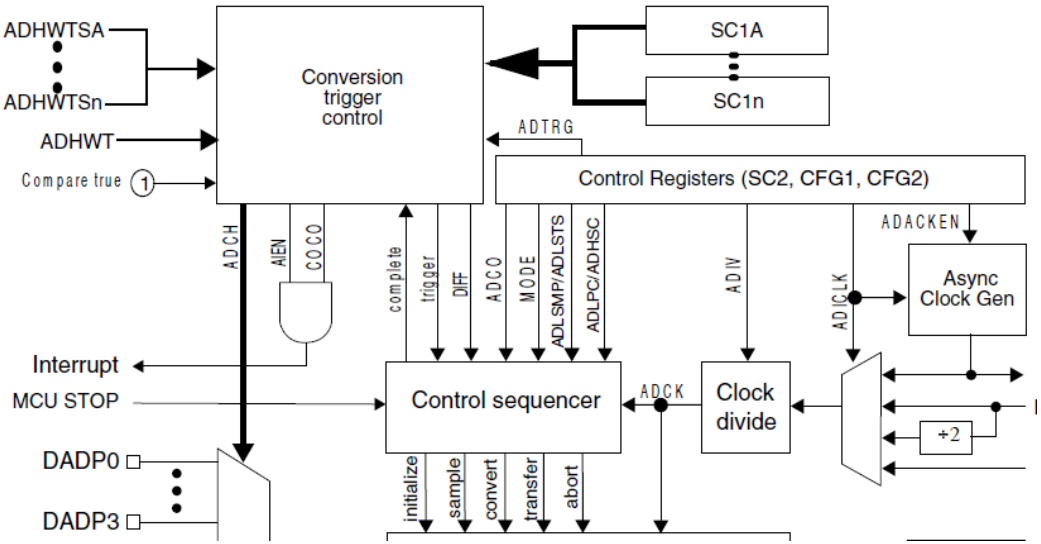
Voltage Reference Selection

- **Two voltage reference pairs available**
  - V_{REFH}, V_{REFL}
  - V_{ALTH}, V_{ALTL}
- **Select with SC2 register's REFSEL bits**
  - 00: V_{REFH}, V_{REFL}
  - 01: V_{ALTH}, V_{ALTL}
  - 10, 11: Reserved
- **KL25Z**
  - V_{ALTH} connected to V_{DDA}



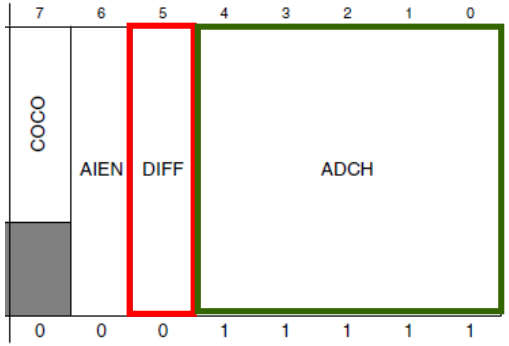
Conversion Trigger Selection

- **Two SC1 registers:**
  - SC1A and SC1B
- **ADTRG in SC2**
  - 0: software trigger
  - 1: hardware trigger
- **Software trigger:**
  - Programme writes to SC1A
- **Ping-pong buffering**
  - SC1A vs. SC1n
- **Hardware trigger:**
  - Rising edge of ADHWT signal
  - ADHWT sources: TPM, LPTMR, PIT, RTC, EXTRG_IN, HSCMP0
  - Selected in System Integration Module
    - SIM_SOPT7 register
    - See section 12.2.6 of Reference Manual

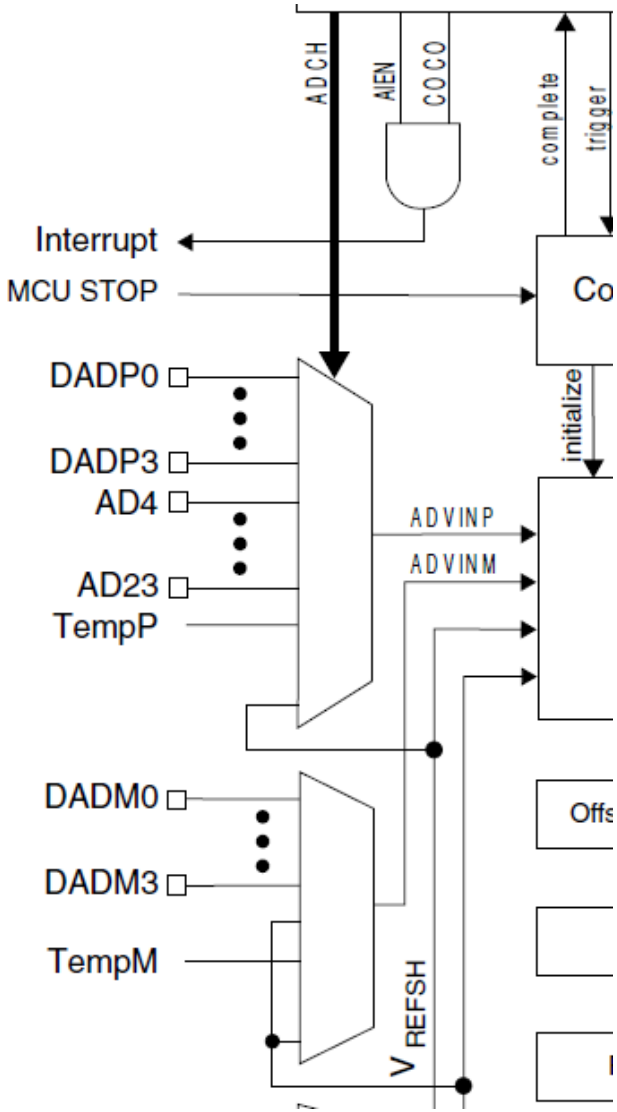


Input Channel Selection

- **Two modes, selected by SC1n[DIFF] bit**
  - 0: Single-ended
  - 1: Differential
- **Input channel selected by value of ADCH**
- **Extras**
  - Reference voltages
  - Temperature
  - Band Gap



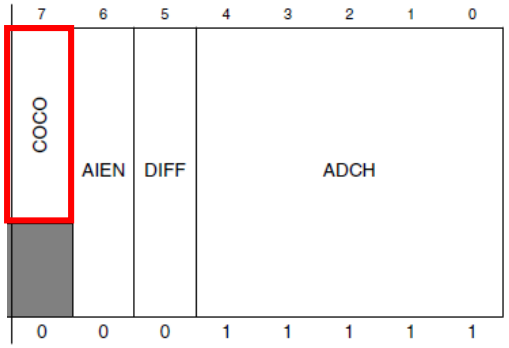
	DIFF	
ADCH	0	1
0	DADP0	DAD0
1	DADP1	DAD1
2	DADP2	DAD2
3	DADP3	DAD3
4	AD4	reserved
...	...	...
23	AD23	reserved
24	reserved	reserved
25	reserved	reserved
26	Temp Sensor	
27	Band Gap	
28	reserved	reserved
29	VREFSH	-VREFSH
30	VREFSL	reserved
31	module disabled	



Conversion Completion

- Signaled by **COCO** bit in SC1n

- Can generate conversion complete interrupt if AIEN in SC1 is set
- CMSIS-defined ISR name for ADC Interrupt is **ADC0_IRQHandler**

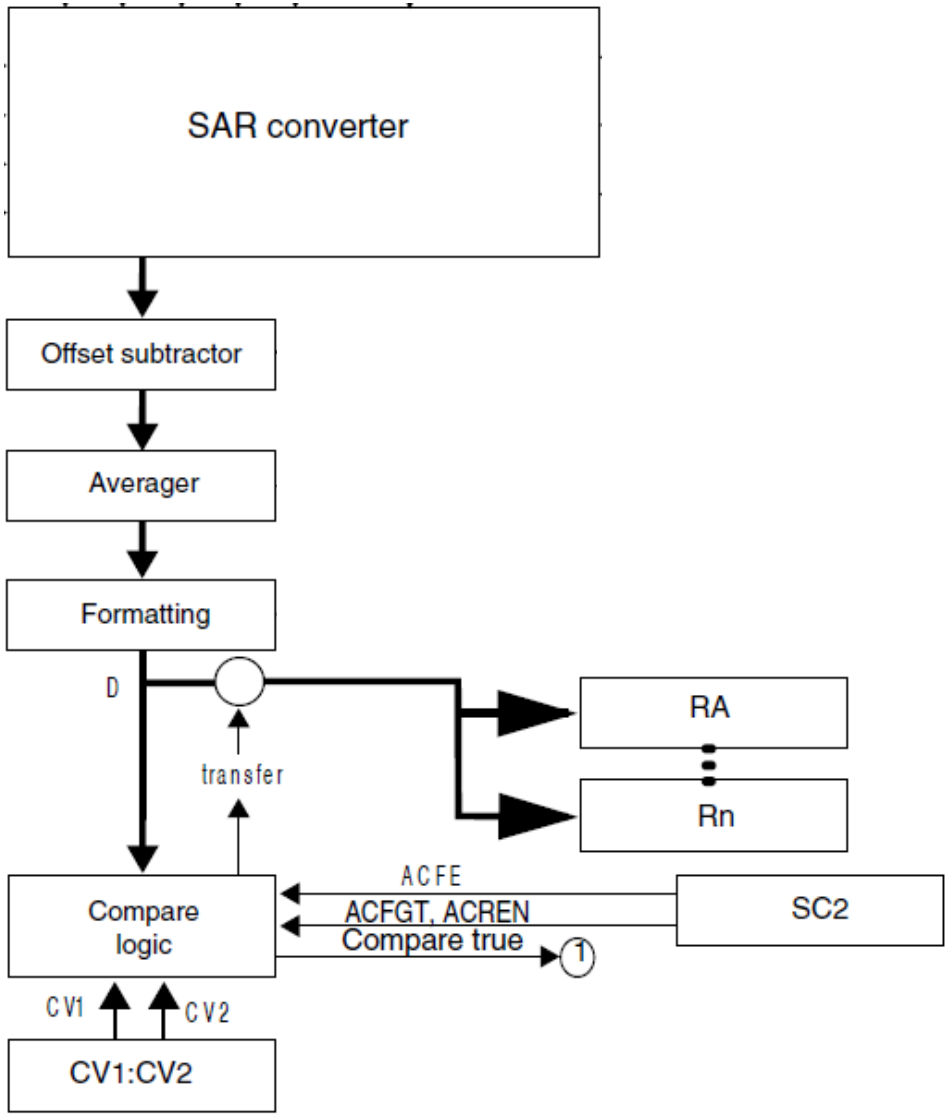


ADC simple code example

```
/* Initialize ADC */
void Init_ADC(void) {

   SIM->SCGC6 |= (1UL << SIM_SCGC6_ADC0_SHIFT); /* Enable ADC clock */
   ADC0->CFG1 = 0x9C; /* Select 16 bit resolution */
   ADC0->SC2 = 0;
}
/* Perform a voltage single measure */
unsigned Measure (void) {
   volatile unsigned res=0;

   ADC0->SC1[0] = 0x00;       /* start conversion on channel 0 */

   while (!(ADC0->SC1[0] & ADC_SC1_COCO_MASK)) ; /* wait conversion end */
   res = ADC0->R[0];          /* read data from result register */

   return res;
}
```

Result Registers

- **Optional output processing before storage in result registers**
  - Offset subtraction from calibration
  - Averaging: 1, 4, 8, 16 or 32 samples
  - Formatting: Right justification, sign- or zero-extension to 16 bits
  - Output comparison
- **Two result registers RA and RB**
  - Conversion result goes into register corresponding to SC1 register used to start conversion (SC1A, SC1n)



Output Averaging

- **Accumulate and average multiple samples before writing the averaged result to the result register**
- **Result rate = sample rate / averaging factor**
- **In SC3 register**
  - **AVGE**: average enable
  - **AVGS**: average sample count

AVGE	AVGS	Number of samples averaged
0	xx	1
1	00	4
1	01	8
1	10	16
1	11	32

Automatic Compare

- **Can ignore ADC result based on comparison with CV1 and CV2**
  - Above or equal to threshold
  - Below threshold
  - Outside range
  - Inside range
- **Ignored result…**
  - **Not saved to RA/Rn**
  - **CoCo not set**