Analog Interfacing

Why It's Needed – *The World is Analog!*

- Embedded systems often need to measure values of physical parameters
- These parameters are usually continuous (*analog*) and not in a digital form
  - Digital computers operate on digital discrete data values

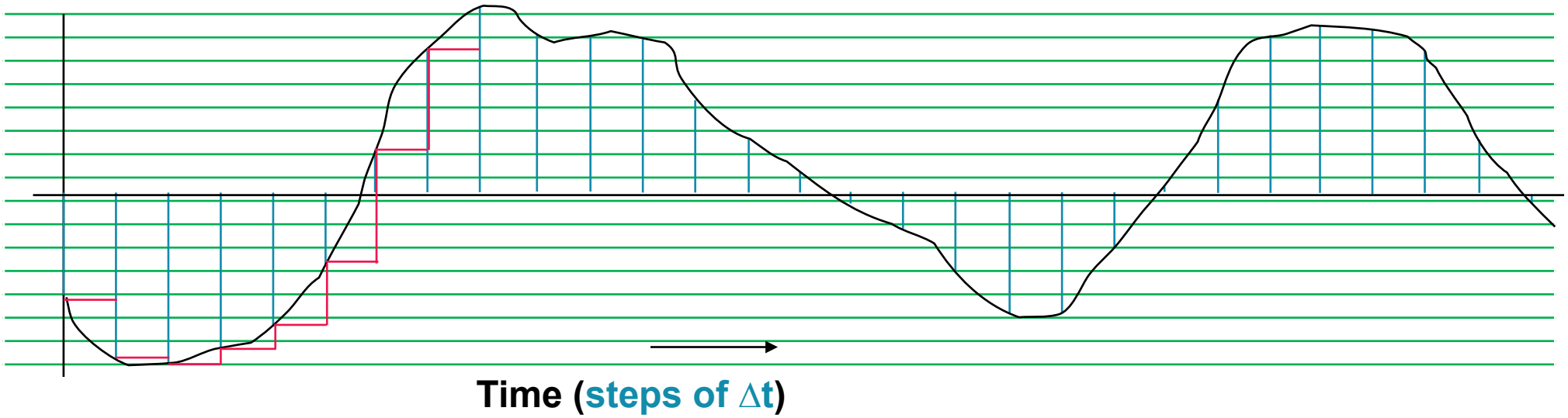
- Temperature
  - Thermometer (do you have a fever?)
  - Thermostat for building, fridge, freezer
  - Car engine controller
  - Chemical reaction monitor
  - Safety (e.g. microprocessor processor thermal management)
- Light (or infrared or ultraviolet) intensity
  - Digital camera
  - IR remote control receiver
  - Tanning bed
  - UV monitor
- Rotary position
  - Wind gauge
  - Knobs

- Pressure
  - Blood pressure monitor
  - Altimeter
  - Car engine controller
  - Scuba dive computer
  - Tsunami detector
- Acceleration
  - Air bag controller
  - Vehicle stability
  - Video game remote
- Mechanical strain
- More…
  - Touch screen controller
  - EKG, EEG

- Some output signals need to be analog (e.g. audio signals)

SAMPLING AND QUANTIZATION

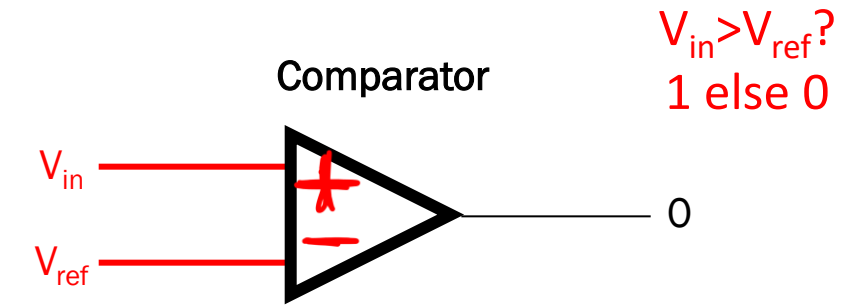
Waveform Sampling and Quantization



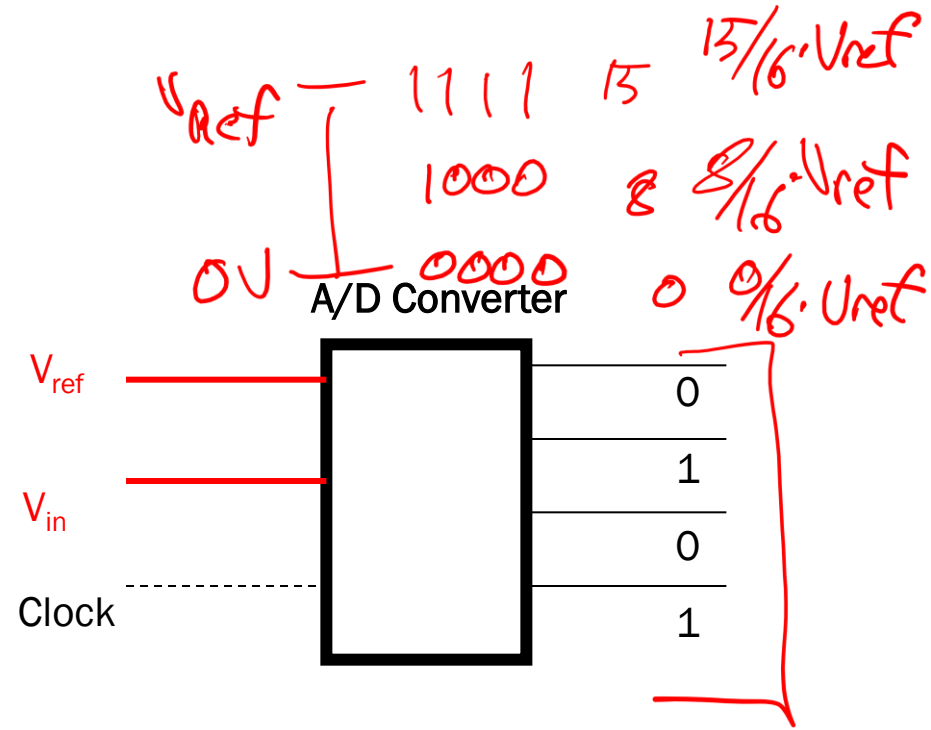
- A waveform (continuous time and value) is sampled every Δt
  - Each such sample represents the instantaneous amplitude at the instant of sampling
  - *"At 37 ms, the input is 1.9134191451345145123431 1… V"*
  - Sampling converts a **continuous time** signal to a **discrete time** signal
- The sample can now be quantized (converted) into a digital value
  - Quantization represents a **continuous** (analog) value with the closest **discrete** (digital) value
  - *"The sampled input voltage of 1.9134191451345145123431 1… V is best represented by the code 0x018, since it is in the range of 1.901 to 1.9980V which corresponds to code 0x018."*

Quantization: Going From Analog to Digital

- A Comparator tells us "Is $V_{in} > V_{ref}$?"
  - Compares an **analog input voltage** with an **analog threshold 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.

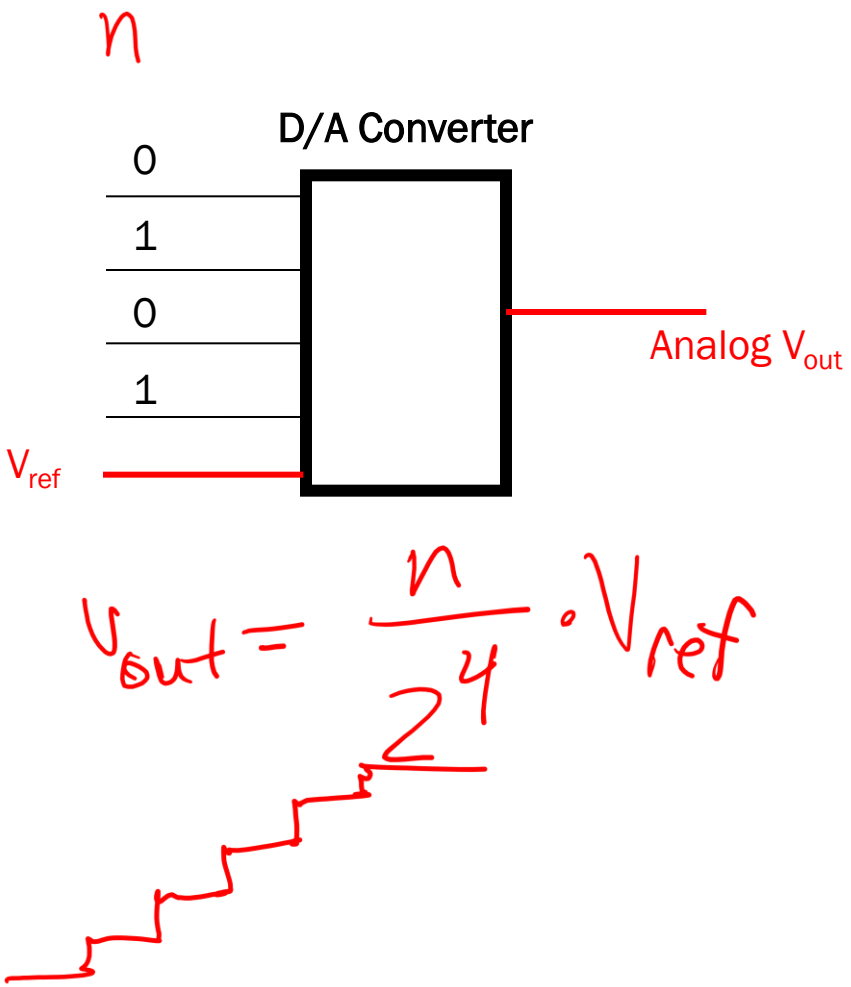


- An Analog to Digital converter [AD or ADC] tells us how large V_{in} is as a fraction of **full-scale reference voltage** V_{ref}.
  - Reads an analog input signal (usually a voltage) and produces a corresponding multi-bit number at the output.
  - E.g. calculate the depth



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. Usually either
    - $V_{out} = V_{ref} * n/(2^N)$ or
    - $V_{out} = V_{ref} * (n+1)/(2^N)$
    - *The offset term (+1) depends on the configuration of the DAC – check the datasheet to be sure*



Forward Transfer Function Equation Example

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

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

General Equation

n = converted code

V_{in} = sampled input voltage

V_{+ref} = upper voltage reference

V_{-ref} = lower voltage reference

N = number of bits of resolution in ADC

Simplification with V_{-ref} = 0 V

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

$$n = \left\lfloor \frac{3.30v\ 2^{10}}{5v} + 1/2 \right\rfloor = 676$$

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

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

Inverse Transfer Function Example

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

General Equation

n = converted code

V_{in_min} = minimum input voltage for code n

V_{in_max} = maximum input voltage for code n

V_{+ref} = upper voltage reference

V_{-ref} = lower voltage reference

N = number of bits of resolution in ADC

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

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

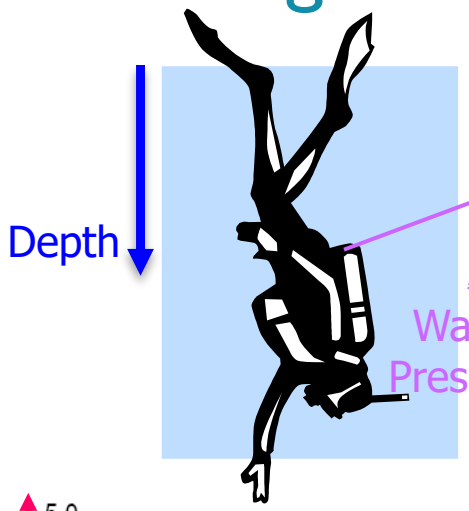
Simplification with V_{-ref} = 0 V

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

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

ANALOG TO DIGITAL CONVERSION CONCEPTS

The Big Picture – A Depth Gauge



Pressure Sensor

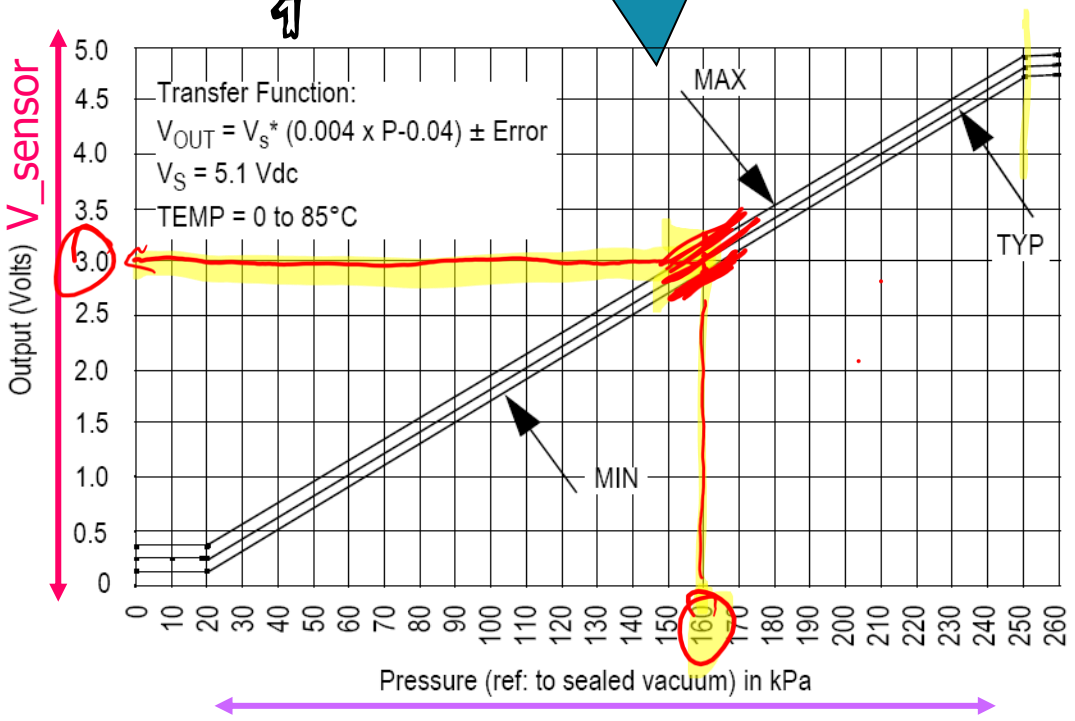
V_ref

Analog to Digital Converter

V_sensor

ADC_Code

```
// Your software
ADC_Code = ADC0->R[0];
V_sensor = (ADC_code/1024)*V_ref;
Pressure_kPa = 250 * (V_sensor/V_supply+0.04);
Depth_ft = 33 * (Pressure_kPa –
                 Atmos_Press_kPa)/101.3;
```



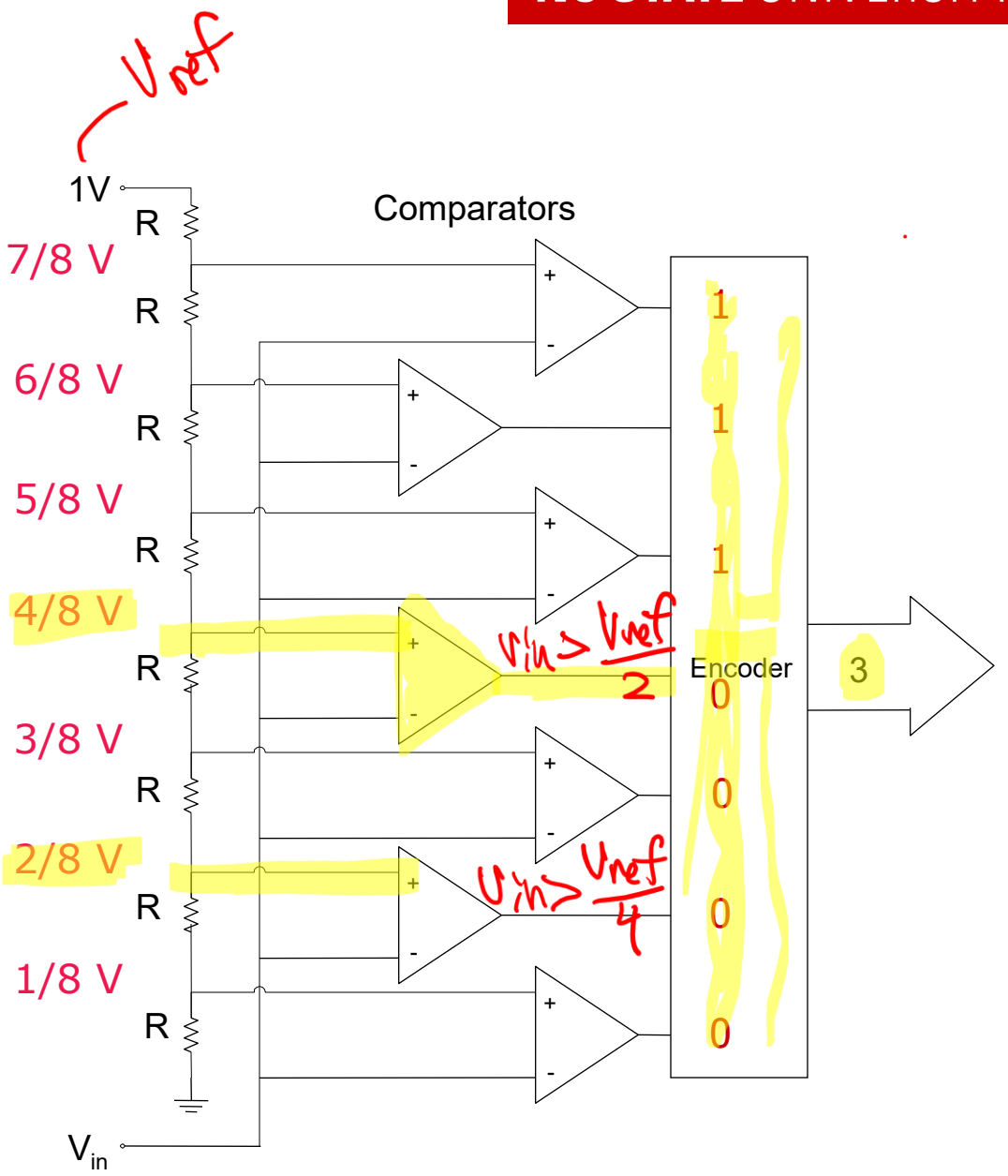
Voltages	ADC Output Codes
V_ref →	111..111 111..110 111..101 111..100
V_sensor →	ADC_Code
Ground (0 V) →	000..001 000..000

1. Sensor detects *water pressure* and generates a proportional *output voltage* V_sensor
2. ADC generates a proportional digital *integer* (ADC_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*

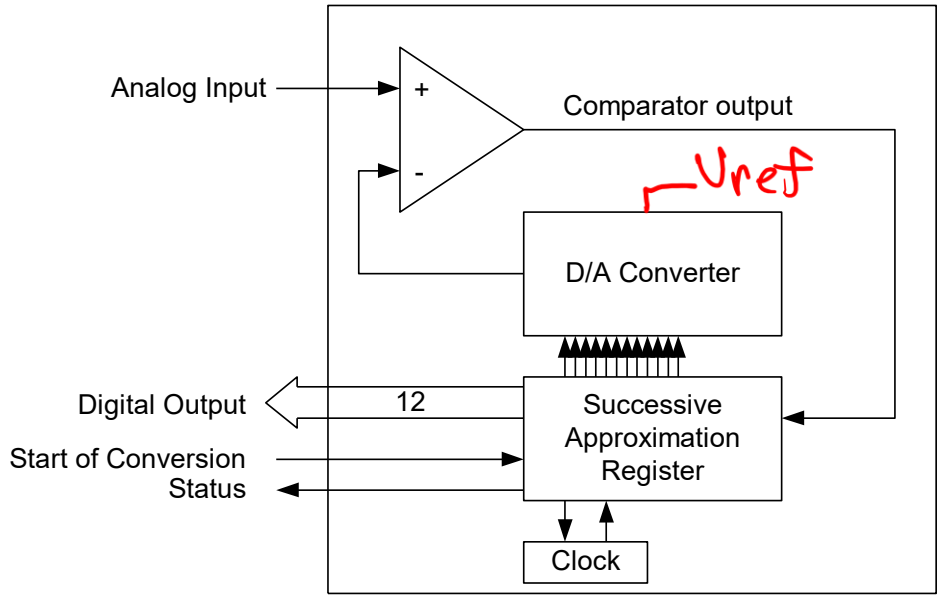
Figure 4. Output vs. Absolute Pressure

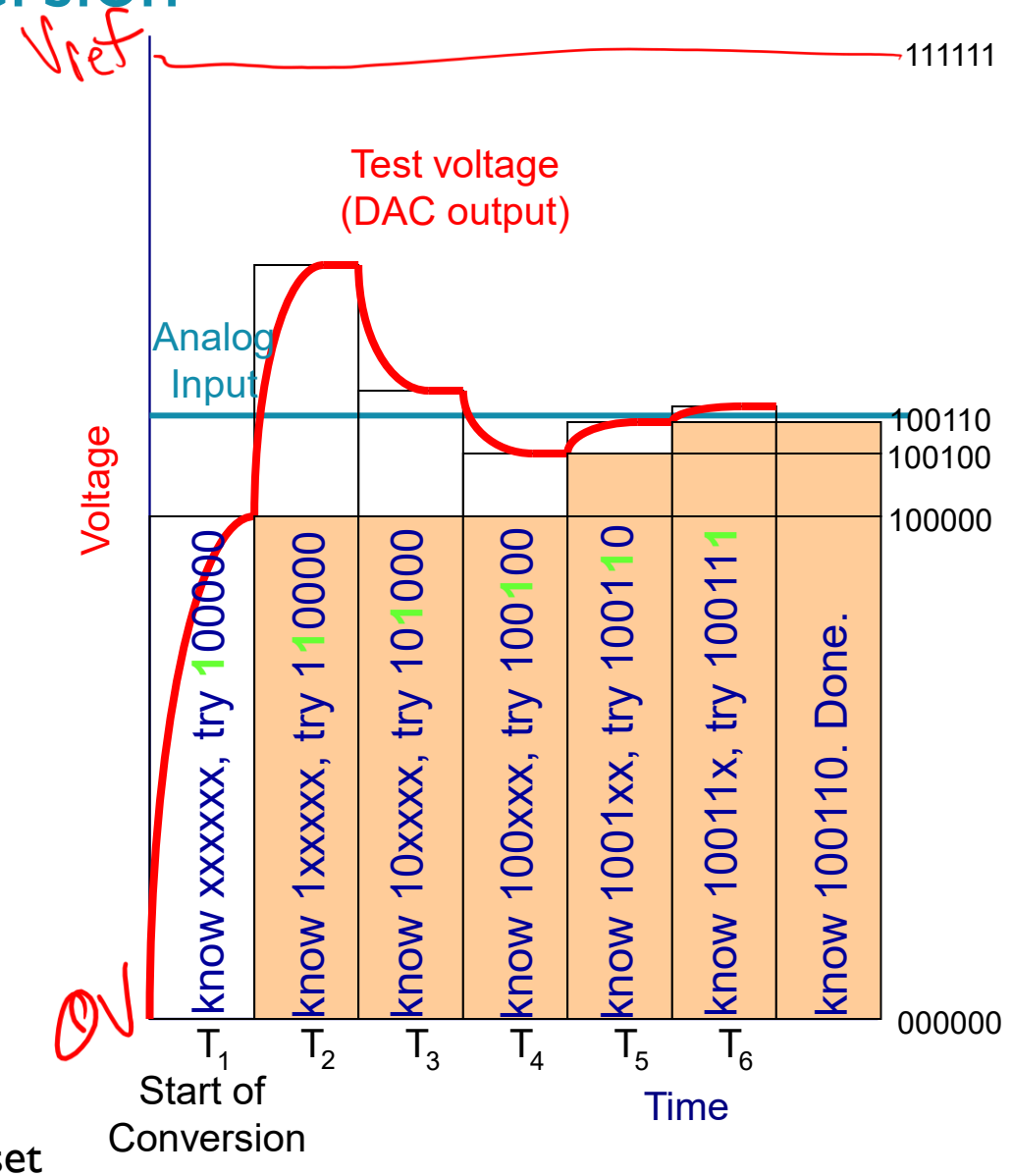
A/D – Flash Conversion

- A multi-level voltage divider is used to set voltage levels over the complete range of conversion.
- A comparator is used at each level to determine whether the voltage is lower or higher than the level.
- The series of comparator outputs are encoded to a binary number in digital logic (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)



ADC - Successive Approximation Conversion

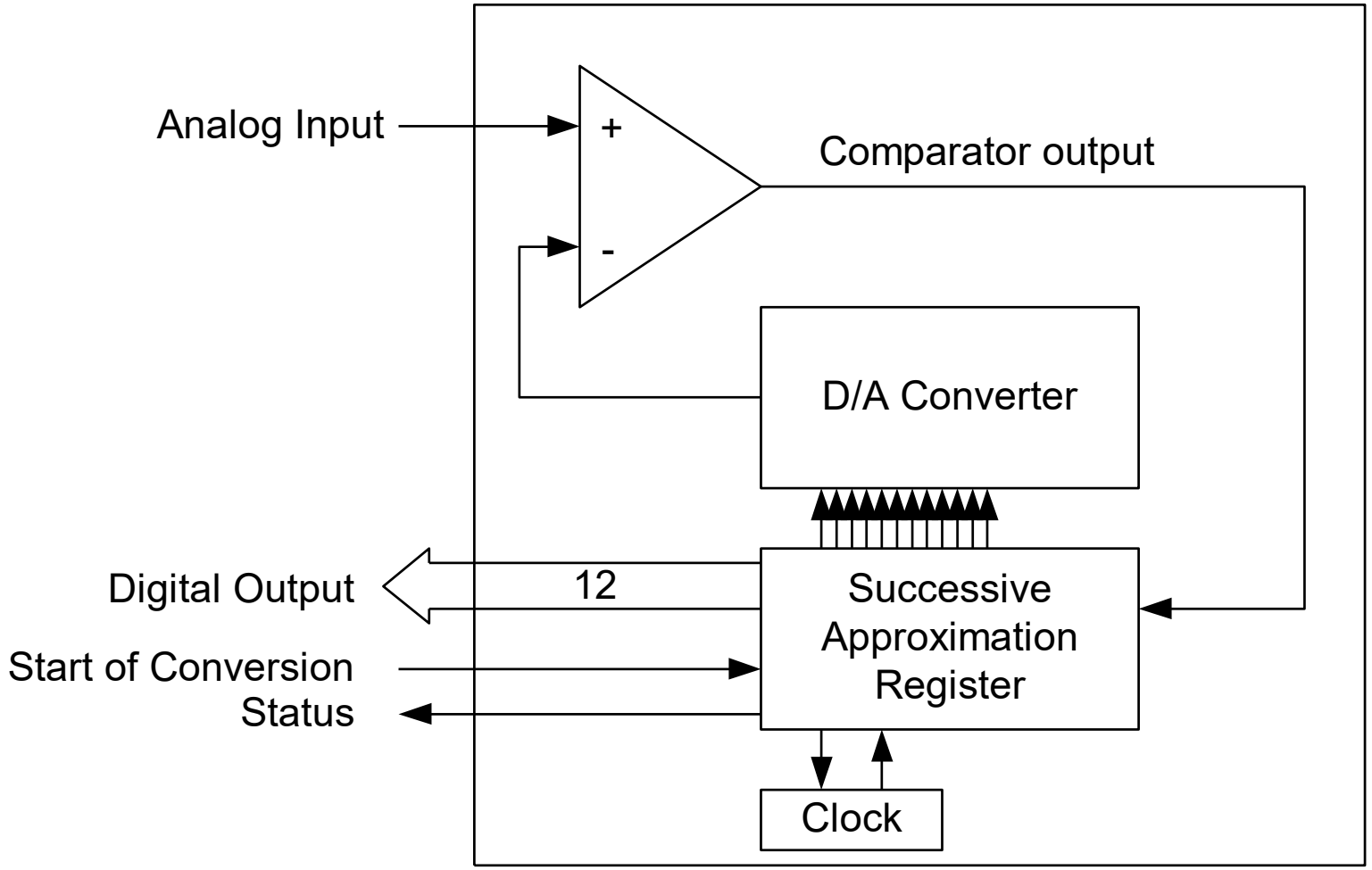




- Approximate input voltage by using a DAC and binary search
- Register feeds DAC, holds current approximation of result
- Set all DAC input bits to 0
- Start with DAC's most significant bit
- Repeat
  - Set next input bit for DAC to 1
  - Wait for DAC and comparator to stabilize
  - If the DAC output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

A/D - Successive Approximation

Converter Schematic



ADC Performance Metrics

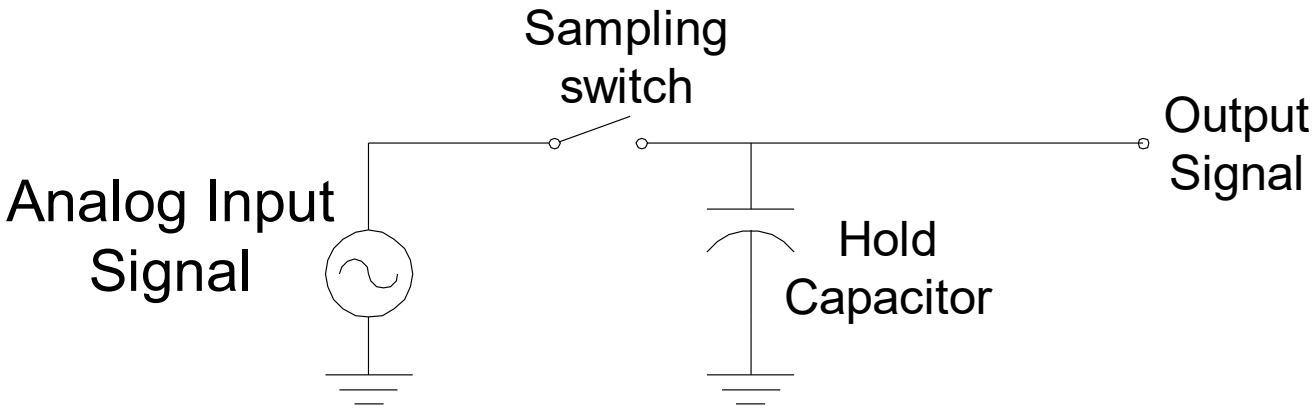
- Linearity measures how well the transition voltages lie on a straight line.
- Differential linearity measure the equality of the step size.
- Conversion time: between start of conversion and generation of result
- Conversion *rate* = inverse of conversion *time*

Sampling Problems

- 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 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

Sample and Hold Devices



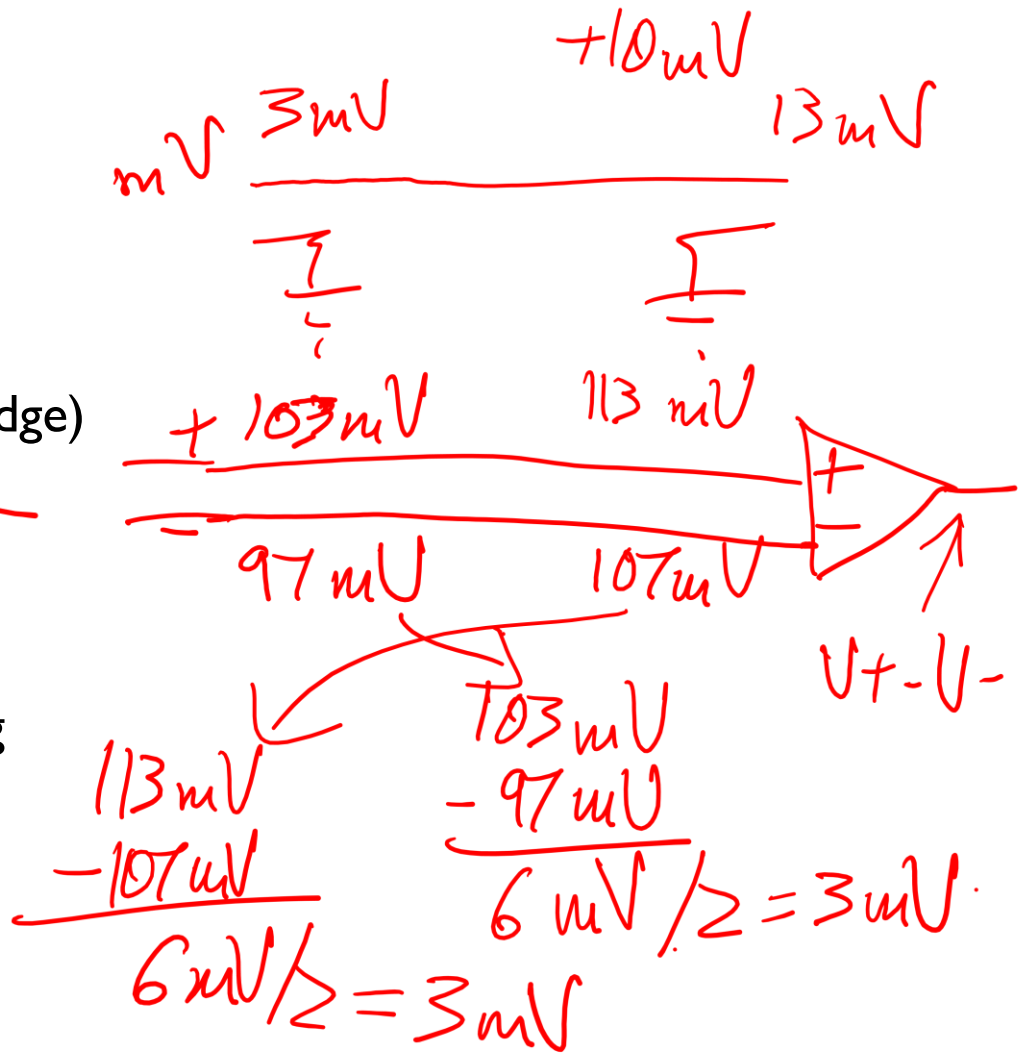
- Some A/D converters require the input analog signal to be held constant during conversion (e.g. successive approximation devices)
- In other cases, peak capture or sampling at a specific point in time requires a sampling device.
- A "sample and hold" circuit performs this operation
- Many A/D converters include a sample and hold circuit

Inputs

- Differential
  - Use two channels, and compute difference between them
  - Very good noise immunity
  - Some sensors offer differential outputs (e.g. Wheatstone Bridge)

- Multiplexing
  - Typically share a single ADC among multiple inputs
  - Need to select an input, allow time to settle before sampling

- Signal Conditioning
  - Amplify and filter input signal
  - Protect against out-of-range inputs with clamping diodes



What if the Reference Voltage is not known?

- Example - running off an unregulated battery (to save power)
- Measure a known voltage and an unknown voltage

$$V_{unknown} = V_{known} \frac{n_{unknown}}{n_{known}}$$

- Many MCUs include an internal fixed voltage source which ADC can measure for this purpose
- Can also solve for Vref

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

"My ADC tells me that channel 27 returns a code of 0x6543, so I can calculate that V_{REFSH} = 1.0V * 2^{16}/0x6543 = …

KL25 ANALOG INTERFACING PERIPHERALS

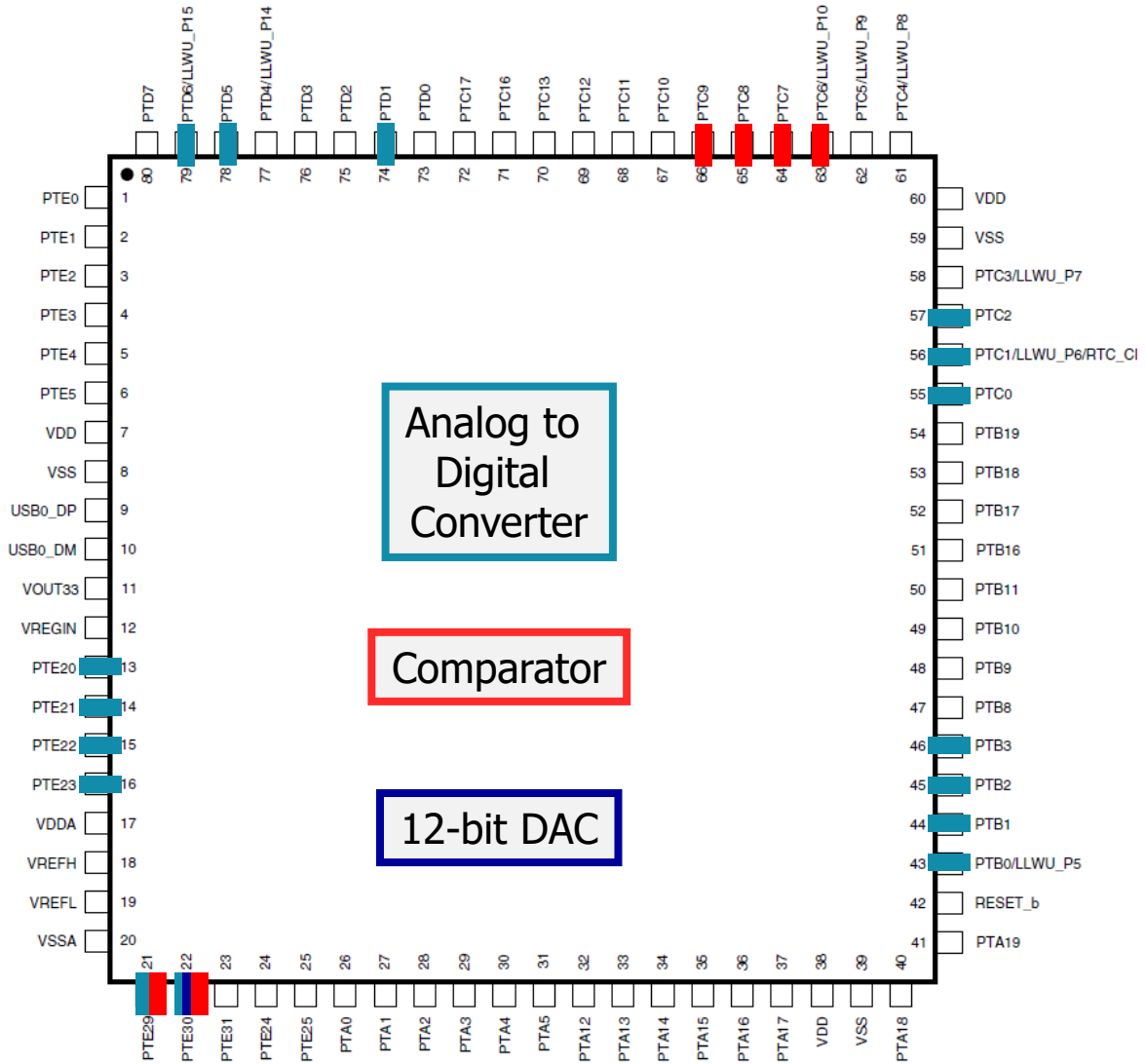
Sources of Information

- 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

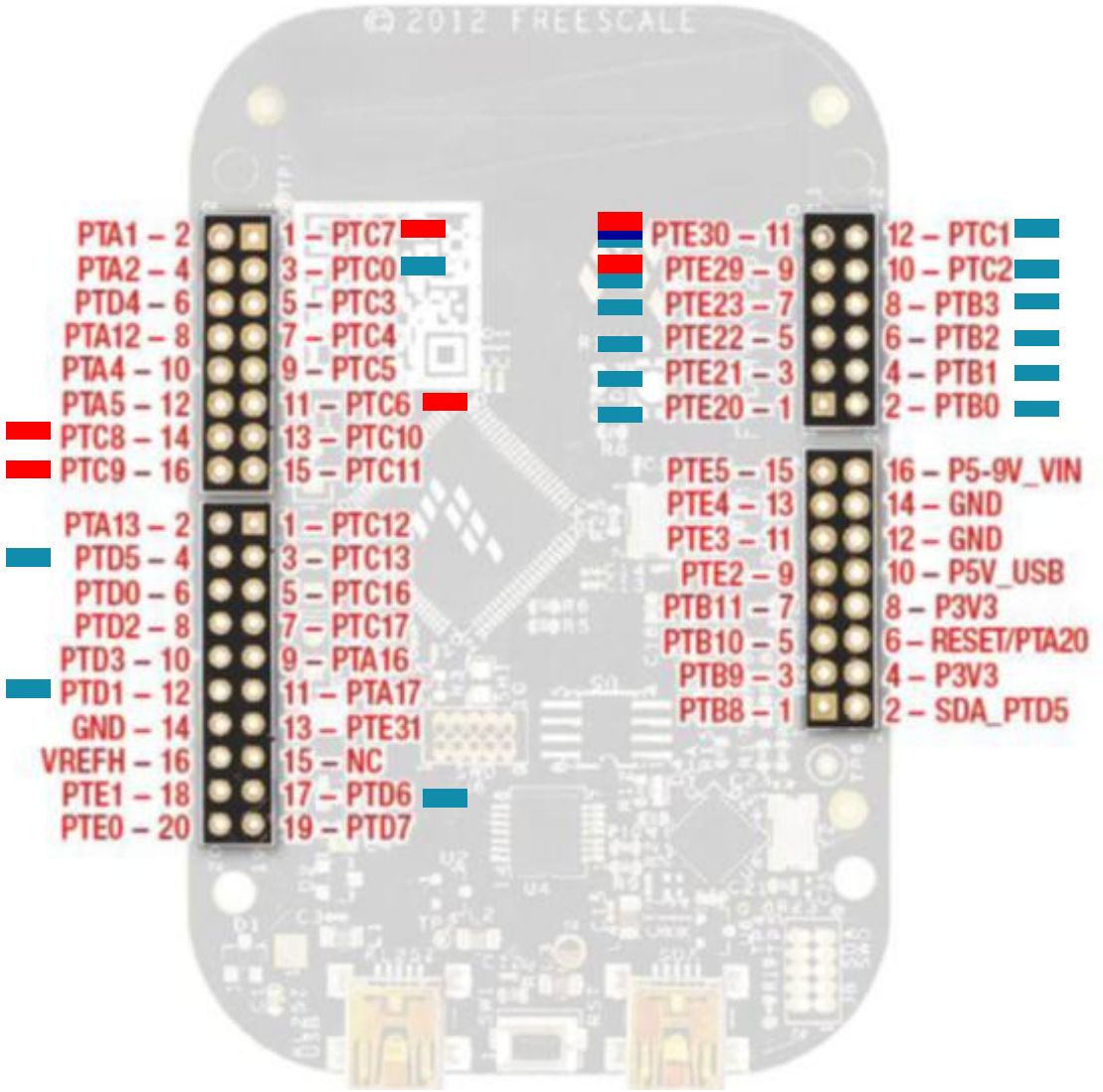
- KL25 Sub-family Data Sheet (Rev. 3, 9/19/2012)
  - Describes circuit-specific performance parameters: operating voltages, min/max speeds, cycle times, delays, power and energy use

KL25Z Analog Interface Pins

- 80-pin QFP
- Inputs
  - 1 16-bit ADC with 14 input channels
  - 1 comparator with 6 external inputs (and one 6-bit DAC)
- Output
  - 1 12-bit DAC



Freedom KL25Z Analog I/O



Inputs

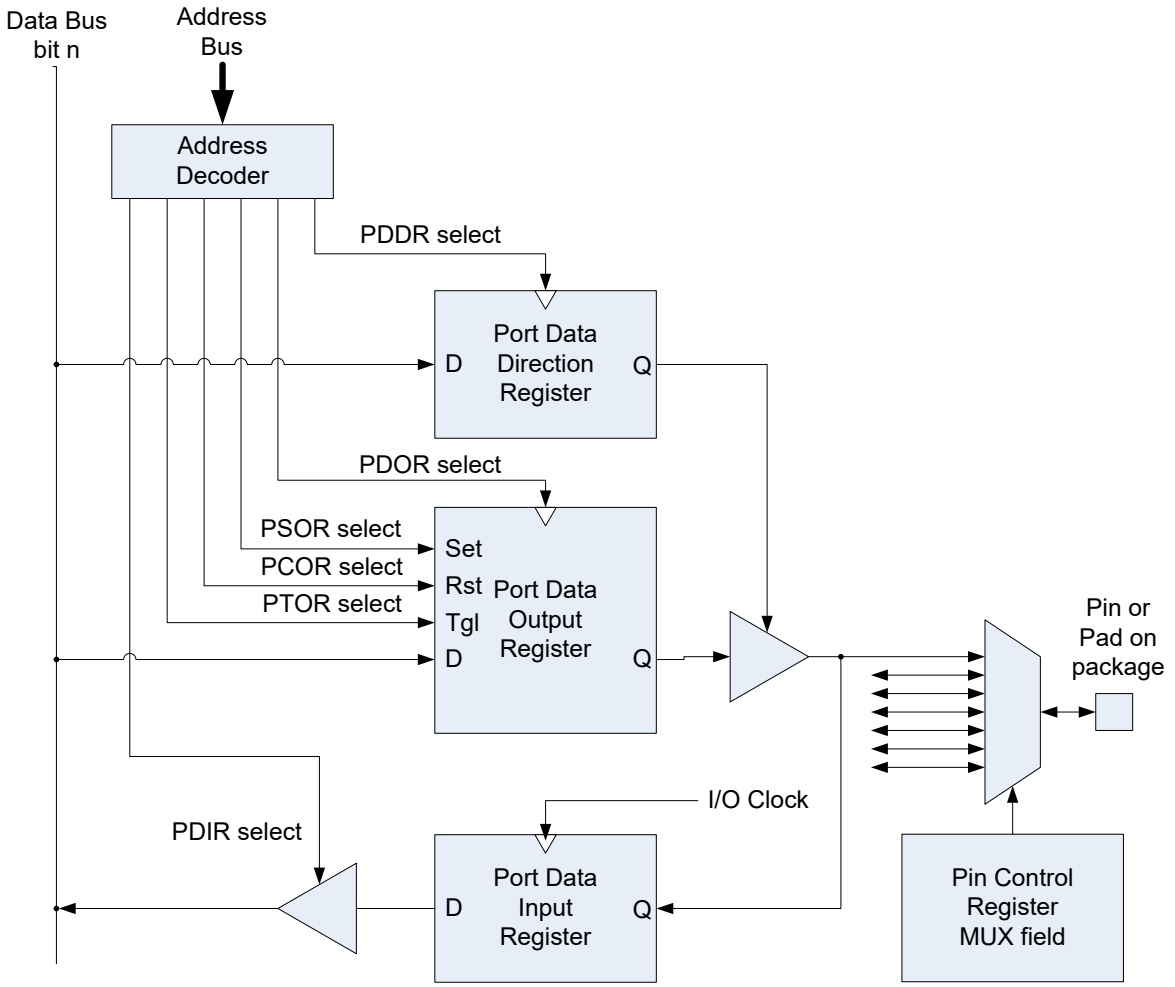
- 14 external ADC channels
- 6 external comparator channels

Output

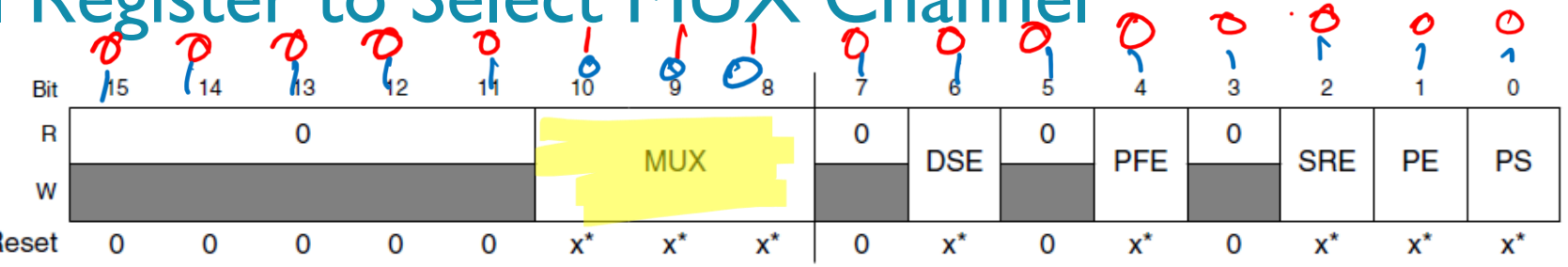
- 1 12-bit DAC

Using a Pin for Analog Input or Output

- 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

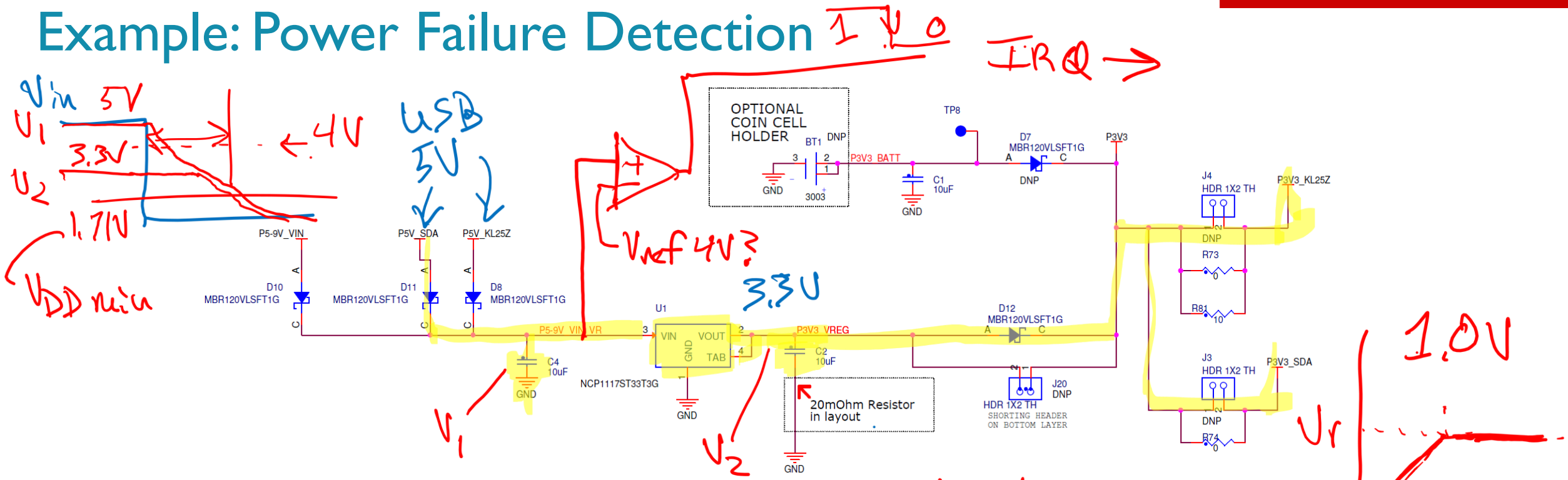
MUX (bits 10-8)	Configuration
000	Digital circuits disabled, analog enabled
001	Alternative 1 – GPIO
010	Alternative 2
011	Alternative 3
100	Alternative 4
101	Alternative 5
110	Alternative 6
111	Alternative 7

RMW: Read And with 1111-1000-1=1 Write to PCR[7]

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

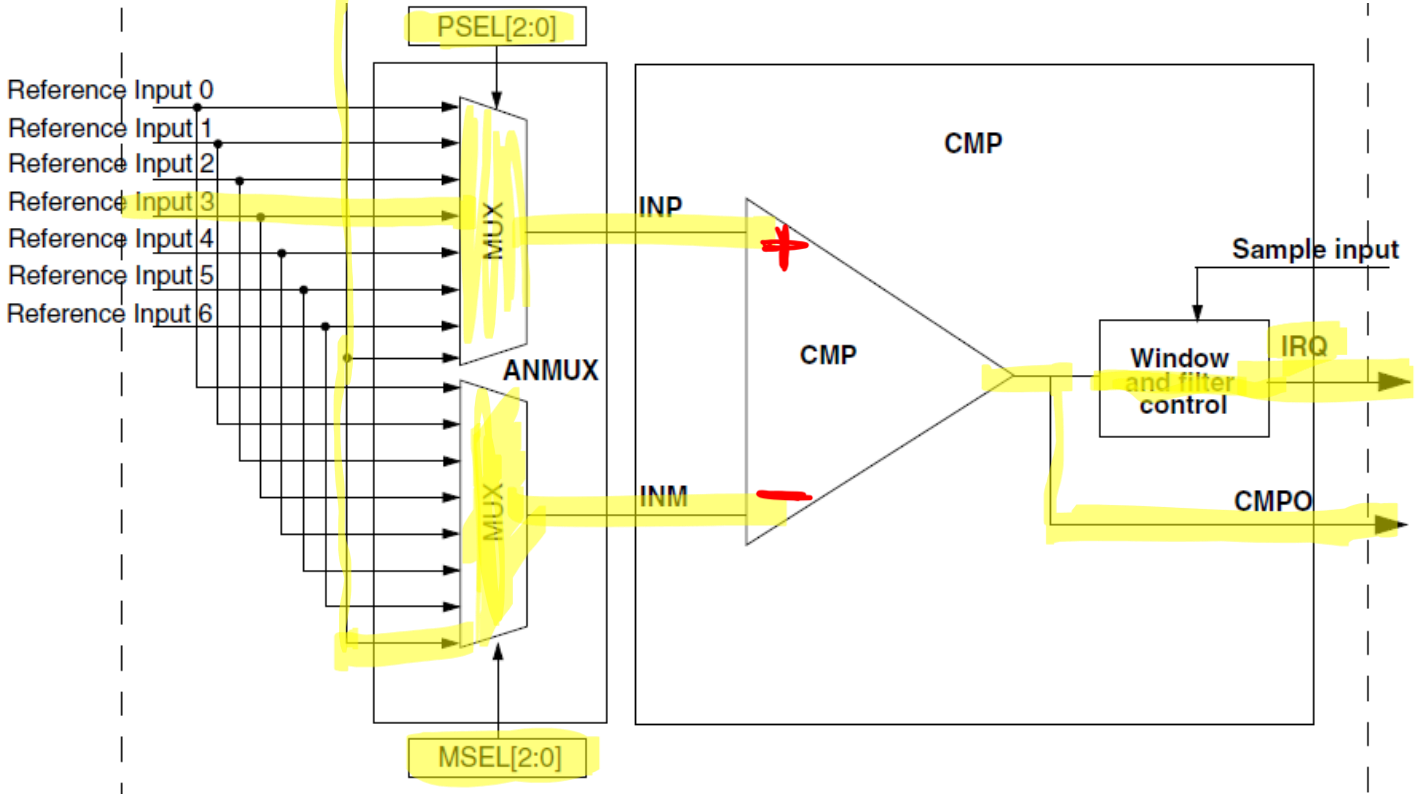
ANALOG COMPARATOR

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

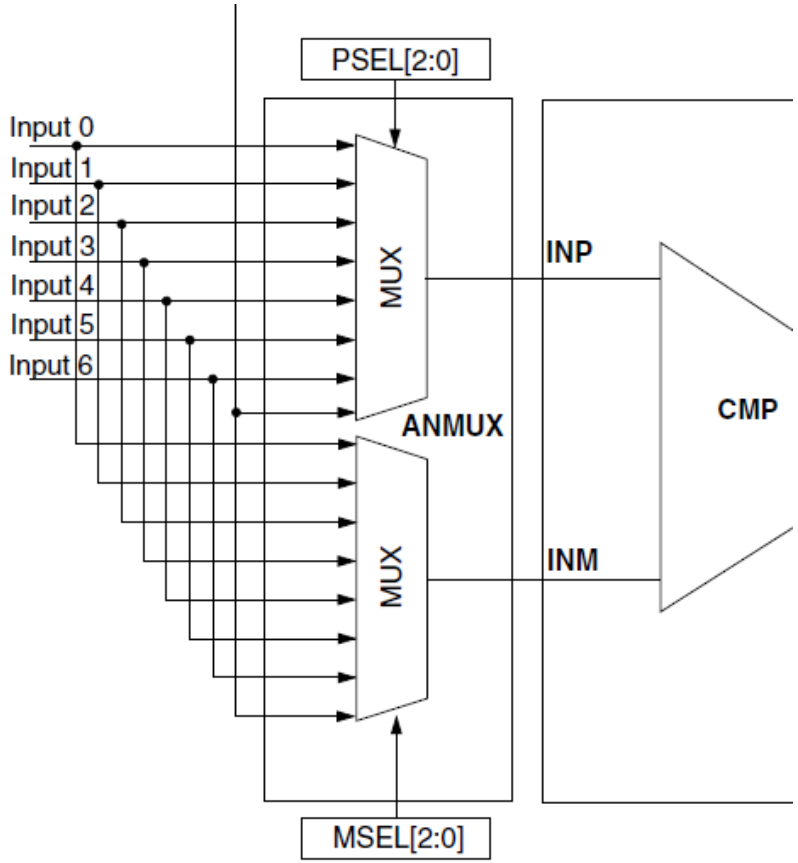
Bit	7	6	5	4	3	2	1	0
Read Write	SE	WE	TRIGM	PMODE	INV	COS	OPE	EN
Reset	0	0	0	0	0	0	0	0

- 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

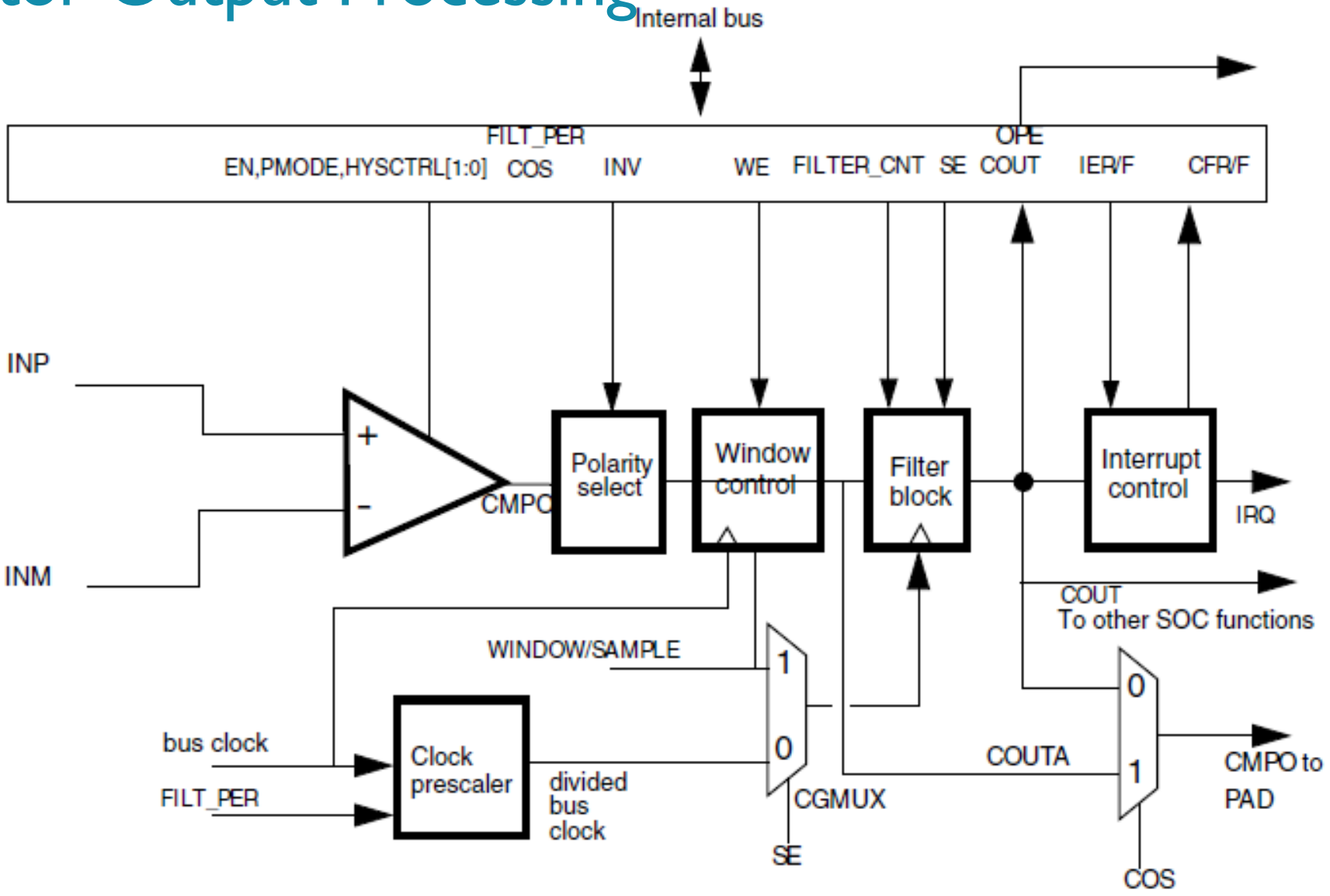
MUX Control Register CMPx_MUXCR

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

- PSTM: Enable Pass Through Mode
- 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

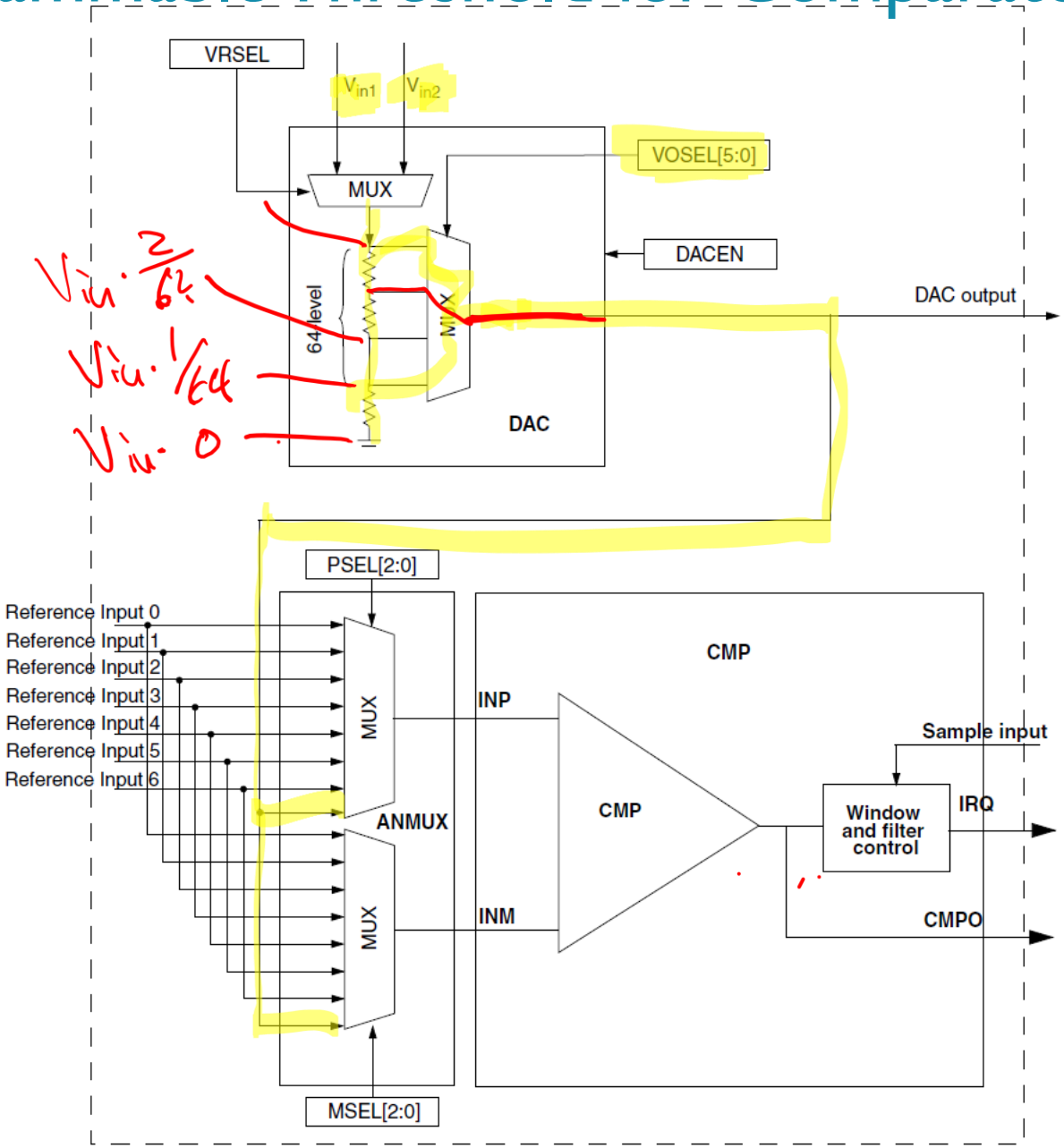


Comparator Interrupt

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

Programmable Threshold for Comparator



- Comparator has 6-bit DAC
- Can use DAC to set threshold voltage for comparator
- Supports 64 different threshold voltages

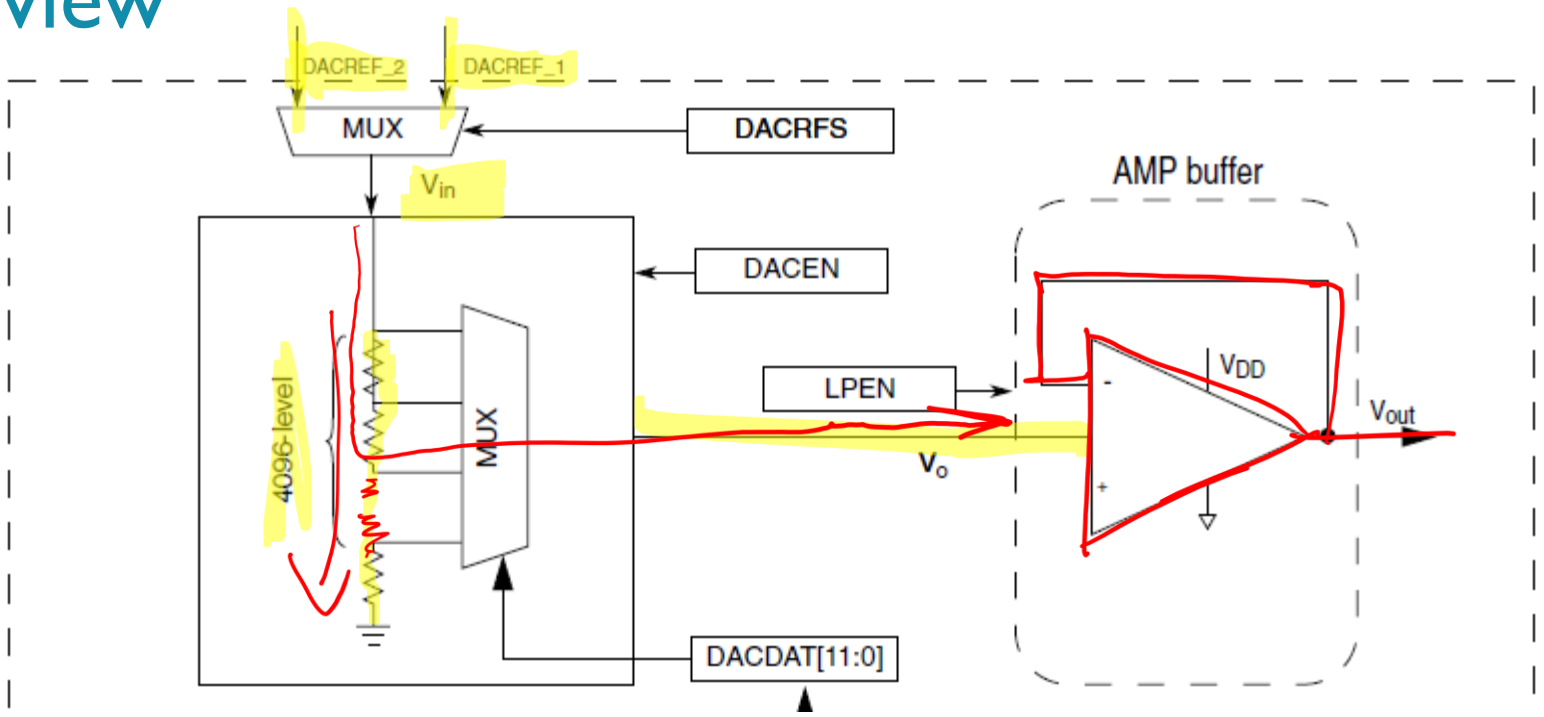
DAC Control Register CMPx_DACCR

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

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 Registers

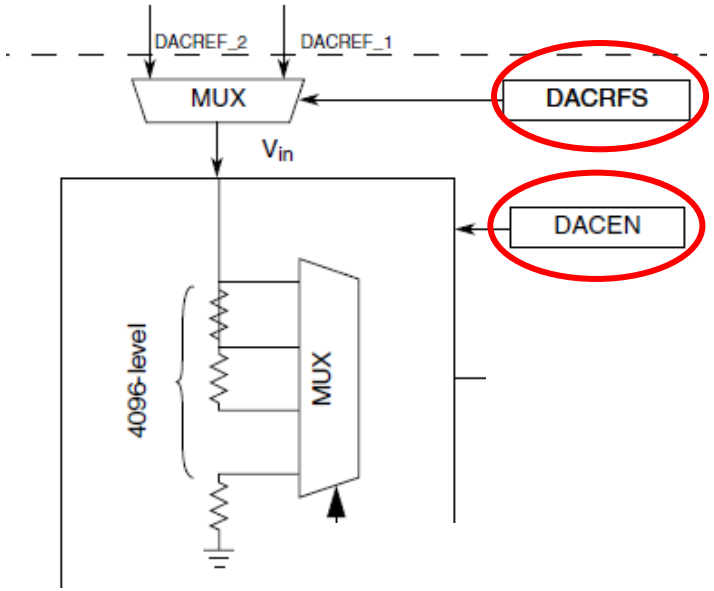
DAC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_F000	DAC Data Low Register (DAC0_DAT0L)	8	R/W	00h	30.4.1/531
4003_F001	DAC Data High Register (DAC0_DAT0H)	8	R/W	00h	30.4.2/532
4003_F002	DAC Data Low Register (DAC0_DAT1L)	8	R/W	00h	30.4.1/531
4003_F003	DAC Data High Register (DAC0_DAT1H)	8	R/W	00h	30.4.2/532
4003_F020	DAC Status Register (DAC0_SR)	8	R	02h	30.4.3/532
4003_F021	DAC Control Register (DAC0_C0)	8	R/W	00h	30.4.4/533
4003_F022	DAC Control Register 1 (DAC0_C1)	8	R/W	00h	30.4.5/534
4003_F023	DAC Control Register 2 (DAC0_C2)	8	R/W	0Fh	30.4.6/534

- This peripheral's registers are only eight bits long. Legacy peripheral?
- DATA[11:0] stored in two registers
  - DATA0: Low byte [7:0] in DACx_DATnL
  - DATA1: High nibble [11:0] in DACx_DATnH

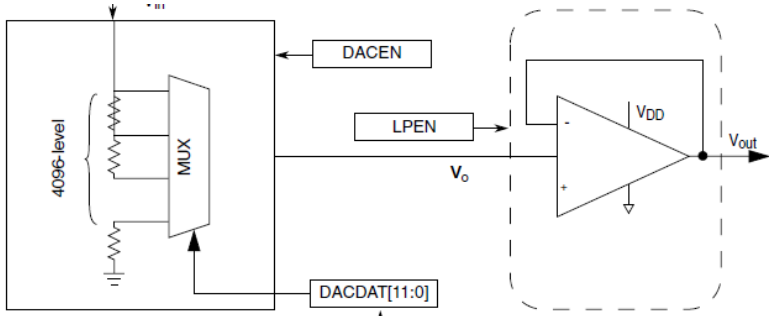
DAC Control Register 0: DACx_C0

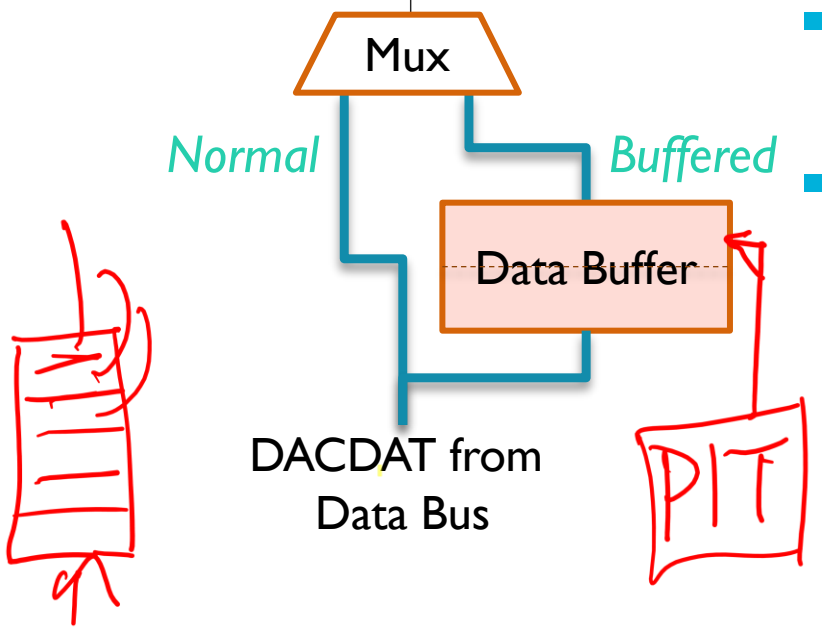
Bit	7	6	5	4	3	2	1	0
Read	DACEN	DACRFS	DACTRGSEL	0	LPEN	0	DACBTIEN	DACBBIEN
Write				DACSWTRG				
Reset	0	0	0	0	0	0	0	0

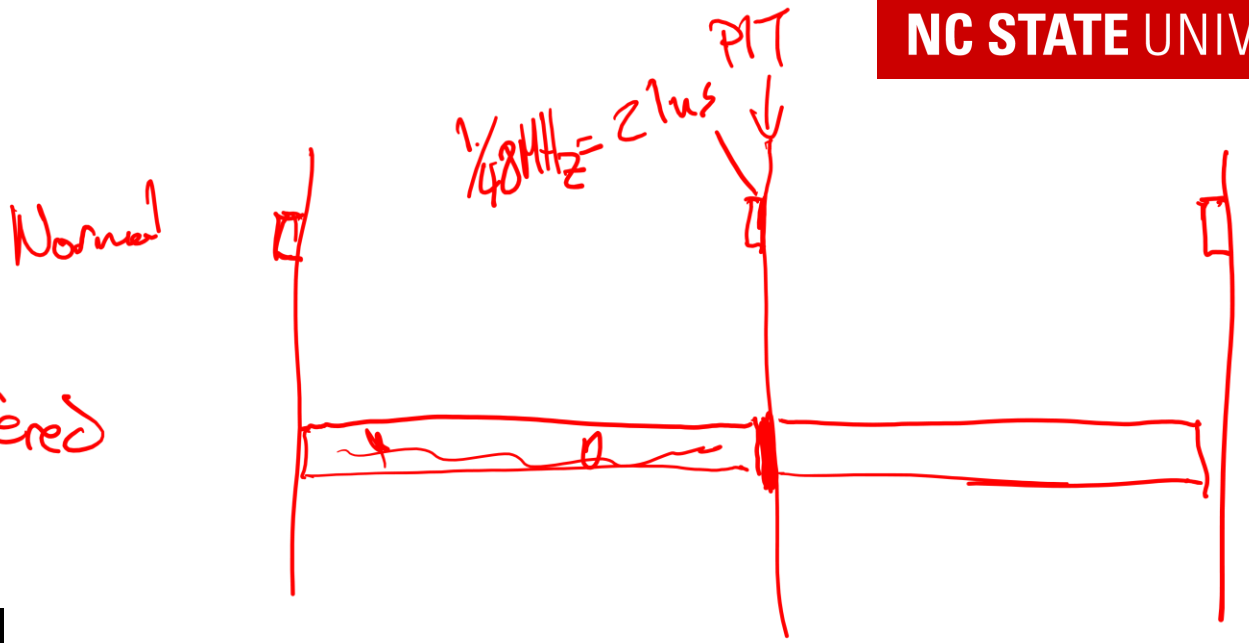


- DACEN - DAC Enabled when 1
- DACRFS - DAC reference voltage select
  - 0: DACREF_1. Connected to VREFH
  - 1: DACREF_2. Connected to VDDA
- LPEN - low-power mode
  - 0: High-speed mode. Fast (15 us settling time) but uses more power (up to 900 uA supply current)
  - 1: Low-power mode. Slow (100 us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

DAC Operating Modes

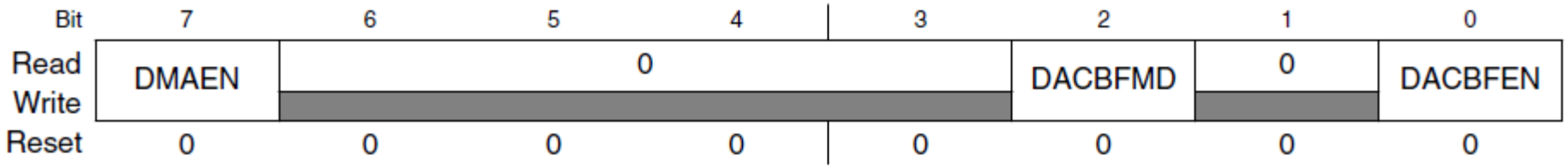






- Normal
  - Value written to DACDAT is converted to voltage immediately
- Buffered mode eases timing requirements
  - Value written to DACDAT is stored in data buffer for later conversion
  - Next data item is sent to DAC when triggered
    - Software Trigger - write to DACSWTRG field in DACx_C0
    - Hardware Trigger - from PIT timer peripheral
  - Normal Mode: Circular buffer
  - One-time Scan Mode: Pointer advances, stops at end of buffer
  - Status flags in DACx_SR

DAC Control Register 1: DACx_C1

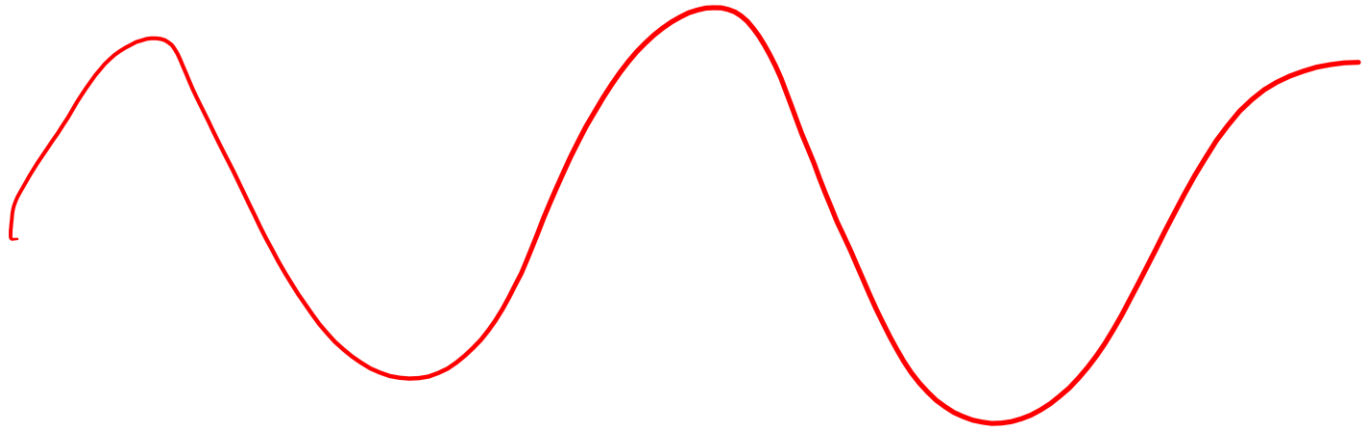


Bit	7	6	5	4	3	2	1	0
Read	DMAEN	0				DACBFMD	0	DACBFEN
Write								
Reset	0	0	0	0	0	0	0	0

- DACBFEN
  - 0: Disable buffer mode
  - 1: Enable buffer mode
- DACBFMD - Buffer mode select
  - 0: Normal mode (circular buffer)
  - 1: One-time scan mode

Example: Waveform Generator

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

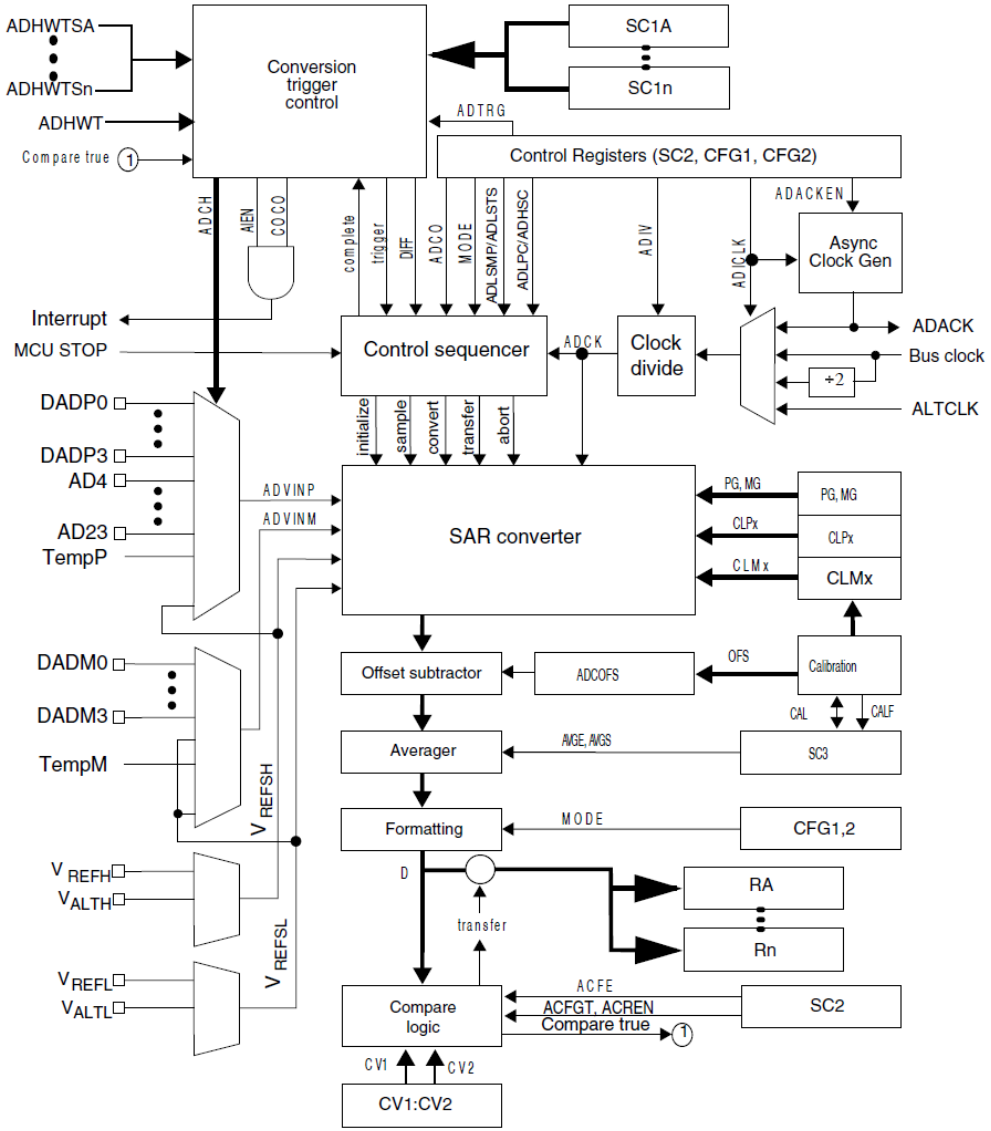


ANALOG TO DIGITAL CONVERTER

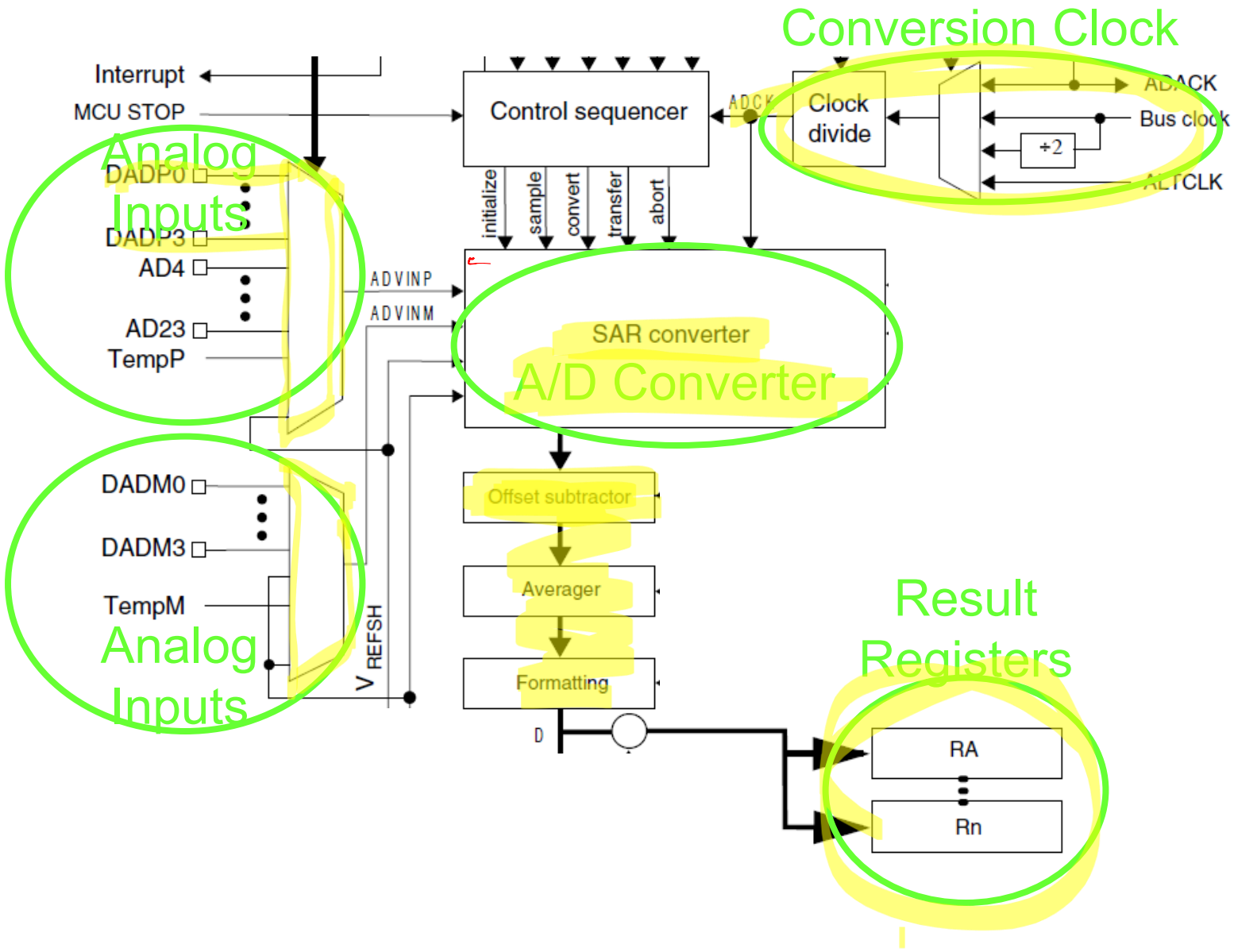
ADC Overview

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

ADC System Overview



ADC System Fundamentals



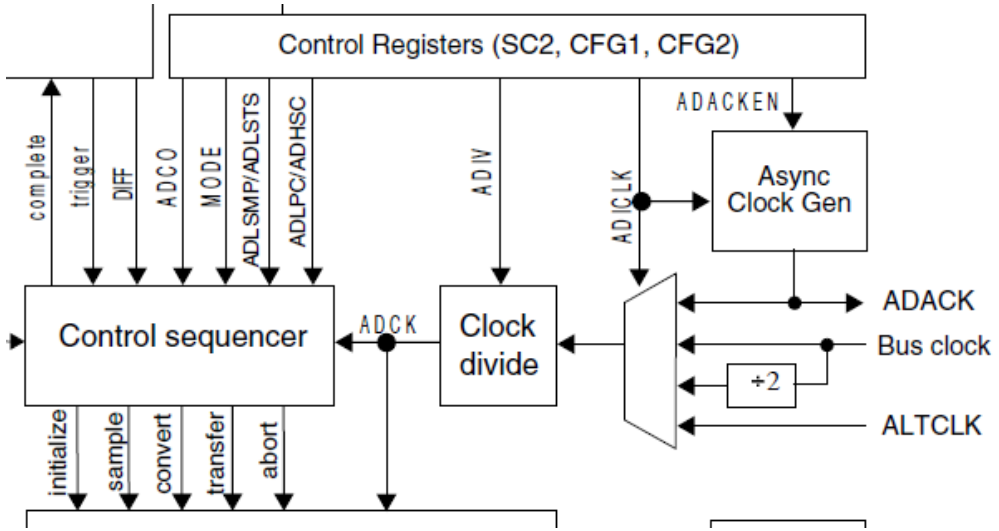
Using the ADC

- ADC initialization
  - Configure clock
  - Select voltage reference
  - Select trigger source
  - Select input channel
  - Select other parameters
- Trigger conversion
- Read results

Init Trigger ..... -> Done, Read results

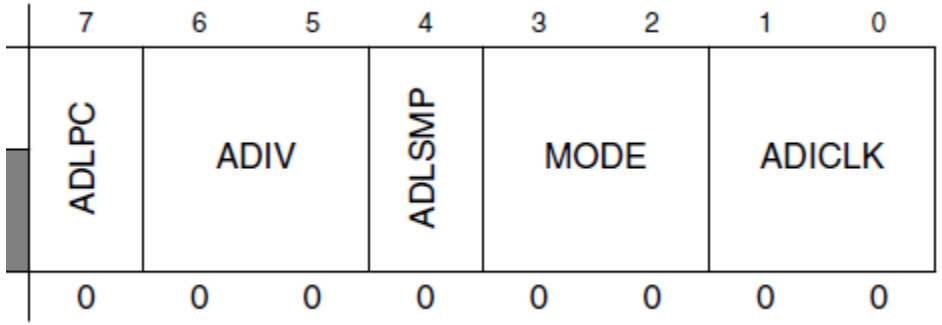
Clock Configuration

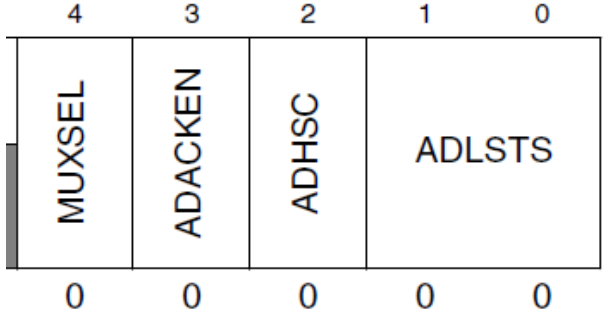
- Select clock source with ADICLK
  - Bus Clock (default)
  - ADACK: Local clock, allows ADC operation while rest of CPU is in stop mode
  - ALTCLK: alternate clock (MCU-specific)
- Divide down selected clock by factor of ADIV, creating ADCK
- Resulting ADCK must be within valid range to ensure accuracy (See KL25 Subfamily datasheet)
  - 1 to 18 MHz (<= 13-bit mode)
  - 2 to 12 MHz (16-bit mode)



Clock Configuration Registers

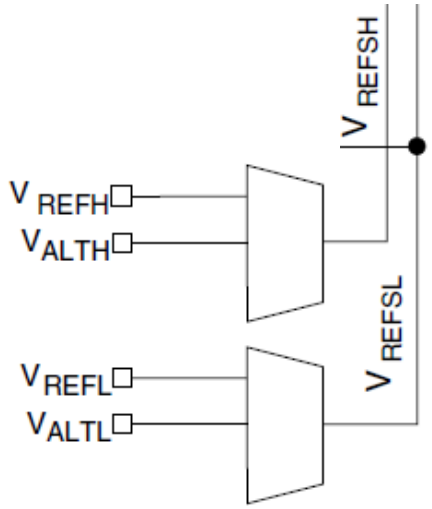
- ADCx_CFG1
  - ADIV: divide clock by 2^{ADIV}
    - 00: 1
    - 01: 2
    - 10: 4
    - 11: 8
  - ADICLK: Input clock select
    - 00: Bus clock
    - 01: Bus clock/2
    - 10: ALTCLK
    - 11: ADACK
- ADCx_CFG2
  - ADACKEN: Enable asynchronous clock





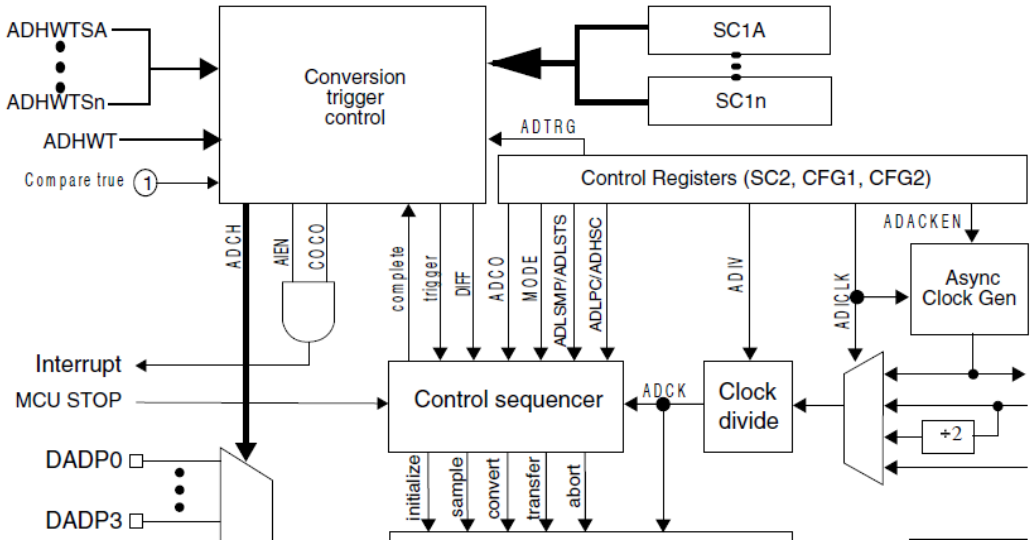
Voltage Reference Selection

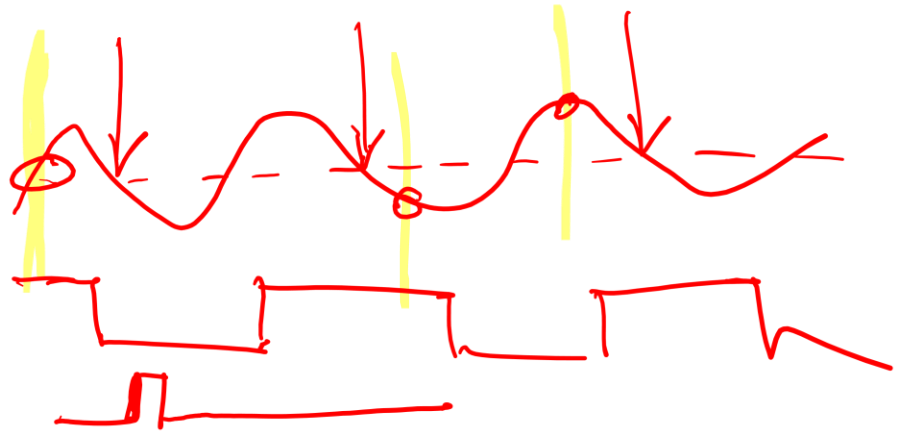
- Two voltage reference pairs available
  - V_{REFH}, V_{REFL}
  - V_{ALTH}, V_{ALTL}
- Select with SC2 register's REFSEL bits
  - 00: V_{REFH}, V_{REFL}
  - 01: V_{ALTH}, V_{ALTL}
  - 10, 11: Reserved
- KL25Z
  - V_{ALTH} connected to V_{DDA}



Conversion Trigger Selection

- ADTRG in SC2
  - 0: software trigger
  - 1: hardware trigger
- Software trigger:
  - Write to SC1A
- Ping-pong buffering
  - SC1A vs. SC1n
- Hardware trigger:
  - Rising edge of ADHWT signal
  - ADHWT sources: TPM, LPTMR, PIT, RTC, EXTRG_IN, HSCMP0



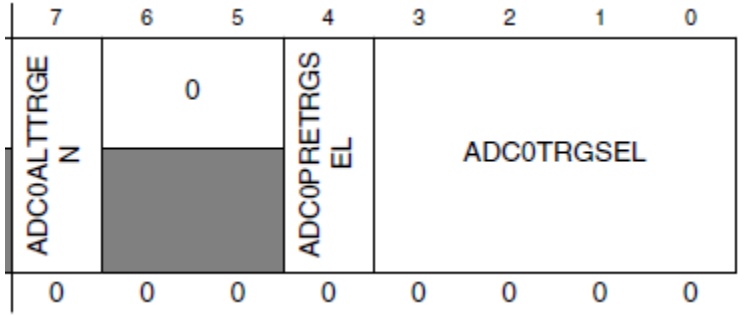


Hardware Trigger Sources

- System Integration Module
  - SIM_SOPT7 register
  - See section 12.2.6 of Reference Manual

- ADC0ALTTRGEN: Alternate trigger enable

- ADC0PRETRGSEL: ADC pre-trigger select

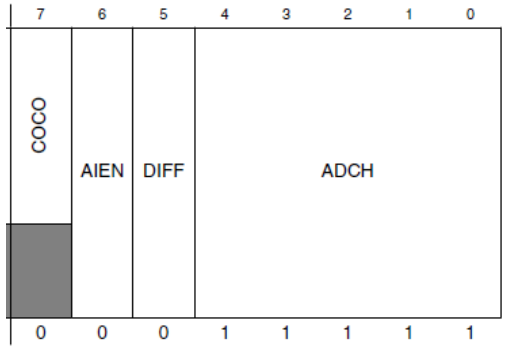


3–0 ADC0TRGSEL	ADC0 trigger select Selects the ADC0 trigger source

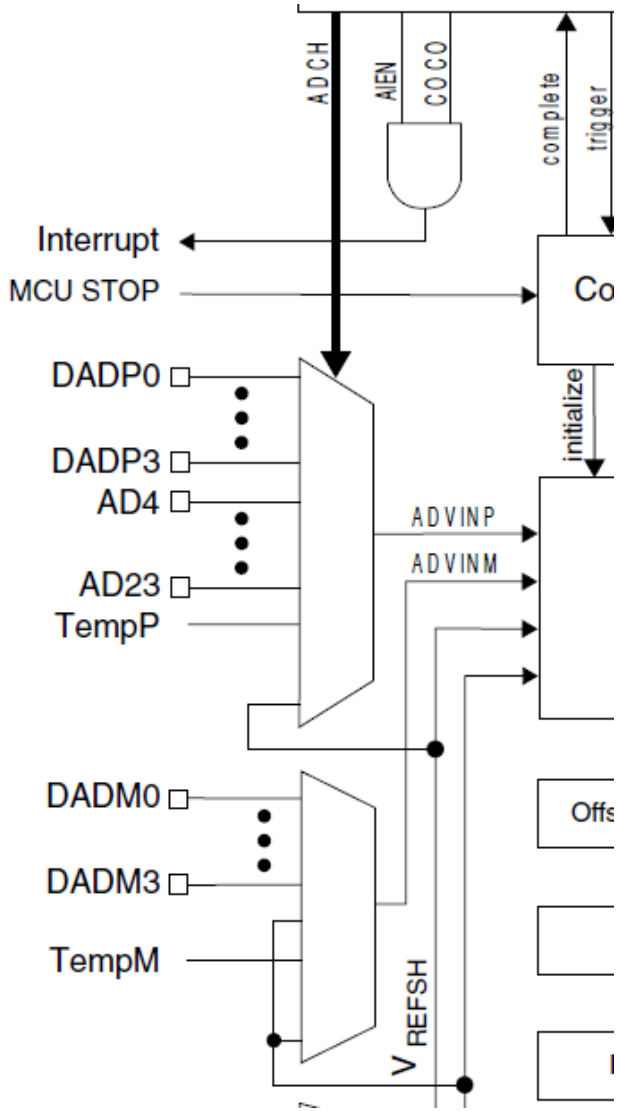
0000	External trigger pin input
0001	HSCMP0 output
0010	Reserved
0011	Reserved
0100	PIT trigger 0
0101	PIT trigger 1
0110	Reserved
0111	Reserved
1000	TPM0 overflow
1001	TPM1 overflow
1010	TPM2 overflow
1011	Reserved
1100	RTC alarm
1101	RTC seconds
1110	LPTMR0 trigger
1111	Reserved

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

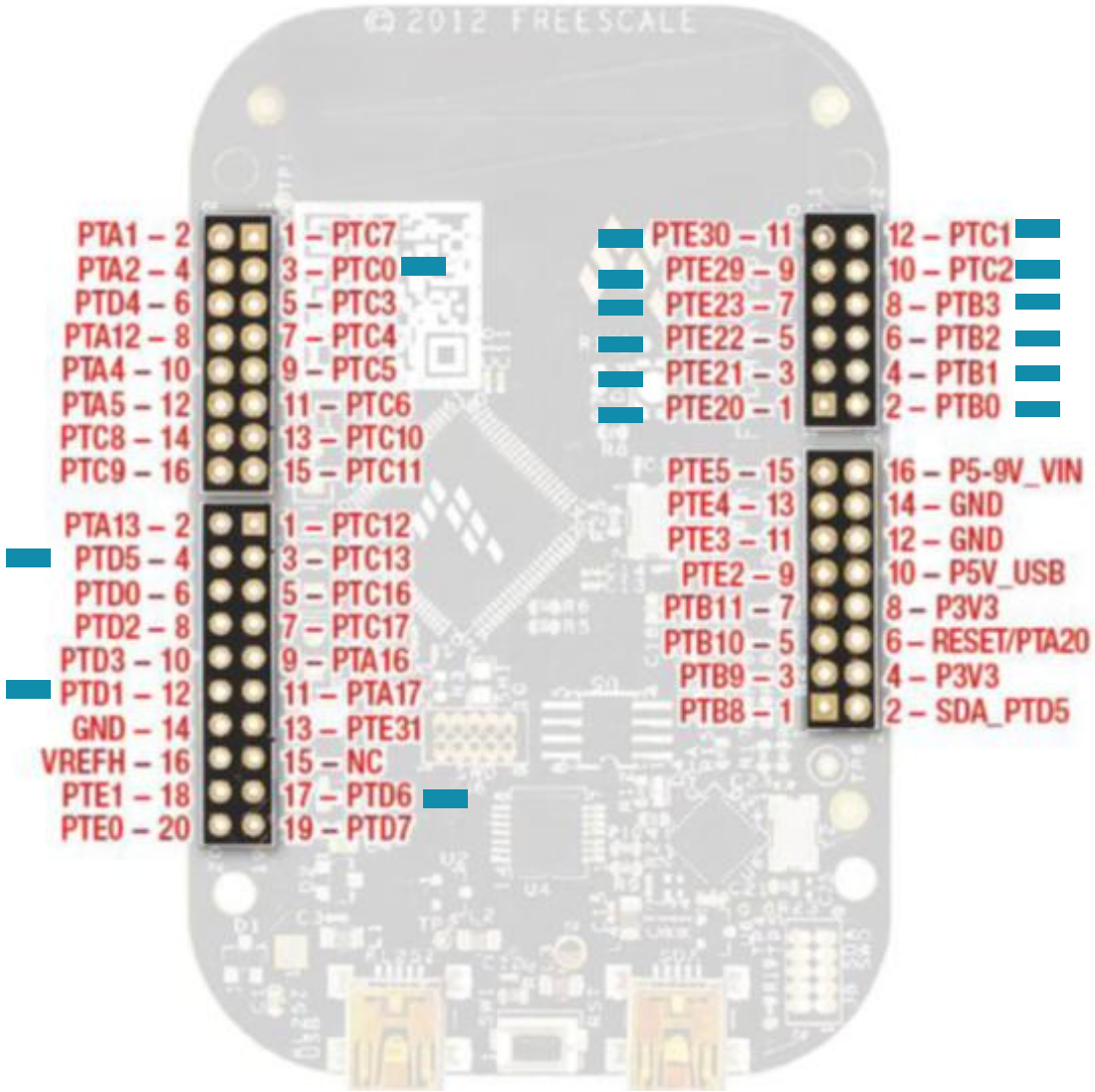


	DIFF	
ADCH	0	1
0	DADP0	DAD0
1	DADP1	DAD1
2	DADP2	DAD2
3	DADP3	DAD3
4	AD4	reserved
...	...	...
23	AD23	reserved
24	reserved	reserved
25	reserved	reserved
26	Temp Sensor	
27	Band Gap	
28	reserved	reserved
29	VREFSH	-VREFSH
30	VREFSL	reserved
31	module disabled	



ADC Inputs on Freedom Board

ADC Channel (Single-Ended)	MCU Signal (ADCx_CFG2 MUXSEL, default a)	Freedom KL25Z Connector and Pin Number
0	PTE20	J10 1
3	PTE22	J10 5
4	PTE21 (a), PTB29 (b)	J10 3 (a), J10 9 (b)
5	PTD1 (b)	J2 12 (b)
6	PTD5 (b)	J2 4 (b)
7	PTE23 (a), PTD6 (b)	J10 7 (a), J2 17(b)
8	PTB0	J10 2
9	PTB1	J10 4
11	PTC2	J10 10
12	PTB2	J10 6
13	PTB3	J10 8
14	PTC0	J1 3
15	PTC1	J10 12
23	PTE30	J10 11

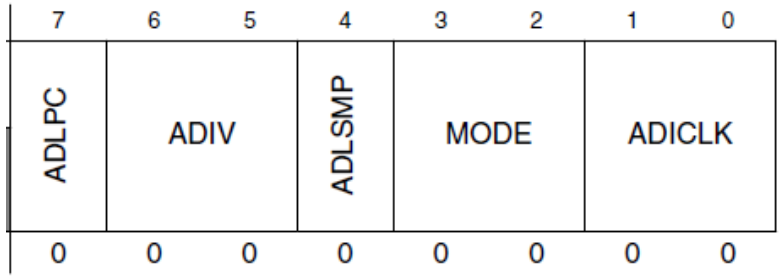


Special Input Channels

- 26: Temperature Sensor
  - $T = 25°C - ((V_{Temp} - V_{Temp25})/m)$
  - $m = 1.715$ mV/°C
  - $V_{TEMP25} = 719$ mV
  - V_{Temp} is derived from ADC conversion result
- 27: Band Gap
  - Nominally 1.0 V, +/- 3%
  - Used to allow calculation of reference voltage if it is not calibrated/regulated
  - Enable BG buffer by setting BGBE bit in Power Management Control REGSC
  - "My ADC tells me that channel 27 returns a code of 36000, and channel 0 returns a code of 25145. So V_0 = 36000/25145 * 1.0 V = 1.432 V
- 29, 30: Reference voltages
  - 29: V_{REFSH}
  - 30: V_{REFSL}

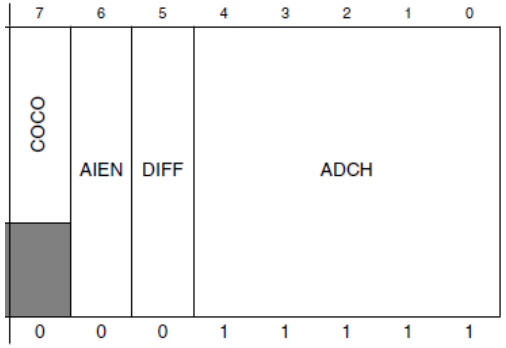
Conversion Options Selection

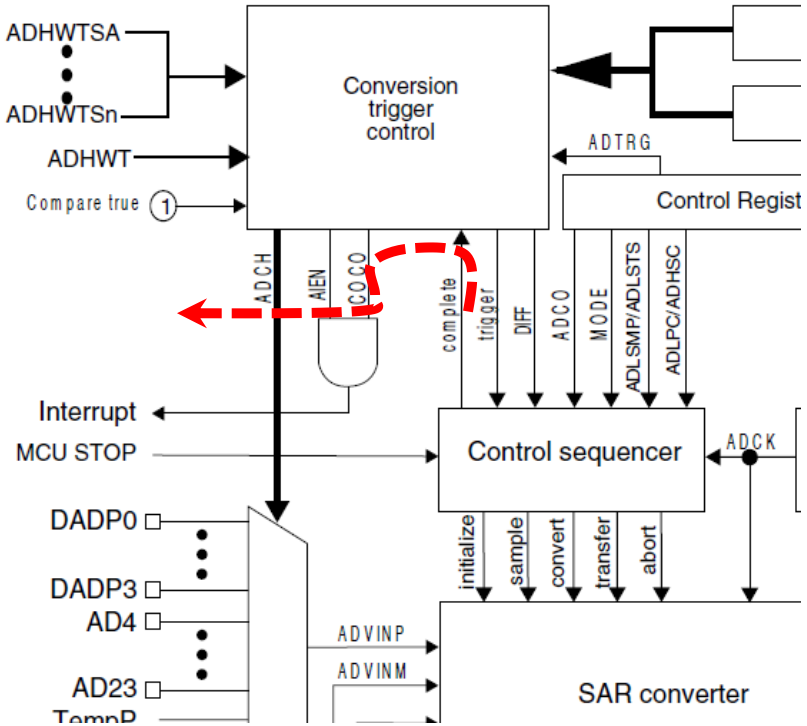
- Low power
  - Set ADLPC (in ADCx_CFG1) to 1
  - Slower max clock speed
- Long sample time select
  - Set ADLSMP (in ADCx_CFG1) to 1
  - Can select longer sample time with ADLSTS bits (in ADCx_CFG2) to add 20, 16, 10 or 6 ADCK cycles)
- Conversion mode
  - MODE (in ADCx_CFG1)
  - Sets result precision (8 through 16 bits)
- Continuous vs. single conversion
  - Set ADCO (in ADCx_SC3) to 1 for continuous conversions



	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

Conversion Completion



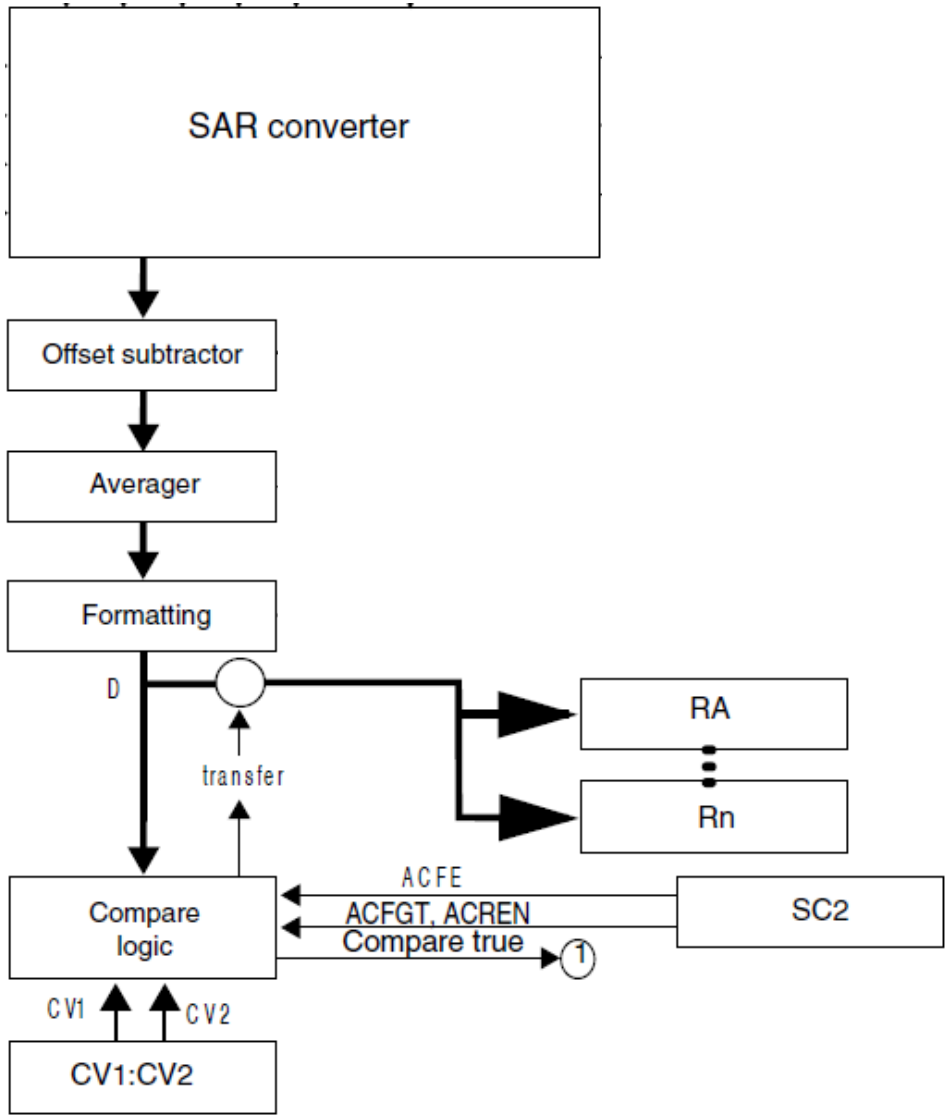


- Signaled by COCO bit in SC1n

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

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 Rn
  - Conversion result goes into register corresponding to SC1 register used to start conversion (SC1A, SC1n)



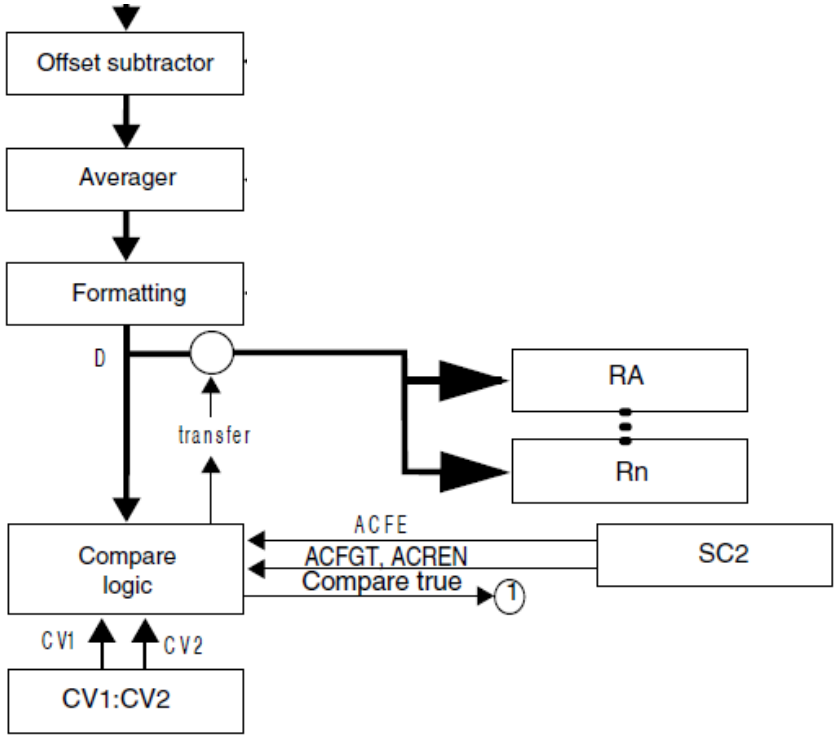
Output Averaging

- Accumulate and average multiple samples before writing the averaged result to the result register
- Result rate = sample rate / averaging factor
- 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



Conversion Time Requirements

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder} + \text{HSCAdder})$$

- Conversion time depends on multiple factors
  - Precision in bits
  - Single-ended vs. differential
  - Adders: LST, HSC, SFC
  - Averaging

Precision (bits)	Mode	Base Conversion Time BCT
8	single-ended	17 ADCK cycles
9	differential	27
10	single-ended	20
11	differential	30
12	single-ended	20
13	differential	30
16	single-ended	25
16	differential	34

Using ADC Values

- The ADC gives an integer representing the input voltage relative to the reference voltages
- Several conversions may be needed
  - For many applications you will need to compute the approximate input voltage
    - V_{in} = …
  - For some sensor-based applications you will need to compute the physical parameter value based on that voltage (e.g. pressure) – this depends on the sensor's transfer function
  - You will likely need to do additional computations based on this physical parameter (e.g. compute depth based on pressure)
- Data type
  - It's likely that doing these conversions with integer math will lead to excessive loss of precision, so use floating point math
  - AFTER you have the application working, you can think about accelerating the program using fixed-point math (scaled integers).
- Sometimes you will want to output ASCII characters (to the LCD, for example). You will need to convert the floating point number to ASCII using sprintf, ftoa, or another method.