# Lecture 12: STM32 Analog Interfacing

Seyed-Hosein Attarzadeh-Niaki

Based on the slides by ARM



## Review

- Overview of timer modes
- Pulse-width modulation
- Direct Memory Access

# Outline

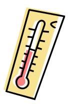
- ADC Basics
  - Concept
  - Characteristics
- Converting between analog and digital values
- STM32F4 analog interfacing peripherals
  - Analog-to-digital converter
  - Analog watchdog
  - Digital-to-analog converter
  - DAC using PWM



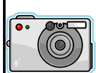
# ADC BASICS

# Why It's Needed

- Embedded systems often need to measure values of physical parameters
- A physical quantity is converted to electrical (voltage, current) signals using a device called a *transducer* or *sensor*.



- **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

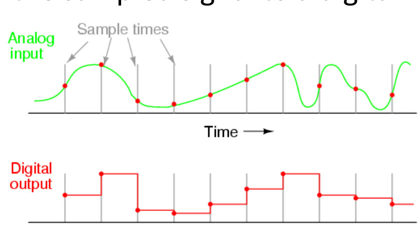
- **Pressure**
  - Blood pressure monitor
  - Altimeter
  - Car engine controller
  - Scuba dive computer
  - Tsunami detector

- **Acceleration**
  - Air bag controller
  - Vehicle stability
  - Video game remote
- **Mechanical strain**
- Other
  - Touch screen controller
  - EKG, EEG
  - Breathalyzer

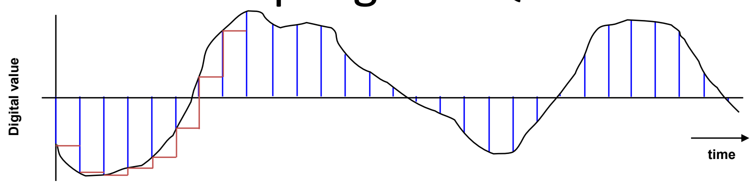


# Analog to Digital Conversion

- Digital computers use *binary (discrete)* values, but in the physical world signals are *analog (continuous)*.
- An analog-to-digital converter (ADC or A/D) translates the analog signals to digital numbers so that computers can read and process them.
  - An ADC samples an analogue signal at discrete times and
  - Discretizes the sampled signal to a digital value.



# Waveform Sampling and Quantization




- A waveform is **sampled** at a constant rate – every $\Delta t$
  - Each such sample represents the instantaneous amplitude at the instant of sampling
  - *"At 37 ms, the input is 1.91341914513451451234311… 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 1.91341914513451451234311… V is best represented by the code 0x018, since it is in the range of 1.901 to 1.9980 V which corresponds to code 0x018."*

# CONVERTING BETWEEN ANALOG AND DIGITAL VALUES

# The Big Picture – A Depth Gauge

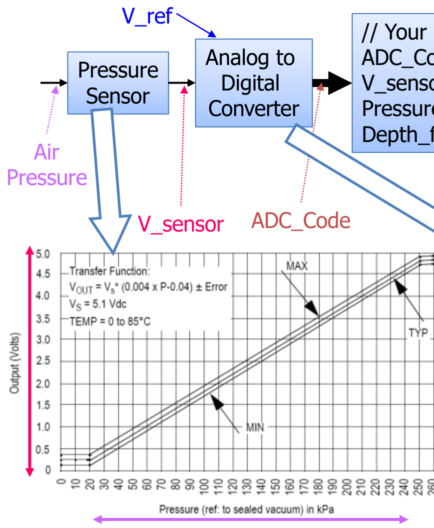
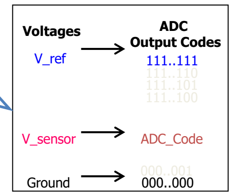



Figure 4. Output vs. Absolute Pressure

```
// 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;
```



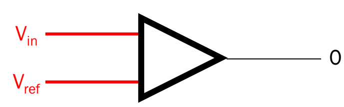

1. Sensor detects *air pressure* and generates a proportional *output voltage* V_sensor
2. ADC generates a proportional digital *integer* (code) based on V_sensor and V_ref
3. Code can convert that integer to a something more useful
   1. first a float representing the *voltage*,
   2. then another float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



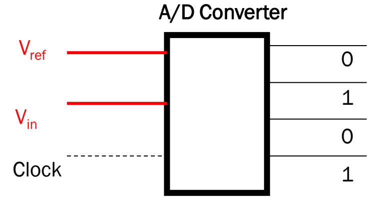
# Getting From Analog to Digital

- 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

Comparator




A/D Converter



# Forward Transfer Function Equations

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

***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

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

***Simplification with $V_{-ref}$ = 0 V***

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

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

$\lfloor X \rfloor = I$ *floor function: nearest integer I such that I <= X*
*floor(x+0.5) rounds x to the nearest integer*



# Inverse Transfer Function

*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***

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

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

## 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 $V_{ref}$

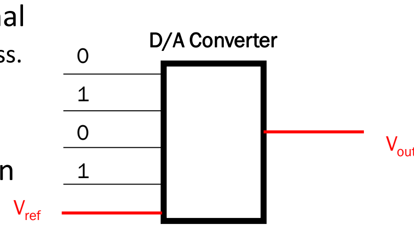
$$V_{ref} = V_{known} \frac{2^N}{n}$$

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



## 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

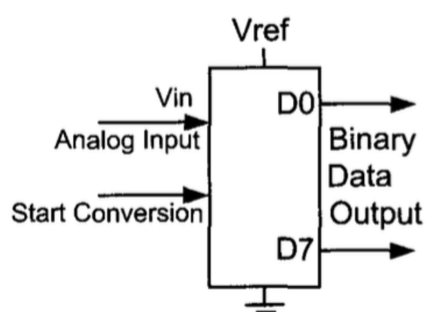


# ADC CHARACTERISTICS



# Resolution

- An ADC has n-bit resolution.
  - n can be 8, 10, 12, 16, or even 24 bits.
- Higher-resolution ADCs provide a smaller *step size*
  - Step size is the smallest change that can be measured by an ADC.
- Resolution is fixed for an ADC in design time
- Step size can be changed by $V_{ref}$.




| n-bit | Number of steps | Step size (mV) $V_{ref}$=5V |
|---|---|---|
| 8 | 256 | 5/256 = 19.53 |
| 10 | 1024 | 5/1024 = 4.88 |
| 12 | 4096 | 5/4096= 1.2 |
| 16 | 65,536 | 5/65,536 = 0.076 |

# Reference Voltage ($V_{ref}$)

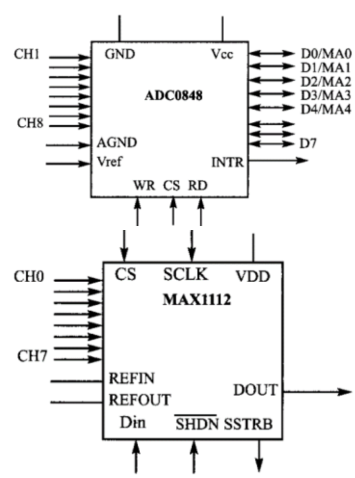
- $V_{ref}$ is an input voltage used for the *reference voltage.*
- $V_{ref}$ and the ADC resolution together dictate the step size.
  - For an *8-bit ADC*, the step size is $V_{ref}/256$ because it is an 8-bit ADC, and $2^8$ = 256 steps.
  - If the analog input range is 0 to 4 volts, then $V_{ref}$ is connected to 4 volts => 4 V/256 = 15.62 step size

| $V_{ref}$ (V) | $V_{in}$ Range (V) | Step Size (mV) |
|---|---|---|
| 5.00 | 0 to 5 | 5/256 = 19.53 |
| 4.0 | 0 to 4 | 4/256 = 15.62 |
| 3.0 | 0 to 3 | 3/256 = 11.71 |
| 2.56 | 0 to 2.56 | 2.56/256 = 10 |
| 2.0 | 0 to 2 | 2/256 = 7.81 |
| 1.28 | 0 to 1.28 | 1.28/256 = 5 |
| 1 | 0 to 1 | 1/256 = 3.90 |



# ADC Inputs and Outputs

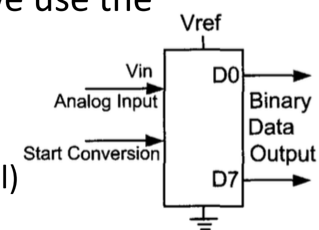
- Parallel vs. serial ADC
  - How digital output data is available
- Analog input channels
  - Differential
    - Use two channels, and compute difference between them
    - Very good *noise immunity*
    - Some sensors offer differential outputs
  - Multiplexing
    - Typically share a single ADC among 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



# Digital Data Output

- In an 8-bit ADC we have an 8-bit digital data output of D0-D7, while in the 10-bit ADC the data output is D0-D9.
- To calculate the output voltage, we use the following formula:

$$D_{out} = V_{in} / \text{step size}$$

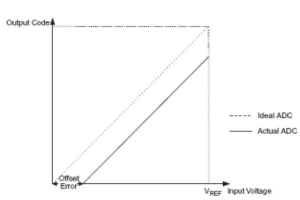



- Where
  - $D_{out}$ = digital data output (in decimal)
  - $V_{in}$ = analog input voltage
  - step size = (resolution) is the smallest change, which is ( $V_{ref}$ / 256 ) for an 8-bit ADC.

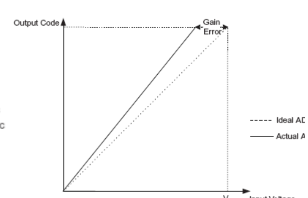


# ADC Errors

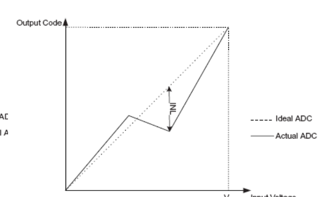
**Types of Conversion Errors:**




Offset error



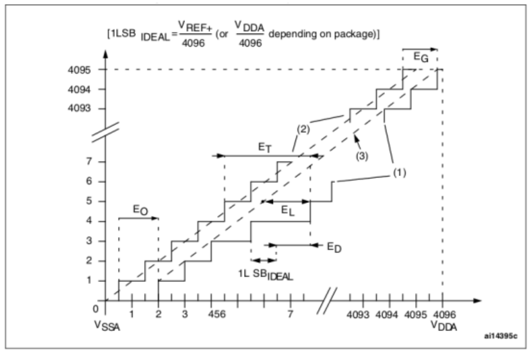

Gain error




Non-linearity

- Linearity measures how well the transition voltages lie on a straight line.
- Differential linearity measure the equality of the step size.

# STM32F401 Accuracy



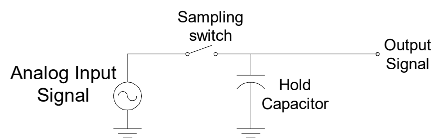

| Symbol | Parameter | Test conditions | Typ | Max[2] | Unit |
|---|---|---|---|---|---|
| ET | Total unadjusted error | $f_{ADC}$ = 18 MHz<br>$V_{DDA}$ = 1.7 to 3.6 V<br>$V_{REF}$ = 1.7 to 3.6 V<br>$V_{DDA} - V_{REF} < 1.2$ V | ±3 | ±4 | LSB |
| EO | Offset error | | ±2 | ±3 | |
| EG | Gain error | | ±1 | ±3 | |
| ED | Differential linearity error | | ±1 | ±2 | |
| EL | Integral linearity error | | ±2 | ±3 | |



# Conversion Time

- Conversion time is defined as “*the time it takes for the ADC to convert the analog input to a digital (binary) number*”.
  - Limits the sampling rate
- Dictated by
  - The clock source connected to the ADC
  - The method used for data conversion
    - More than 10 methods
  - The technology used in fabrication of the ADC chip
- Nyquist criterion
  - $F_{sample} >= 2 * F_{max}$ frequency component
    - Frequency components above ½ $F_{sample}$ are aliased, distort measured signal
  - In practice, the sampling rate must be higher

# Sample and Hold Devices



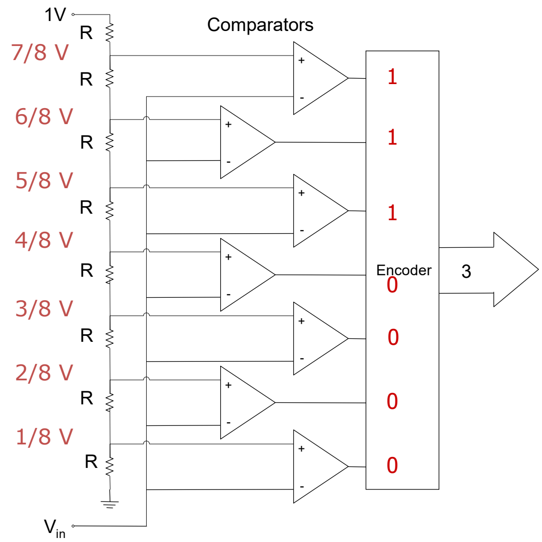

- Some A/D converters require the input analog signal to be held constant during conversion, (e.g. successive approximation devices)
- In other cases, peak capture or sampling at a specific point in time necessitates a sampling device.
- This function is accomplished by a sample and hold device as shown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



# ADC CONVERSION METHODS

# A/D – Flash Conversion

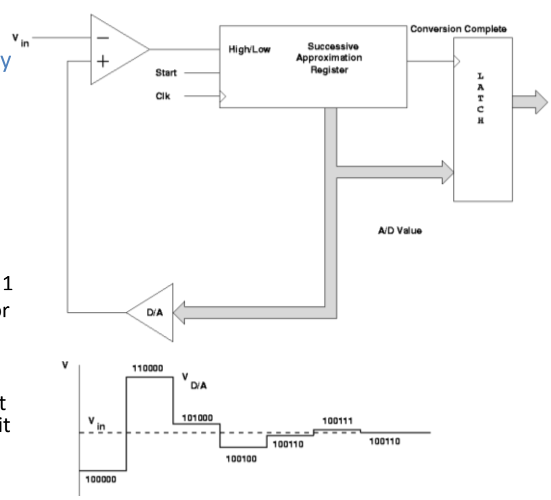
- A multi-level voltage divider is used to set voltage levels over the complete range of conversion.
- A comparator is used at each level to determine whether the voltage is lower or higher than the level.
- The series of comparator outputs are encoded to a binary number in digital logic (a priority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





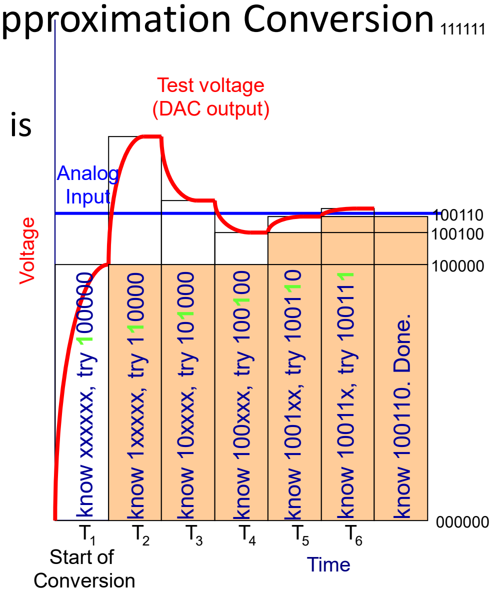

# Successive Approximation ADC

- 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



## ADC - Successive Approximation Conversion

- After n bits, the result is latched out






# STM32F4XX ANALOG INTERFACING PERIPHERALS

# GPIO Configuration Registers

- Mode 11
  - GPIOA->MODER |= GPIO_MODER_MODERx;
- PUPD 00
  - GPIOA->PUPDR &= ~(GPIO_PUPDR_PUPDRx);

| MODER(i) [1:0] | OTYPER(i) | OSPEEDR(i) [B:A] | | PUPDR(i) [1:0] | | I/O configuration | |
|---|---|---|---|---|---|---|---|
| 01 | 0 | SPEED [B:A] | | 0 | 0 | GP output | PP |
| | 0 | | | 0 | 1 | GP output | PP + PU |
| | 0 | | | 1 | 0 | GP output | PP + PD |
| | 0 | | | 1 | 1 | Reserved | |
| | 1 | | | 0 | 0 | GP output | OD |
| | 1 | | | 0 | 1 | GP output | OD + PU |
| | 1 | | | 1 | 0 | GP output | OD + PD |
| | 1 | | | 1 | 1 | Reserved (GP output OD) | |
| 10 | 0 | SPEED [B:A] | | 0 | 0 | AF | PP |
| | 0 | | | 0 | 1 | AF | PP + PU |
| | 0 | | | 1 | 0 | AF | PP + PD |
| | 0 | | | 1 | 1 | Reserved | |
| | 1 | | | 0 | 0 | AF | OD |
| | 1 | | | 0 | 1 | AF | OD + PU |
| | 1 | | | 1 | 0 | AF | OD + PD |
| | 1 | | | 1 | 1 | Reserved | |
| 00 | x | x | x | 0 | 0 | Input | Floating |
| | x | x | x | 0 | 1 | Input | PU |
| | x | x | x | 1 | 0 | Input | PD |
| | x | x | x | 1 | 1 | Reserved (input floating) | |
| 11 | x | x | x | 0 | 0 | Input/output | Analog |
| | x | x | x | 0 | 1 | Reserved | |
| | x | x | x | 1 | 0 | | |
| | x | x | x | 1 | 1 | | |

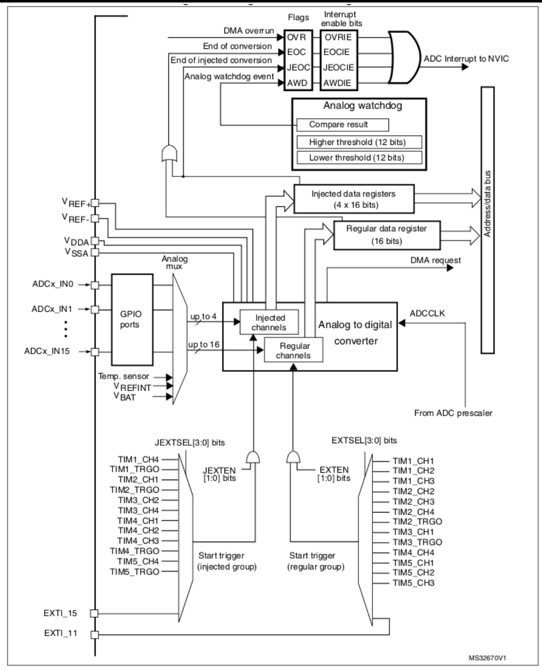
1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.



# ANALOG TO DIGITAL CONVERTER

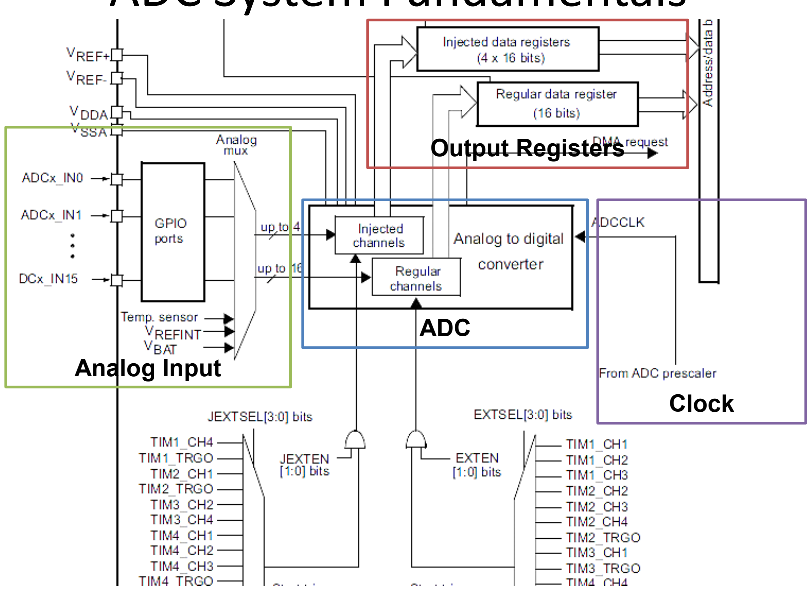
### ADC Overview

- Uses successive approximation for conversion
- Supports multiple resolutions: 12, 10, 8 and 6 bits
- 4 injected channels and 16 regular channels
- Supports single and continuous conversions
- Left/right output data alignment
- DMA
- Analog watchdog
- Temperature sensor
- Conversion range from 0 to 3.6 V
- Different Supply requirement
- Scan mode for automatic conversion of different channels
- Channel by channel programmable sampling time
- Interrupt generation on
  - End of (Injected) conversion
  - Analog watchdog
  - Overrun






# ADC System Fundamentals



# Using the ADC

- ADC initialization
  - Configure GPIO (if using on board pins)
  - Enable clock (in RCC_APB2ENR )
  - Enable ADC
  - Select voltage reference
  - Select trigger source
  - Select input channel
  - Select other parameters
- Trigger conversion
- Read results
- Calibrate? Average?



# On-off Control

- For *power efficiency*, the ADC module is usually turned off (even if it is clocked).
- If **ADON** bit in ADC control register 2 (**ADC_CR2**) is set, the module is powered on; otherwise it is powered off.
- Good practice to shut down ADC whenever you are not using it.

# Clock Configuration

**Analog Clock**

- ADCCLK, common to all ADCs
- From APB2 (Can be prescaled by 2, 4, 6 or 8)
- Refer to datasheet for maximum clock frequency
- ADC common control register (ADC_CCR) bit 17:16

**Digital Interface Clock**

- Used for registers read/write access
- From APB2
- Need to be enabled individually for each channel (RCC_APB2ENR)



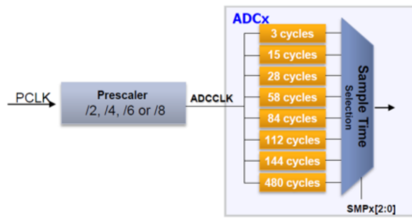
# Common Control Register

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | | TSVREFE | VBATE | Reserved | | | | ADCPRE | |
| | | | | | | | | rw | rw | | | | | rw | rw |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | | | | | | | | | | | | | | | |

| Bit | Field | Descriptions |
|---|---|---|
| 23 | TSVREFE: | Temperature Sensor and VREINT Enable<br>This bit is set and cleared by software to enable/disable the temperature sensor and the VREFINT channel.<br>0: Temperature sensor and VREFINT channel disabled.<br>1: Temperature sensor and VREFINT channel enabled. |
| 22 | VBATE: VBAT enable | VBAT enable<br>0: VBAT channel disabled.<br>1: VBAT channel enabled. |
| 17:16 | | ADCPRE: ADC prescaler<br>Set and cleared by software to select the frequency of the clock to the ADC. The clock is common for all the ADCs.<br><br>00: PCLK2 divided by 2<br>01: PCLK2 divided by 4<br>10: PCLK2 divided by 6<br>11: PCLK2 divided by 8 |

# ADC Conversion Time

- Programmable sample time for all channels
- Sample time register 1 to 2 (ADC_SMPRx)
- Total conversion time = $T_{sampling} + T_{conversion}$




| Resolution | $T_{Conversion}$ |
|---|---|
| 12 bits | 12 Cycles |
| 10 bits | 10 Cycles |
| 8 bits | 8 Cycles |
| 6 bits | 6 Cycles |

With Sample time= 3 cycles @ ADC_CLK = 36MHz ➔ total conversion time is equal to:

| resolution | Total conversion Time | |
|---|---|---|
| **12 bits** | **12 + 3 = 15cycles** | **0.416 us → 2.4 Msps** |
| 10 bits | 10 + 3 = 13 cycles | 0.361 us → 2.71 Msps |
| 8 bits | 8 + 3 = 11 cycles | 0.305 us → 3.27 Msps |
| 6 bits | 6 + 3 = 9 cycles | 0.25 us → 4 Msps |



# ADC sample time registers (ADC_SMPR1/2) to set sampling time

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | SMP18[2:0] | | | SMP17[2:0] | | | SMP16[2:0] | | | SMP15[2:1] | |
| | | | | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

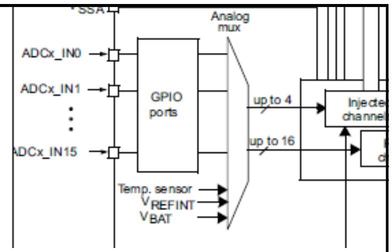
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP15_0 | SMP14[2:0] | | | SMP13[2:0] | | | SMP12[2:0] | | | SMP11[2:0] | | | SMP10[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | SMP9[2:0] | | | SMP8[2:0] | | | SMP7[2:0] | | | SMP6[2:0] | | | SMP5[2:1] | |
| | | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SMP 5_0 | SMP4[2:0] | | | SMP3[2:0] | | | SMP2[2:0] | | | SMP1[2:0] | | | SMP0[2:0] | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

000: 3 cycles
001: 15 cycles
010: 28 cycles
011: 56 cycles
100: 84 cycles
101: 112 cycles
110: 144 cycles
111: 480 cycles

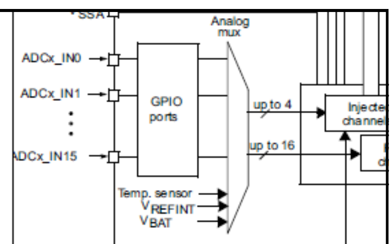
# Channel Selection




- Two groups of channels
  - Regular group
    - Up to 16 conversions
    - Consists of a sequence of conversions that can be done on any channel in any order
    - Specify each sequence by configuring the **ADC_SQRx** registers
    - Specify the total number of conversions by configuring the L[3:0] bits in the ADC_SQR1 register
  - Injected group
    - Up to 4 conversions
    - Similar to regular group
    - But the sequence is specified by the **ADC_JSQR** register
    - Specify the total number of conversions by configuring the JL[1:0] bits in the ADC_JSQR register
  - Modifying either ADC_SQRx or ADC_JSQR will reset the current ADC process.

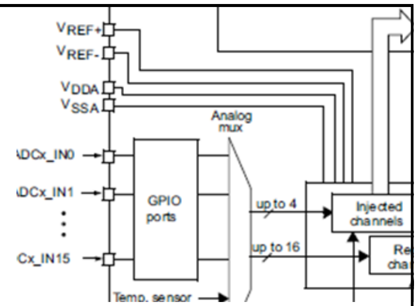


# Channel Selection




- Three other channels
  - ADC1_IN18 is internally connected to the temperature sensor
  - ADC1_IN17 is internally connected to the reference voltage VREFINT
  - ADC1_IN18 is also connected to the VBAT. Can be use as regular or injected channel.
  - But only available on the master ADC1 peripheral.

# Reference Selection

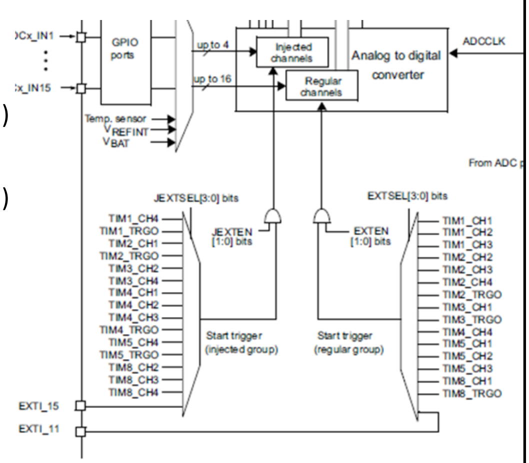



- Input range from $V_{REF-}$ to $V_{REF+}$
- $V_{REF+}$ Positive analog reference
- $V_{DDA}$ equal to $V_{dd}$
- $V_{REF-}$ Negative analog reference, $=V_{SSA}$
- $V_{SSA}$ Grounded and equal to $V_{SS}$
- By default, can convert input range from 0 to 3V



# Conversion Trigger Selection

- Can be triggered by software
  - Setting SWSTART bit in control register 2 (ADC_CR2) for regular group
  - Setting JSWSTART bit in control register 2 (ADC_CR2) for injected group
- Or by external trigger
  - Select the trigger detection mode
  - Specify the trigger event
  - Different bits for specifying regular group and injected group



# ADC control register 2 (ADC_CR2)

- Hardware Trigger Sources and (J)SWSTART bits

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | SWSTART | EXTEN | | EXTSEL[3:0] | | | | reserved | JSWSTART | JEXTEN | | JEXTSEL[3:0] | | | |
| | rw | rw | rw | rw | rw | rw | rw | | rw | rw | rw | rw | rw | rw | rw |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| reserved | | | | ALIGN | EOCS | DDS | DMA | Reserved | | | | | | CONT | ADON |
| | | | | rw | rw | rw | rw | | | | | | | rw | rw |

Bits 27:24 **EXTSEL[3:0]:** External event select for regular group
These bits select the external event used to trigger the start of conversion of a regular group:
0000: Timer 1 CC1 event
0001: Timer 1 CC2 event
0010: Timer 1 CC3 event
0011: Timer 2 CC2 event
0100: Timer 2 CC3 event
0101: Timer 2 CC4 event
0110: Timer 2 TRGO event
0111: Timer 3 CC1 event
1000: Timer 3 TRGO event
1001: Timer 4 CC4 event
1010: Timer 5 CC1 event
1011: Timer 5 CC2 event
1100: Timer 5 CC3 event
1101: Timer 8 CC1 event
1110: Timer 8 TRGO event
1111: EXTI line11

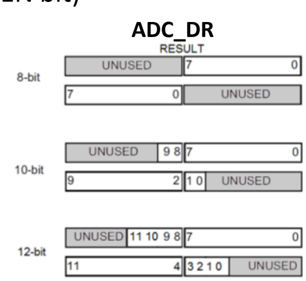
**EXTEN:** External trigger enable for regular channels
These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.
00: Trigger detection disabled
01: Trigger detection on the rising edge
10: Trigger detection on the falling edge
11: Trigger detection on both the rising and falling edges

- Similar for Injected group



# Conversion Options Selection

- Continuous?
  - Single conversion or continuous conversion (CR2 CONT bit)
  - Discontinuous mode available (CR1 DISCEN bit)
- Sample time
- Data alignment
  - CR2 ALIGN




- Scan mode: convert all the channels
  - CR1 SCAN
- Resolution
  - CR1 RES[1:0]

# ADC control register 1 (ADC_CR1)

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reserved | | | | | OVRIE | RES | | AWDEN | JAWDEN | Reserved | | | | | |
| | | | | | rw | rw | rw | rw | rw | | | | | | |

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DISCNUM[2:0] | | | JDISCEN | DISCEN | JAUTO | AWDSGL | SCAN | JEOCIE | AWDIE | EOCIE | AWDCH[4:0] | | | | |
| rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw | rw |

- OVRIE: Overrun interrupt enable
- RES[1:0]: Resolution ->
  - 00: 12-bit (15 ADCCLK cycles)
  - 01: 10-bit (13 ADCCLK cycles)
  - 10: 8-bit (11 ADCCLK cycles)
  - 11: 6-bit (9 ADCCLK cycles)
- AWDEN: Analog watchdog enable on regular channels
- JAWDEN: Analog watchdog enable on injected channels
- SCAN: Scan mode
- JEOCIE: Interrupt enable for injected channels
- EOCIE: Interrupt enable for EOC



# Conversion Completion

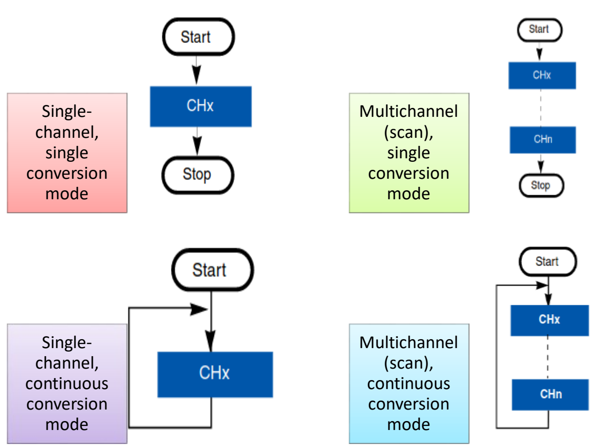
- In single conversion mode
  - Regular channel
    - Store the result into the 16-bit ADC_DR register
    - Set the EOC (end of conversion) flag
    - Interrupt if EOCIE bit is set
  - Injected channel
    - Store the result into the 16-bit ADC_JDR1 register
    - Set the JEOC (end of conversion injected) flag
    - Interrupt if JEOCIE bit is set
- Behave differently in other modes. If there is a sequence of conversions, can be specified to set the flag at the end of the sequence or at the end of every conversion

# Result Registers

- After the conversion, may need extra processing
  - Offset subtraction from calibration
  - Averaging: 1, 4, 8, 16 or 32 samples
  - Formatting: Right justification, sign- or zero-extension to 16 bits
  - Output comparison
- Result registers for two groups
  - ADC_DR for regular group
  - ADC_JDRx (x=1..4) for injected group



# Summary of ADC Modes (AN3116)



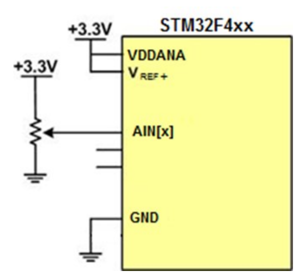
# Using ADC Values

- 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.



# Example: A/D conversion of channel 1

- Channel 1 is connected to input from PA1
- Clock prescaler is left at 0 (divided by 2)
- Sampling time is also left at default of 3 cycles
- Software trigger is used
- The bit 8 of the conversion result is used to turn on/off an LED



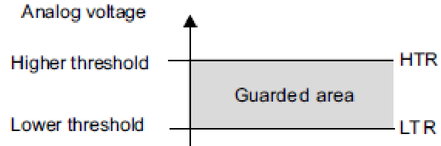
```c
#include "stm32f4xx.h"

int main (void) {
    int result;

    /* set up pin PA5 for LED */
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER &= ~0x00000C00;    /* clear pin mode */
    GPIOA->MODER |=  0x00000400;    /* set pin to output mode */

    /* set up pin PA1 for analog input */
    RCC->AHB1ENR |=  1;             /* enable GPIOA clock */
    GPIOA->MODER |=  0xC;           /* PA1 analog */

    /* setup ADC1 */
    RCC->APB2ENR |= 0x00000100;     /* enable ADC1 clock */
    ADC1->CR2  = 0;                 /* SW trigger */
    ADC1->SQR3 = 1;                 /* conversion sequence starts at ch 1 */
    ADC1->SQR1 = 0;                 /* conversion sequence length 1 */
    ADC1->CR2 |= 1;                 /* enable ADC1 */

    while (1) {
        ADC1->CR2 |= 0x40000000;    /* start a conversion */
        while(!(ADC1->SR & 2)) {}   /* wait for conv complete */
        result = ADC1->DR;          /* read conversion result */
        if (result & 0x100)
            GPIOA->BSRR = 0x00000020;  /* turn on LED */
        else
            GPIOA->BSRR = 0x00200000;  /* turn off LED */
    }
}
```



# ANALOG WATCHDOG

# Analog Watchdog

- Watchdog basically tries to detect exception and recover the MCU from specific situations.
- Analog Watchdog is actually an ADC followed by one (or two) comparator(s).
- ADC1 Channel 17
- Set the status bit (or generate an interrupt) if voltage converted is below a lower threshold or is above a higher.



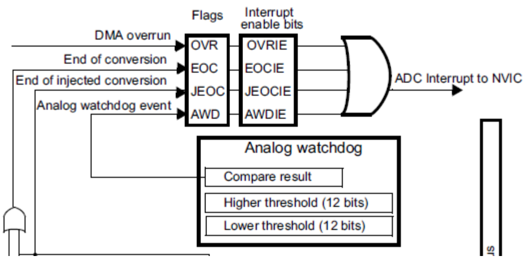

- Can select to watch all channels (either injected or regular groups or even both) or single channels.
- Monitor analog input and bark e.g., if temperature goes crazy!



# Example: Power Failure Detection

- Need warning of when power has failed
- Use continuous mode and the analog watchdog interrupt
- Do the last second jobs!
  - Very limited amount of time before capacitor discharges
  - Save critical information
  - Turn off output devices
  - Put system into safe mode
- Can use a comparator to compare $V_{REFINT}$ (1.2V) against a fixed reference voltage $V_{ref}$
- Save data, money or even life if lucky enough

# Analog Watchdog




- Put the Higher threshold into the 12 least significant bits into the ADC_HTR
- Put the Lower threshold into the 12 least significant bits into the ADC_LTR
- Enable the interrupt by setting the AWDIE bit in ADC_CR1 register



# Channel Selection

| Channels guarded by the analog watchdog | ADC_CR1 register control bits (x = don't care) | | |
|---|---|---|---|
| | AWDSGL bit | AWDEN bit | JAWDEN bit |
| None | x | 0 | 0 |
| All injected channels | 0 | 0 | 1 |
| All regular channels | 0 | 1 | 0 |
| All regular and injected channels | 0 | 1 | 1 |
| Single(1) injected channel | 1 | 0 | 1 |
| Single(1) regular channel | 1 | 1 | 0 |
| Single (1) regular or injected channel | 1 | 1 | 1 |

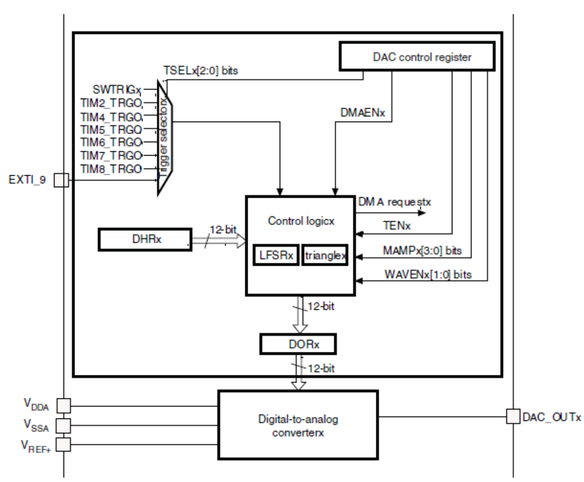
1. Selected by the AWDCH[4:0] bits

- If monitor single channel then needs to select the channel
- For monitoring the $V_{REFINT}$ (Channel 17), write 10001 into AWDCH
- Need to enable the channel by writing setting the TSVREFE bit in CCR (which enables both temperature sensor and $V_{REFINT}$)

# DIGITAL TO ANALOG CONVERTER



# STM32F4xx DAC (NOT available in STM32F401)



# DAC using PWM