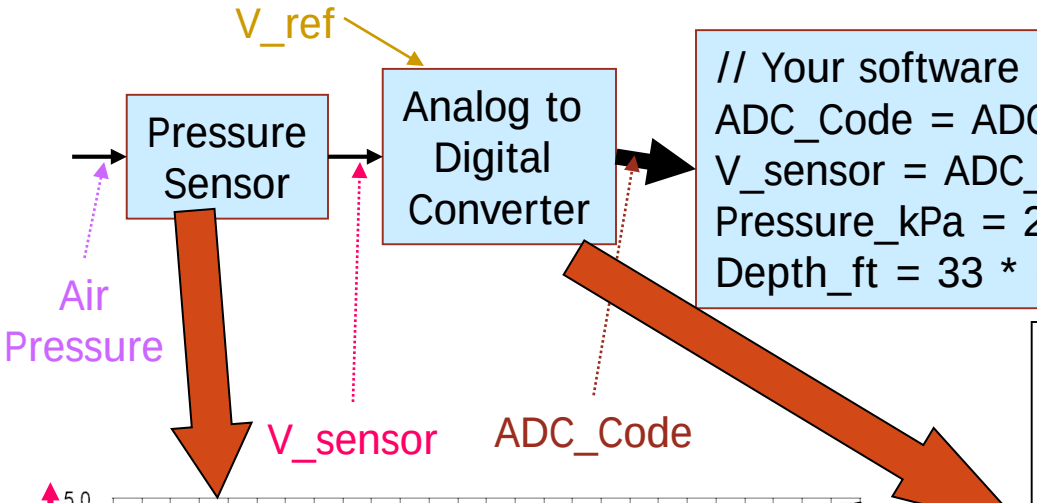
Analog input/output

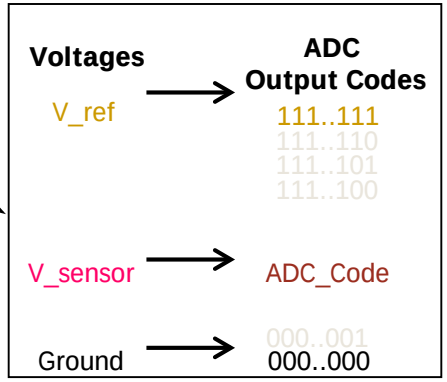
Textbook: Chapter 20, Analog-to-Digital Converter
Chapter 21, Digital-to-Analog Converter

STM32F4xx Technical Reference Manual:
Chapter 11 – Analog to Digital Converters
Chapter 12 – Digital to Analog Converters

The Big Picture – A Depth Gauge




```
// 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;
```



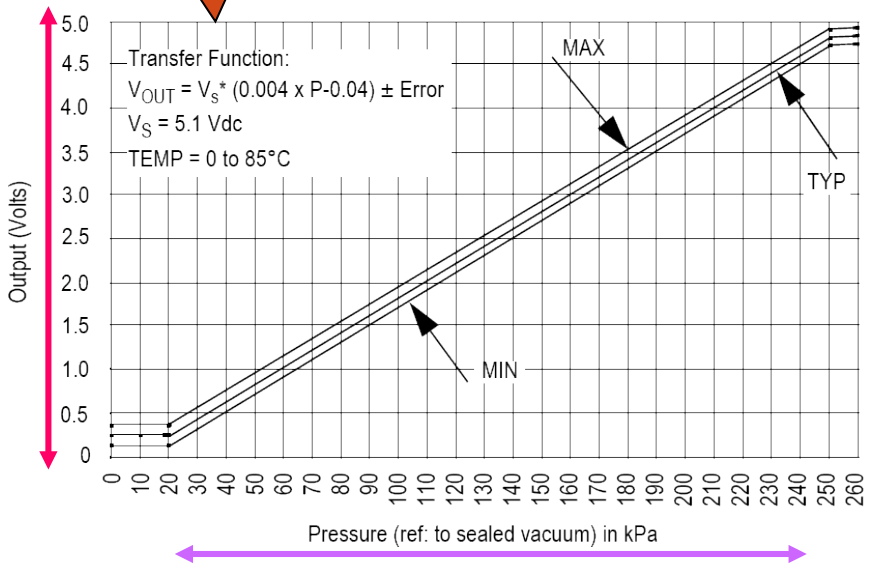


Figure 4. Output vs. Absolute Pressure

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*

Analog to digital conversion

- Given: continuous-time electrical signal

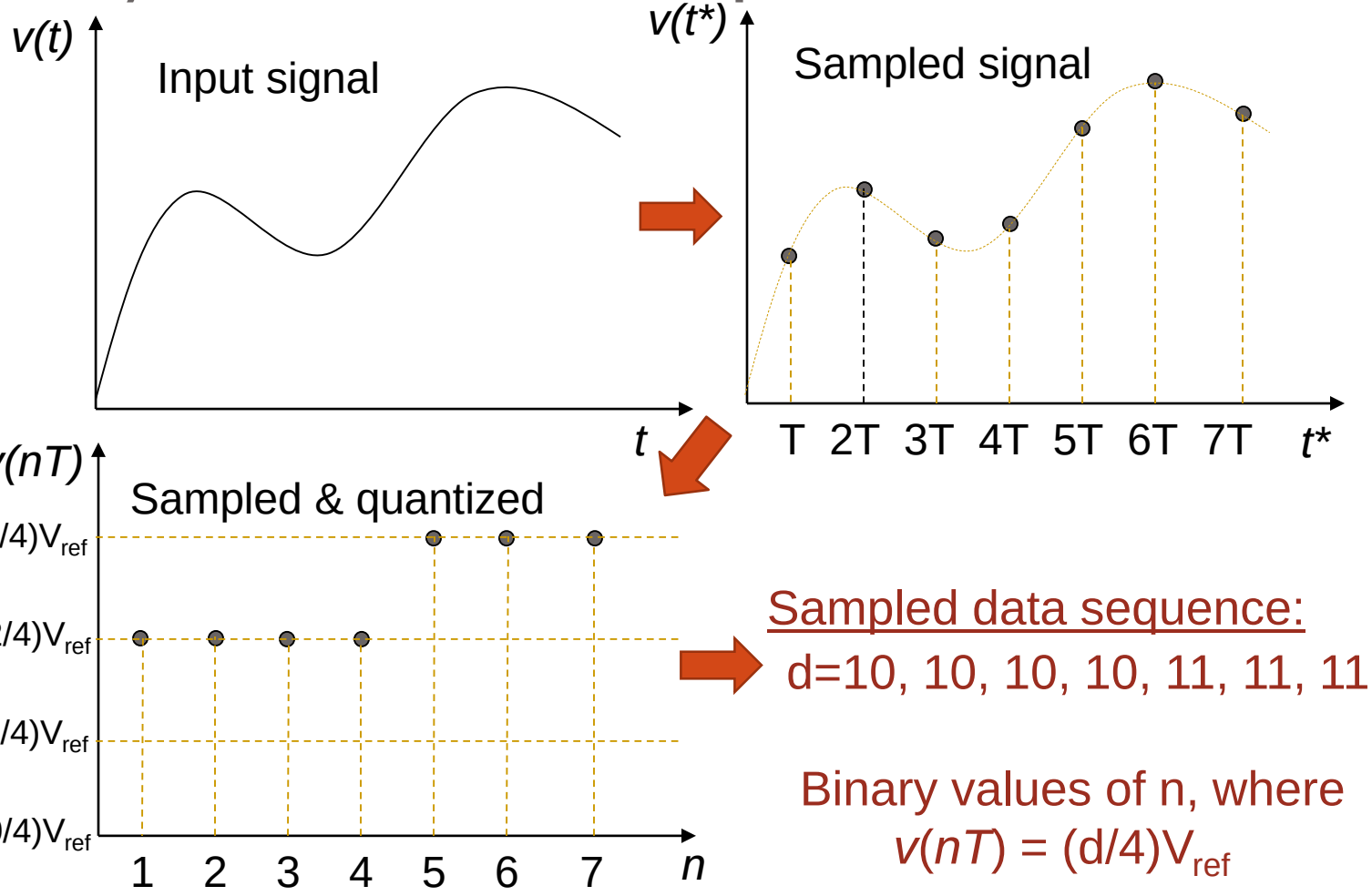
  $v(t), t >= 0$

- Desired: sequence of discrete numeric values that represent the signal at selected sampling times :

  $v(0), v(T), v(2T), \ldots v(nT)$

  - T = "sampling time": v(t) "sampled" every T seconds
  - n = sample number
  - v(nT) = value of v(t) measured at the n^{th} sample time and quantized to one of 2^k discrete levels: k = #bits used to represent v(nT)

A/D conversion process



Sampled data sequence:

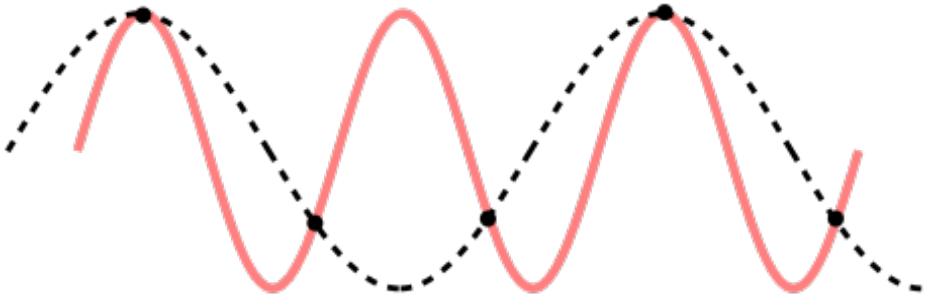
d=10, 10, 10, 10, 11, 11, 11

Binary values of n, where

$v(nT) = (d/4)V_{ref}$

Minimum Sampling Rate: Nyquist–Shannon Sampling Theorem

- In order to be able to reconstruct the analog input signal, the sampling rate should be at least twice the maximum frequency component contained in the input signal
- Example of two sine waves have the same sampling values. This is called aliasing.



from wiki.com

- Antialiasing (beyond the scope of this course)
  - Pre-filtering: use analog hardware to filtering out high-frequency components and only sampling the low-frequency components. The high-frequency components are ignored.
  - Post-filtering: Oversample continuous signal, then use software to filter out high-frequency components

Digital to Analog Converter

Digital-to-analog converter (DAC)

- Converts digital data into a voltage signal by a N-bit DAC

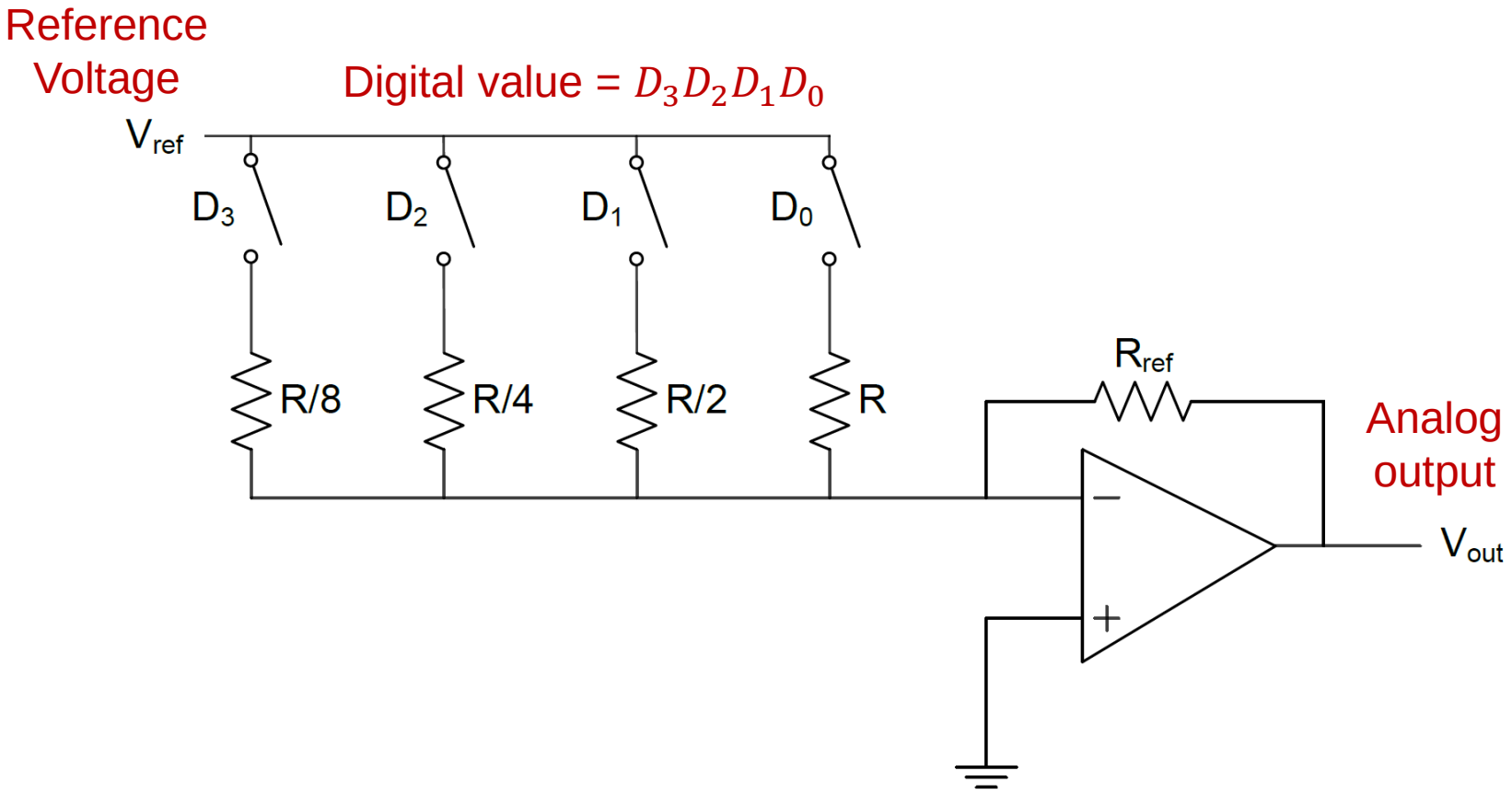
$$DAC_{output} = V_{ref} \times \frac{Digital\ Value}{2^N - 1}$$

- For 12-bit DAC

$$DAC_{output} = V_{ref} \times \frac{Digital\ Value}{4095}$$

- Many applications:
  - digital audio
  - waveform generation
- Performance parameters
  - speed
  - resolution
  - power dissipation

Binary-weighted Resistor DAC

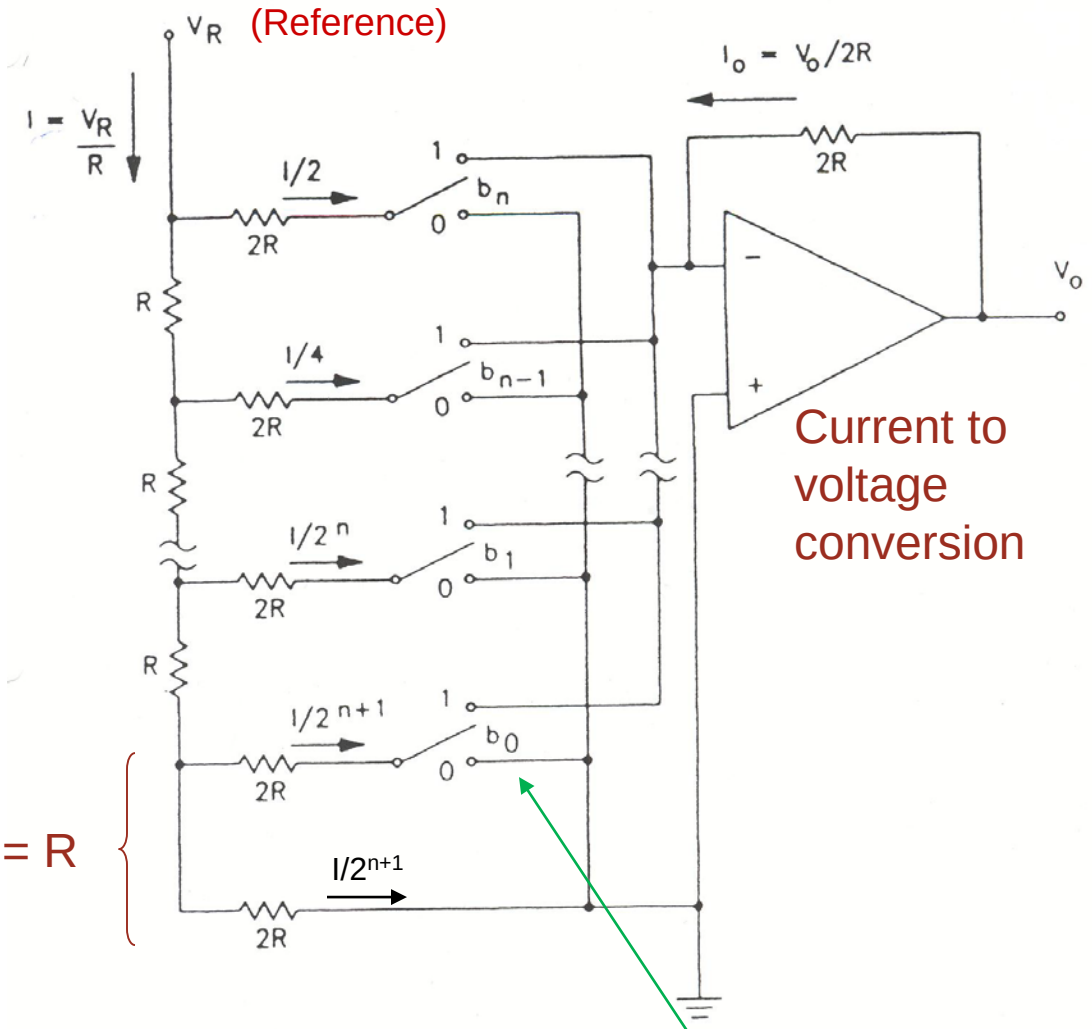


$$V_{out} = V_{ref} \times \frac{R_{ref}}{R} \times (D_3 \times 2^3 + D_2 \times 2^2 + D_1 \times 2 + D_0)$$

Y. Zhu, Chapter 21

R-2R Resistor Ladder DAC

$$V_o = V_R \sum_{k=0}^{n} b_k \left(\frac{1}{2^k}\right)$$



Equivalent resistance = R

Equivalent resistance = R

Number = $b_n b_{n-1} \ldots b_1 b_0 = b_n * 2^n + b_{n-1} * 2^{n-1} + \ldots + b_1 * 2^1 + b_0 * 2^0$

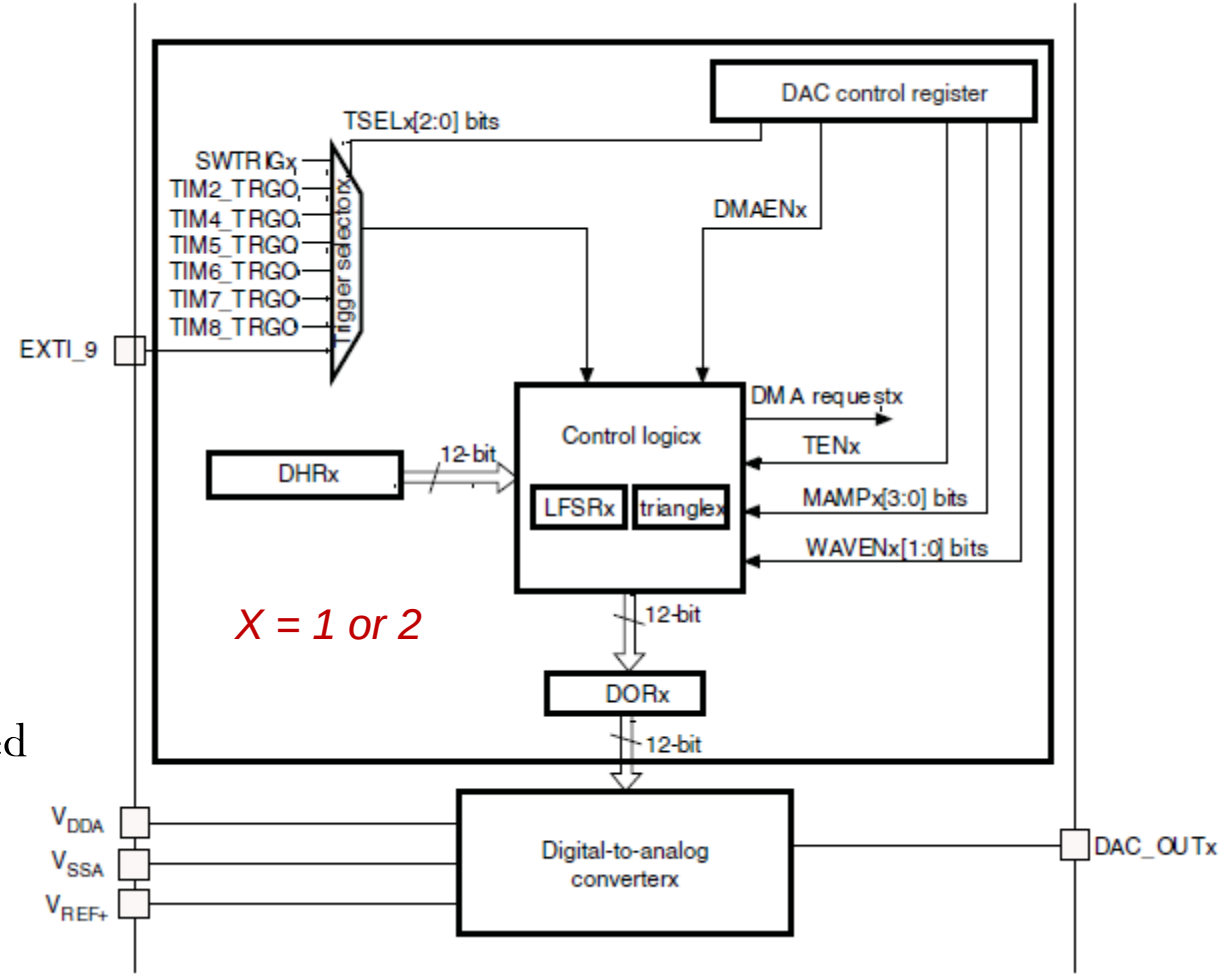
STM32F4xx DAC Overview

- Two independent channels

  DAC1 connects to **PA4**

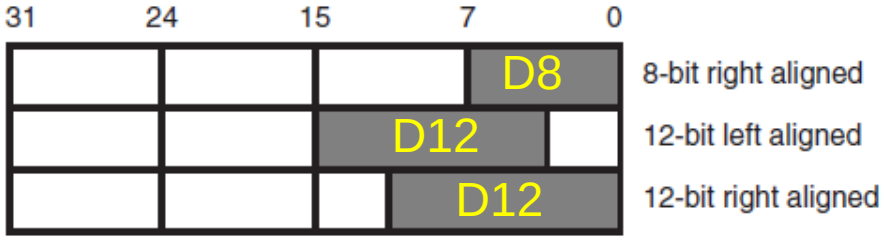
  DAC2 connects to **PA5**

- $DAC_{output(12bit)} = V_{REF} \times DOR \div 4095$
  - Range from 0 to $V_{REF}+$
- Several data formats
- Dual-channel mode
- Supports DMA
- Output buffers can be enabled for better ability to drive external loads



DAC data holding registers

Single-Channel Data Formats



Data Holding Register:

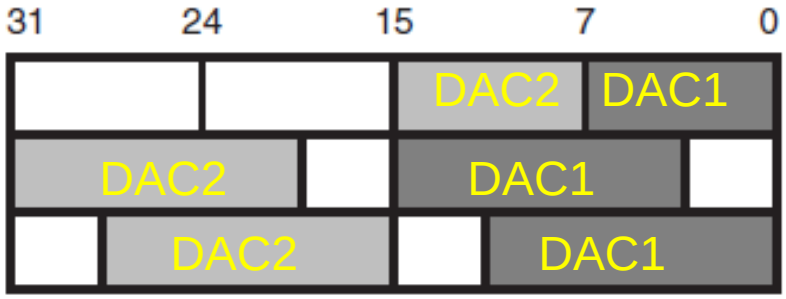
DAC_DHR8Rx

DAC_DHR12Lx

DAC_DHR12Rx

(x = 1 (DAC1) or 2 (DAC2)

Dual-DAC mode: trigger simultaneous conversions in both DACs by writing data to Dual-Data Holding Registers: DAC_DHRxyD[31:0]



Data Holding Register:

DAC_DHR8RD

DAC_DHR12LD

DAC_DHR12RD

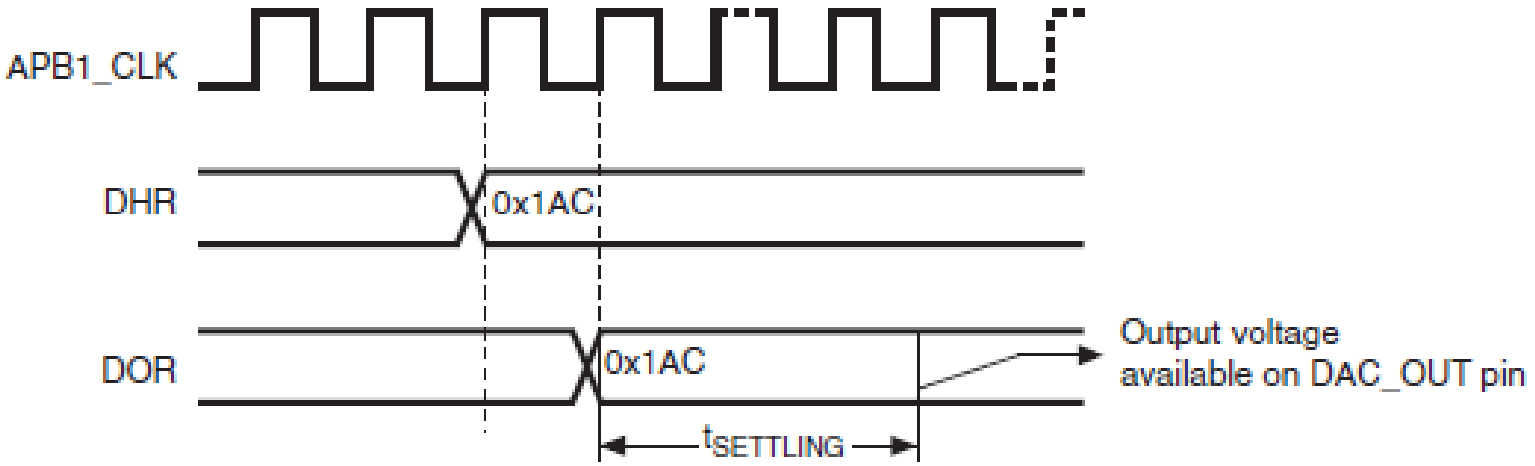
DAC conversion

- Data sample is written to one of the data holding registers DAC_DHRx *(DHR corresponding to data format)*
- No hardware trigger selected (TENx=0 in DAC_CR register):
  - DAC_DHRx updates DAC_DORx after one APB1 clock cycle
- Hardware trigger selected: (TENx=1 in DAC_CR register)
  - Update delayed to three APB1 clock cycles after trigger event.
- Depending on power supply voltage and output load, the analog output voltage will be available after a time $t_{SETTLING}$ (typically 3us) following DAC_DORx update.

  - $DAC_{output} = V_{REF} \times DOR \div 4095$ *(12-Bit Mode)*
    - Output voltage range from 0 to V_{REF+}

TEN(Trigger Enable)

- The timing diagram when trigger is **disabled** (TEN=0)
  - Conversion begins one APB1_CLK cycle after data written to DHR



- When the trigger is enabled (TEN=1), an external event is selected to trigger the DAC
  - Select hardware triggers from 6 on-chip timers or EXTI line 9
  - Can also be triggered by software (set bit in SWTRIG register)

DAC Registers (partial list)

- DAC Register Boundary = 0x4000 7400 (on APB1)
  - **DAC_CR** (control register) – address offset 0x00
  - **DAC_SR** (status register) – 0x34 (only has DMA underrun flags)
- Channel 1 Data Holding Registers
  - **DAC_DHR12R1** (Channel 1, 12-bit, right-aligned data) – 0x08
  - **DAC_DHR12L1** (Channel 1, 12-bit left-aligned data) – 0x0C
  - **DAC_DHR8R1** (Channel 1, 8-bit right-aligned data) – 0x10
- Channel 2 Data Holding Registers
  - **DAC_DHR12R2** (Channel 2, 12-bit, right-aligned data) – 0x14
  - **DAC_DHR12L2** (Channel 2, 12-bit left-aligned data) – 0x18
  - **DAC_DHR8R2** (Channel 2 8-bit right-aligned data) – 0x1C
- Dual-Channel Data Holding Registers (DAC_DHRxyD)

DAC Control Register : DAC_CR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		DMAUDRIE2	DMAEN2	MAMP2[3:0]				WAVE2[1:0]		TSEL2[2:0]			TEN2	BOFF2	EN2
		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		DMAUDRIE1	DMAEN1	MAMP1[3:0]				WAVE1[1:0]		TSEL1[2:0]			TEN1	BOFF1	EN1
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

- DAC_CR[15:0] for channel 1; DAC_CR[31:16] for channel 2
  - **EN** bit: enable the DAC channel
  - **TEN** bit: trigger enable bit (0 to disable, 1 to enable)
    (also determines #clock cycles before DHR load into DOR)
  - **TSEL** bits: trigger selection (if TEN=1)
  - BOFF bit: output buffer disable, to enhance the driving ability
    but the output may not reach 0 if enabled, set 1 to disable
  - MAMP/WAVE bits: To generate noise wave or triangle wave (Only use when TEN is 1)
  - DMA EN/DMAU DRIE bits: Enable/configure DMA xfers from memory to DHR

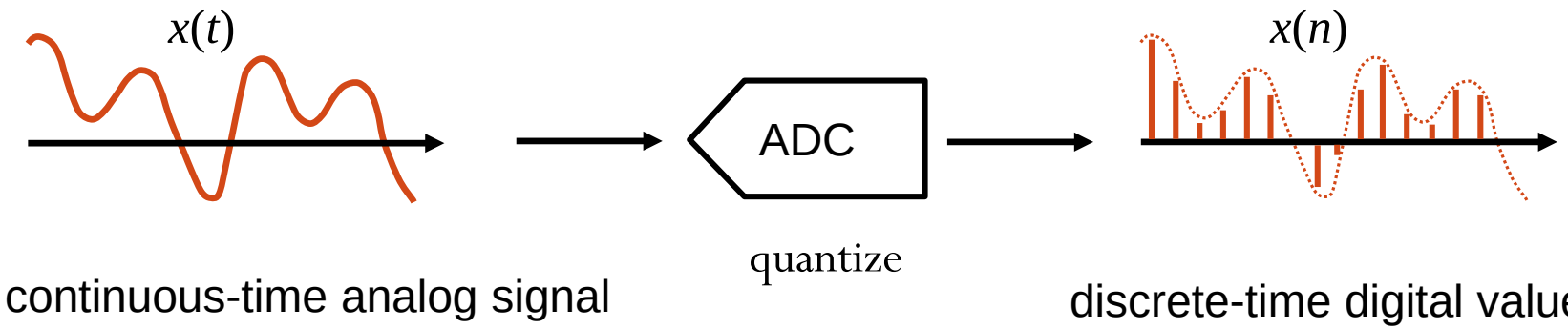
Example: Waveform Generator

- Configure corresponding GPIO pin in analog mode
- Enable the DAC Clock (APB1)
- Enable the DAC
- Configure the DAC
  - Trigger Enable
    - TEN=0: No trigger/normal mode (not buffered)
    - TEN=1: Trigger on selected source
- Periodically write data to DAC DHR data register (8 or 12 bits)

Analog to Digital Converter

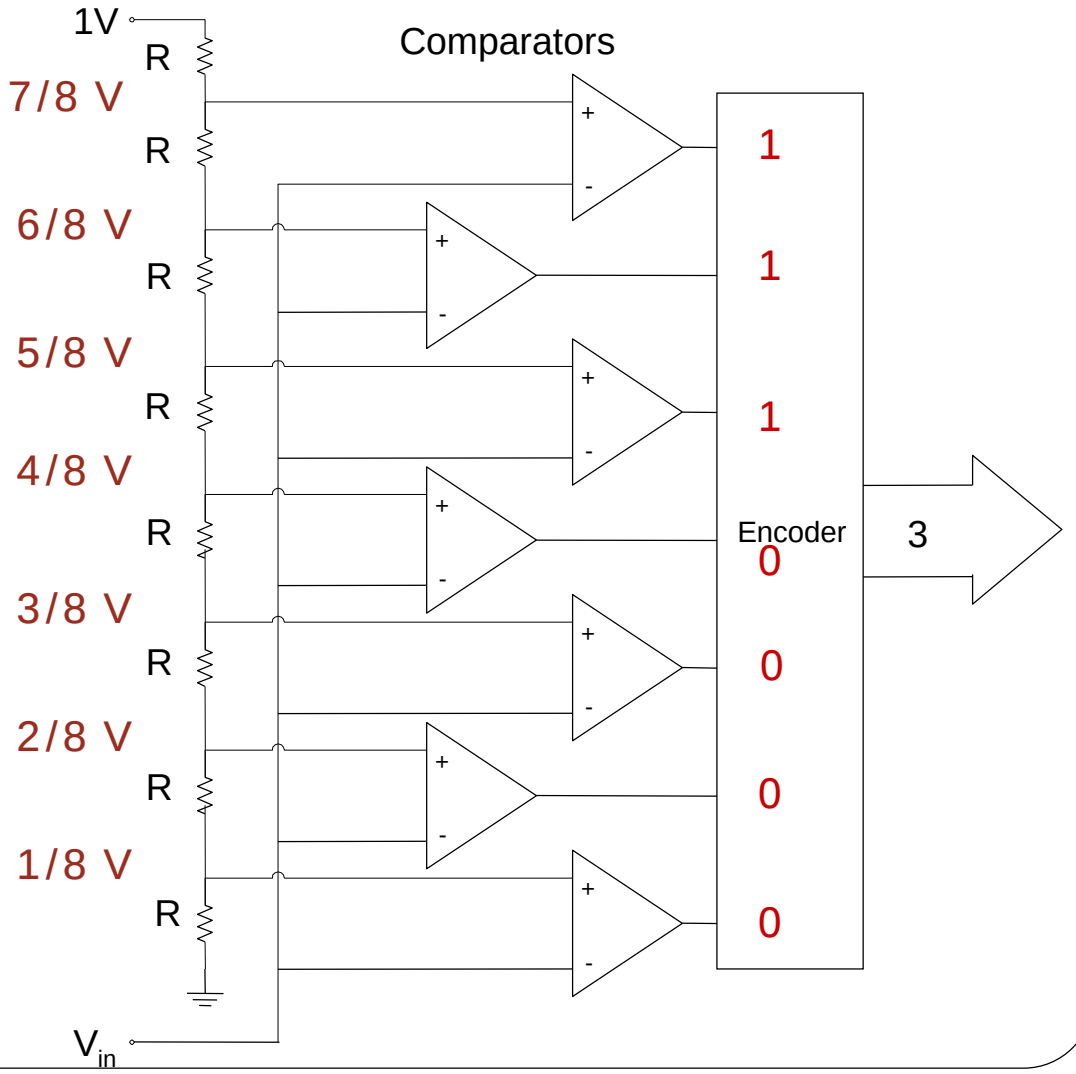
Analog-to-Digital Converter (ADC)

- ADC is important almost to all application fields
- Converts a continuous-time voltage signal within a given range to discrete-time digital values to quantify the voltage's amplitudes

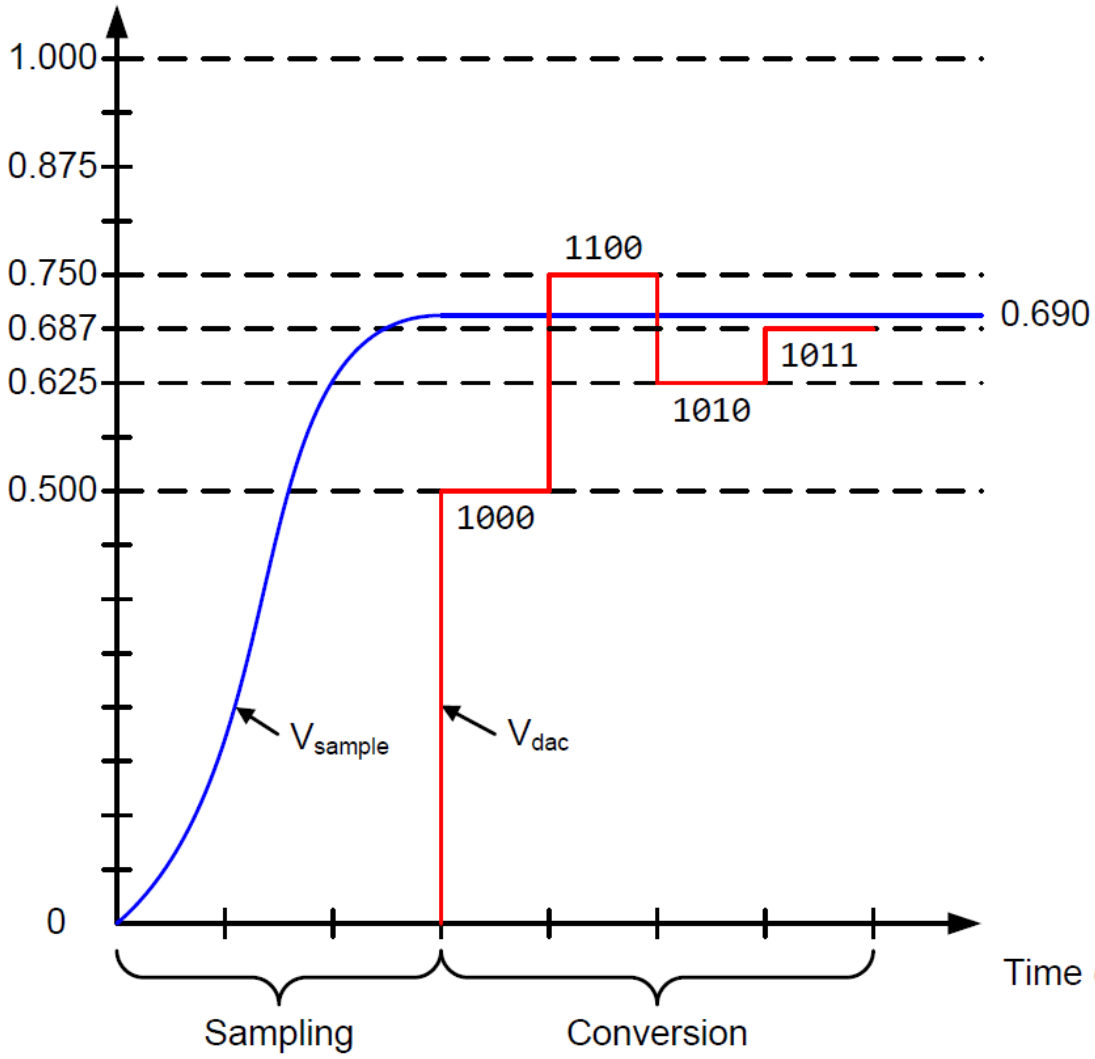


A/D – Flash Conversion

- 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)



Successive-approximation (SAR) ADC



- Binary search algorithm to gradually approaches the input voltage
- Settle into ±½ LSB bound within the time allowed

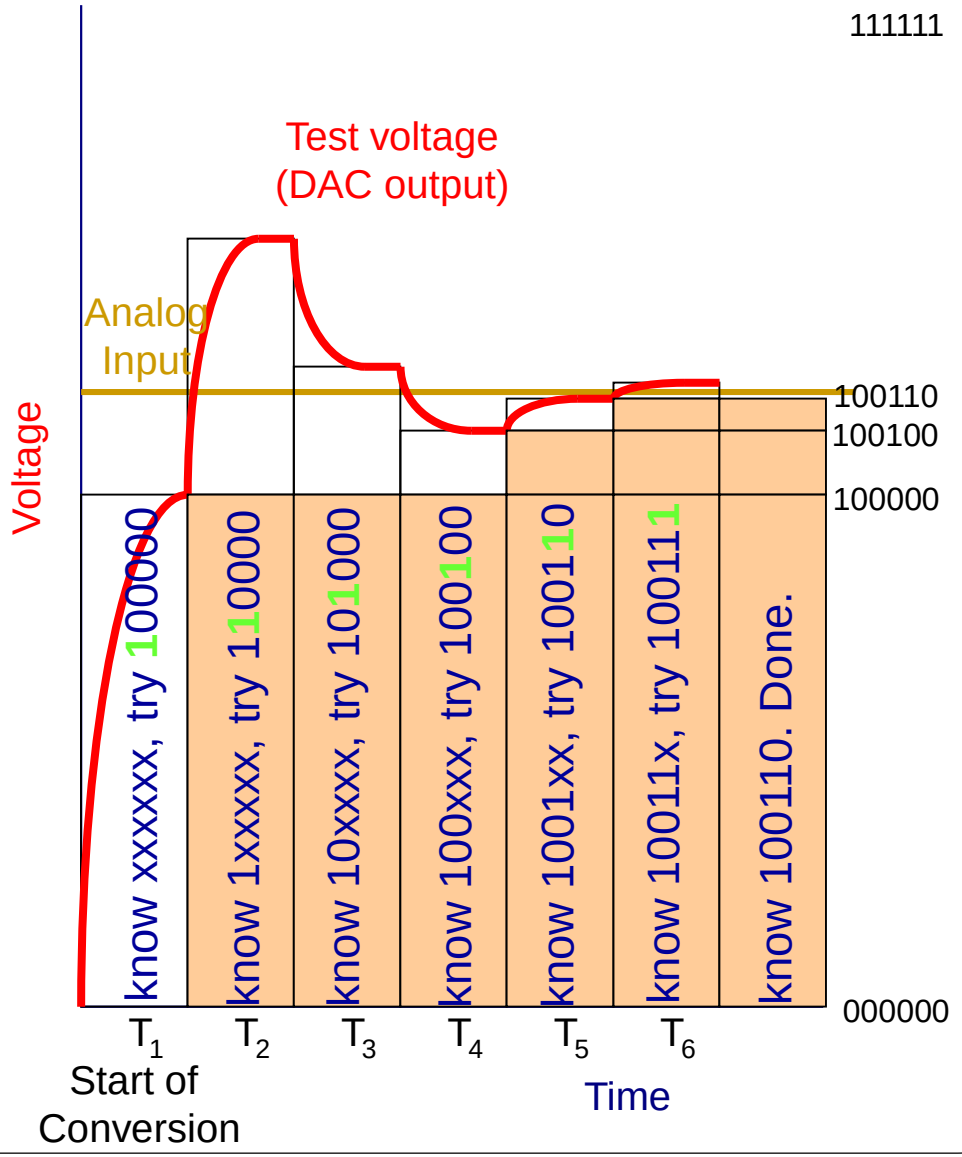
$$T_{ADC} = T_{sampling} + T_{Conversion}$$

$$T_{Conversion} = \mathrm{N} \times T_{ADC\_Clock}$$

$T_{sampling}$ is software configurable

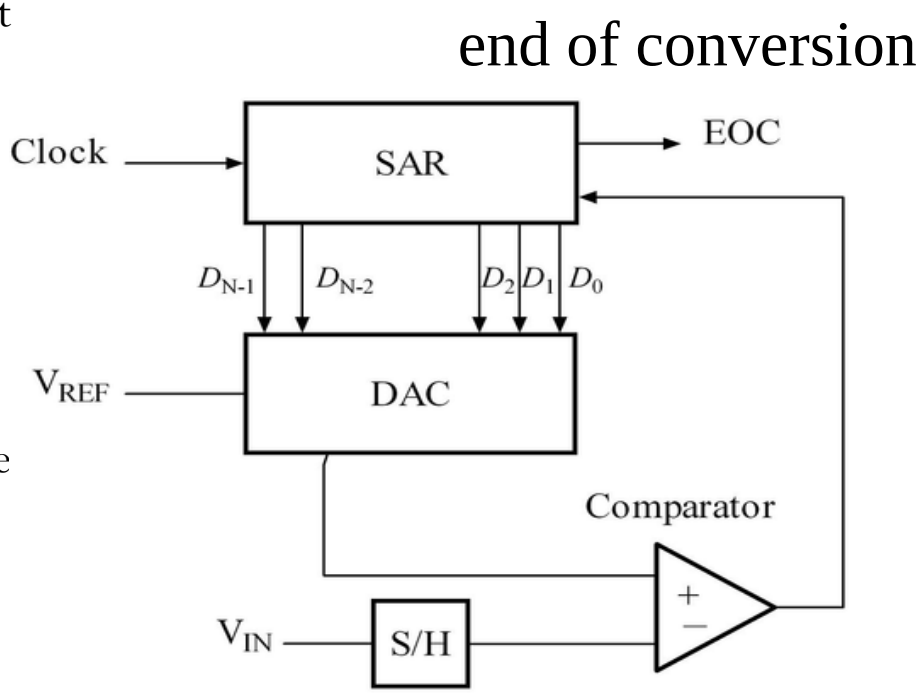
ADC - Successive Approximation Conversion

- 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



Analog to Digital Converter (ADC)

- Successive approximation ADC
  - V_{IN} is approximated as a static value in a *sample and hold* (S/H) circuit
  - the *successive approximation register* (SAR) is a counter that increments each *clock* as long as it is enabled by the *comparator*
  - the output of the SAR is fed to a DAC that generates a voltage for comparison with V_{IN}
  - when the output of the DAC = V_{IN} the value of SAR is the digital representation of V_{IN}



Bard, Gerstlauer, Valvano, Yerraballi

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

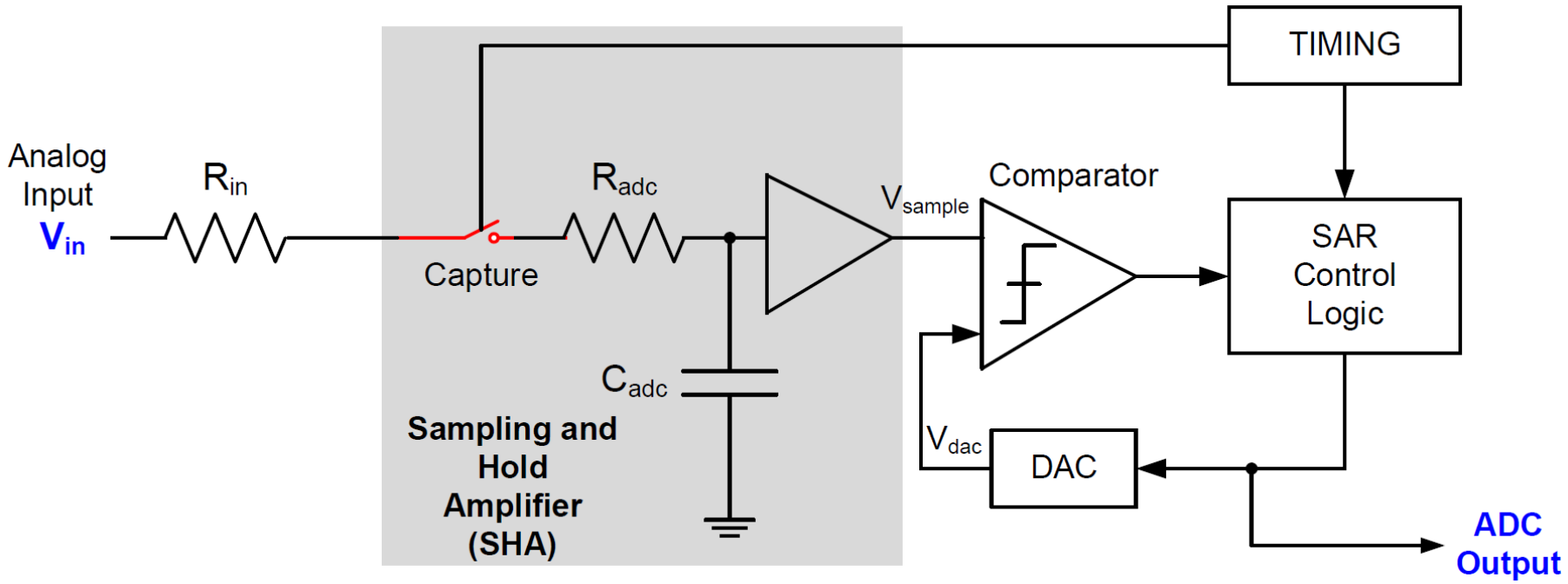
- Nyquist and the real world
  - This theorem assumes we have a perfect filter with "brick wall" roll-off
  - Real world filters have more gentle roll-off
  - Inexpensive filters are even worse (e.g. first order filter is 20 dB/decade, aka 6 dB/octave)
  - So we have to choose a sampling frequency high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

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

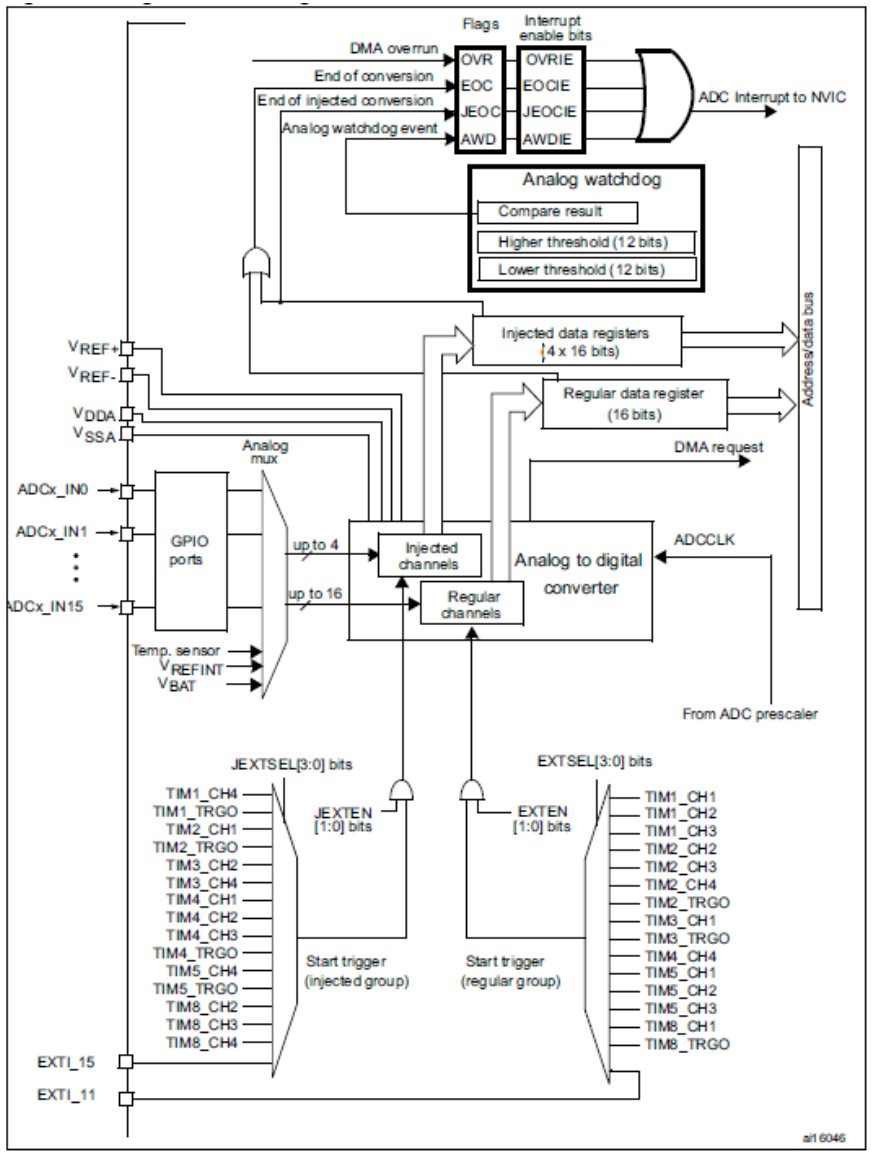
Sample and Hold Devices

- May require analog input signal to be held constant during conversion.
- Peak capture or sampling at a specific point in time may necessitate a sampling device.
- "Sample and Hold": Analog Input (Vin) is sampled when the Capture switch is closed and its value is *held* on capacitor Cadc, where it becomes the analog output Vsample
  - S&H devices are incorporated into some A/D converters

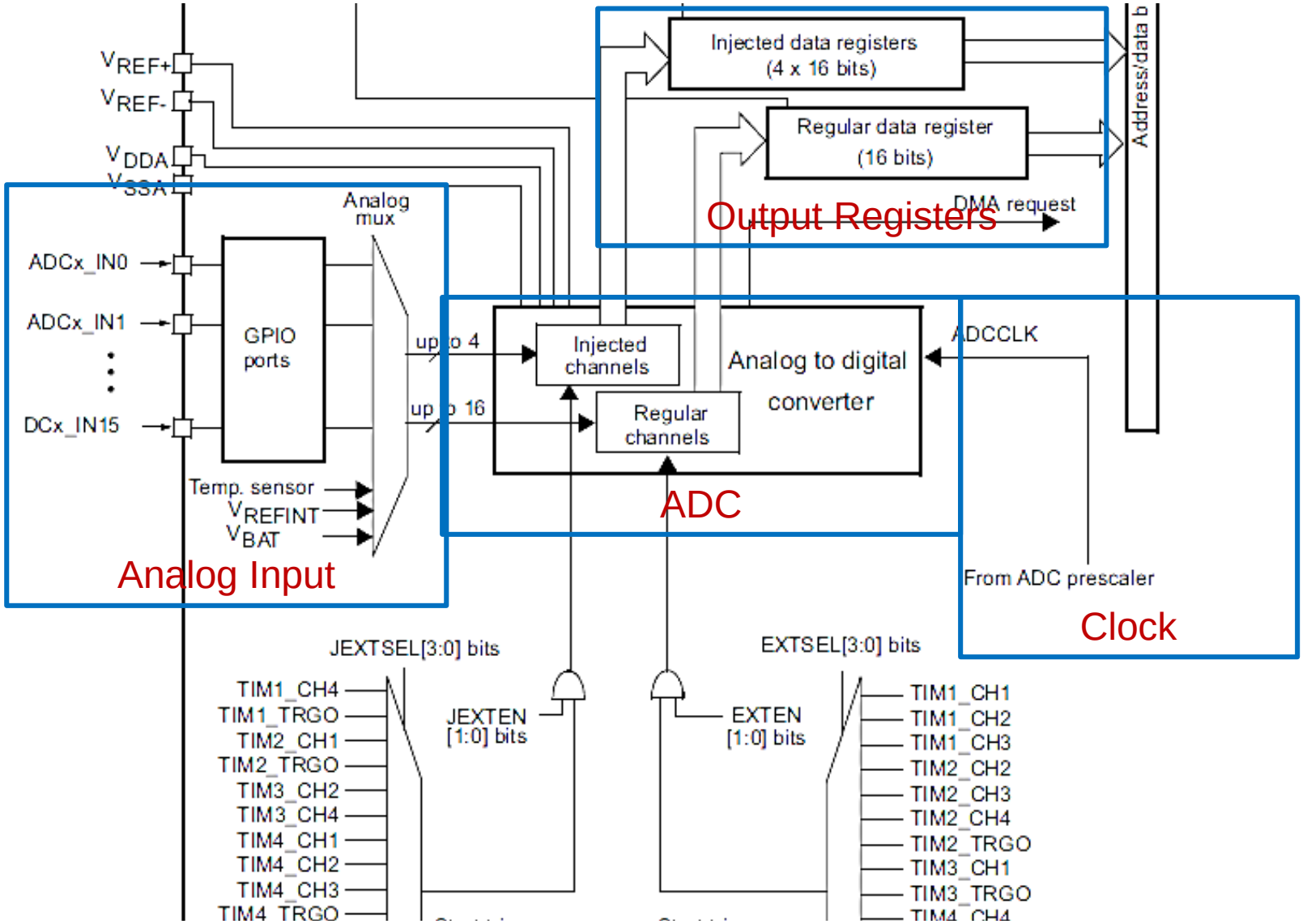


STM32F4xx ADC Overview

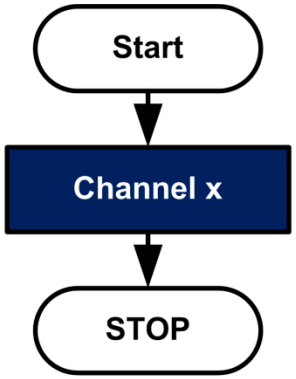
- Successive approximation conversion
- High sampling speeds
- Conversion range 0 to 3.6 volts
- Supports multiple resolutions:
  12, 10, 8 and 6 bits
- 16 regular and 4 injected channels
- Supports single and continuous conversions
- Scan mode for sequence of inputs
- Interrupt generation on
  - End of conversion
  - Analog watchdog
  - Overrun
- Supports DMA transfers
- Analog watchdog
- Temperature sensor



ADC System Fundamentals

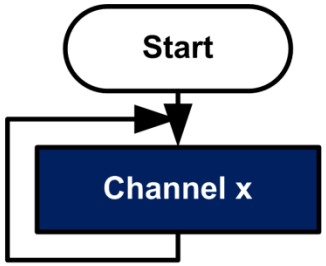


ADC Modes



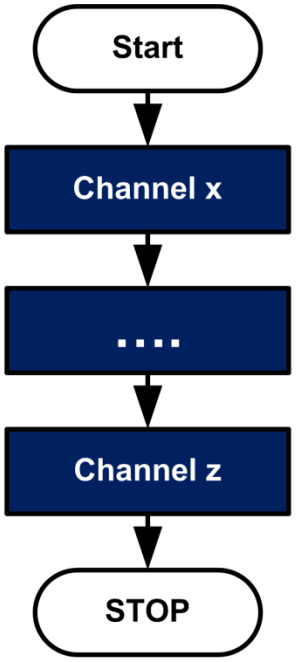
Single Channel, Single Conversion Mode

CONT in ADC_CR2 = 0
SCAN in ADC_CR2 = 0



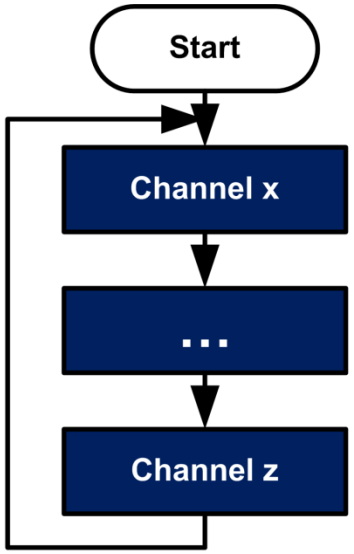
Single Channel, Continuous Conversion Mode

CONT in ADC_CR2 = 1
SCAN in ADC_CR2 = 0



Scan Mode with Single Conversion

SCAN in ADC_CR2 = 1
CONT in ADC_CR2 = 0



Scan Mode with Continuous Conversion

SCAN in ADC_CR2 = 1
CONT in ADC_CR2 = 1

Using the ADC

- ADC initialization
  - Configure GPIO pin (analog mode)
  - Enable ADC clock
  - Enable ADC
  - Select voltage reference
  - Select trigger source
  - Select input channel
  - Select other parameters
- Trigger conversion
- Read results
- Calibrate? Average?

On-off Control

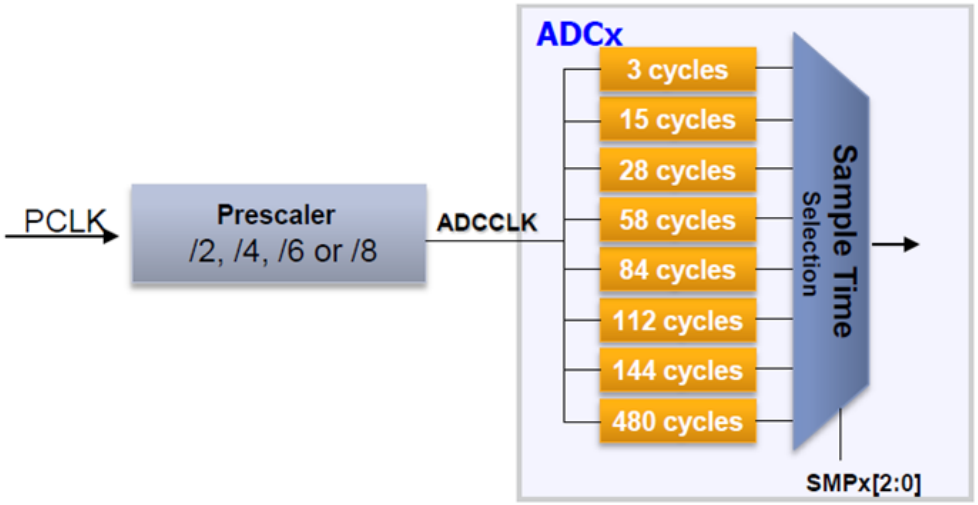
- For power efficiency, the ADC module is usually turned off (even if it is clocked).
- If ADON bit in ADC control register 2 is set, the module is powered on; otherwise it is powered off.
- Good practical to shut down ADC whenever you are not using it.

Clock Configuration

- Analog Clock
  - ADCCLK, common to all ADCs
  - From APB2 (72Mhz) (Can be prescale by 1,2,4,8 or 16)
  - Can be prescaled by 2, 4, 6 or 8, which means at most 36MHz
  - ADC common control register(ADC_CCR) bit 17:16
- Digital Interface Clock
  - Used for registers read/write access
  - From APB2 (72Mhz)
  - Need to be enable individually for each channel (RCC_APB2ENR)

ADC Conversion Time

- Programmable sample time for all channels

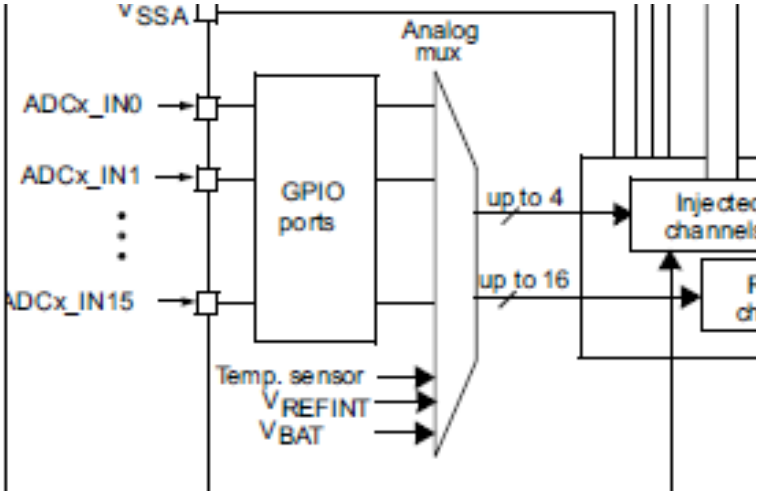


Resolution	$T_{Conversion}$
12 bits	12 Cycles
10 bits	10 Cycles
8 bits	8 Cycles
6 bits	6 Cycles

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

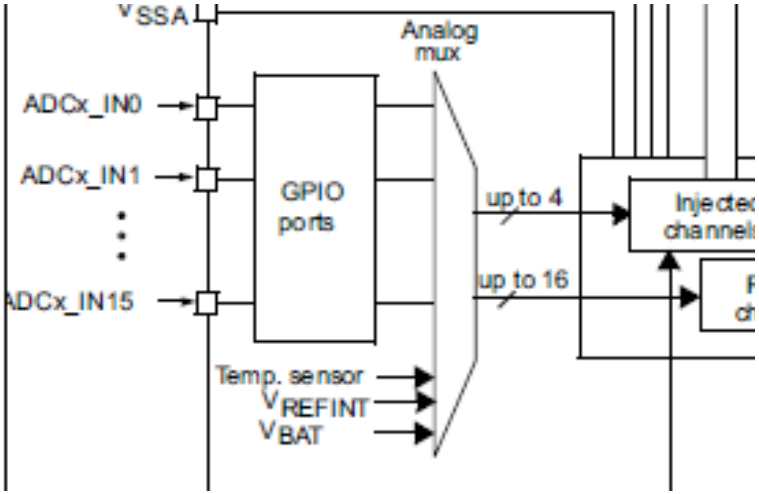
Channel Selection



- 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 least 4 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 least 2 bits in the ADC_JSQR register
  - Modifying either ADC_SQRx or ADC_JSQR will reset the current ADC process.

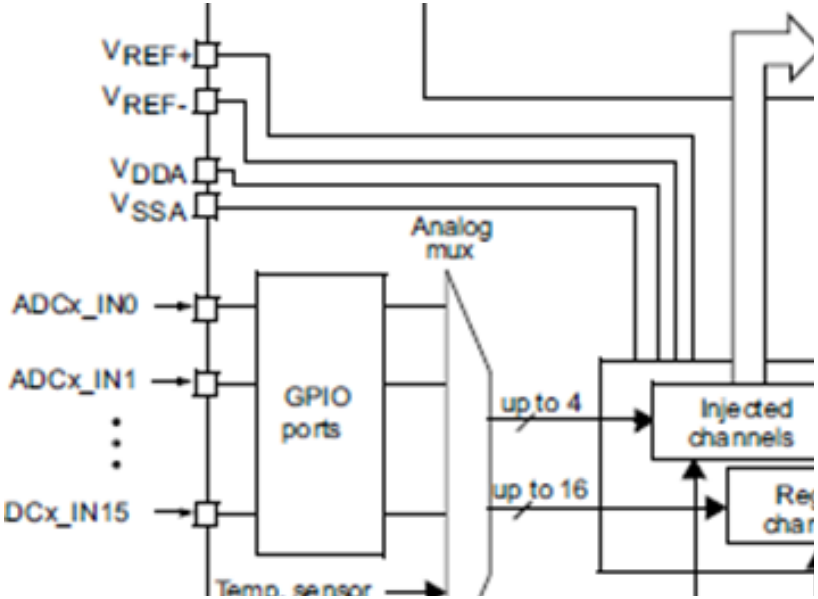
Channel Selection

- Three other channels
  - ADC1_IN16 is internally connected to the temperature sensor
  - ADC1_IN17 is internally connected to the reference voltage VREFINT
  - ADC1_IN18 is connected to the VBAT. Can be use as regular or injected channel.
  - But only available on the master ADC1 peripheral.



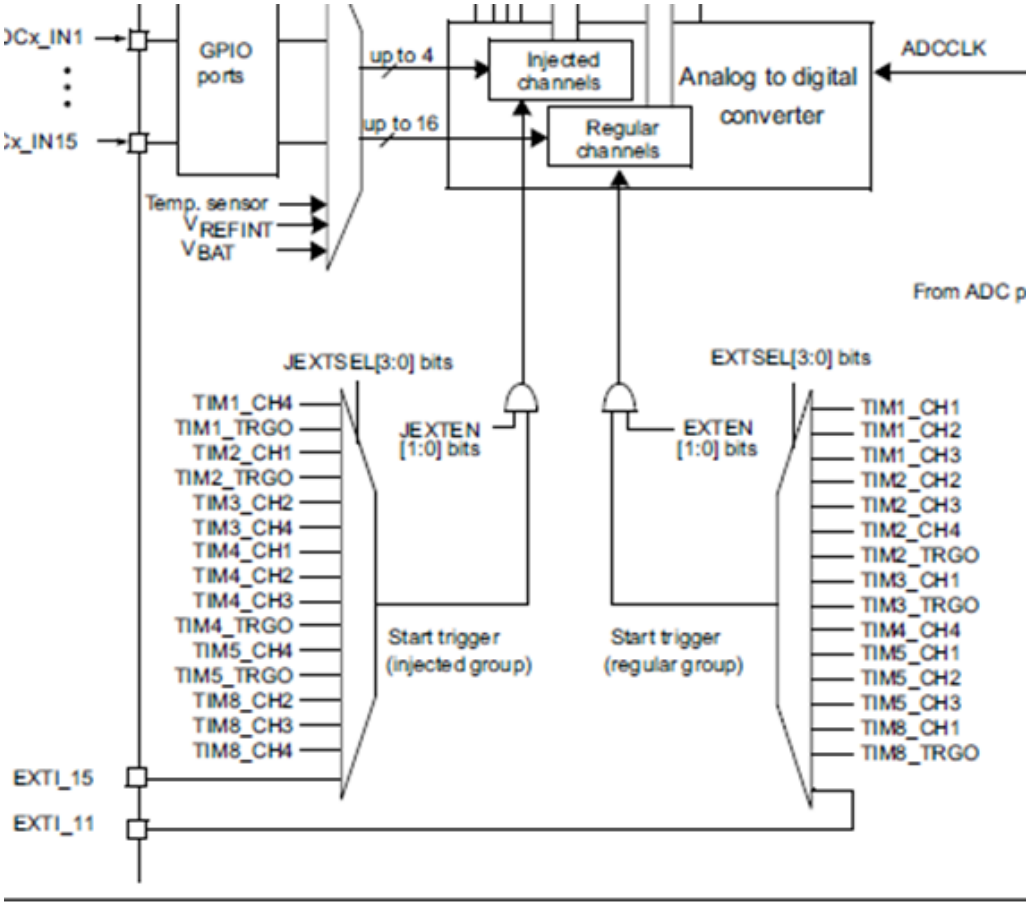
Voltage Reference Selection

- Input range from V_{REF-} to V_{REF+}
- V_{REF+} Positive analog reference
- V_{DDA} equal to Vdd
- 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



Hardware Trigger Sources

- ADC control register 2

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

- 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]

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. And 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

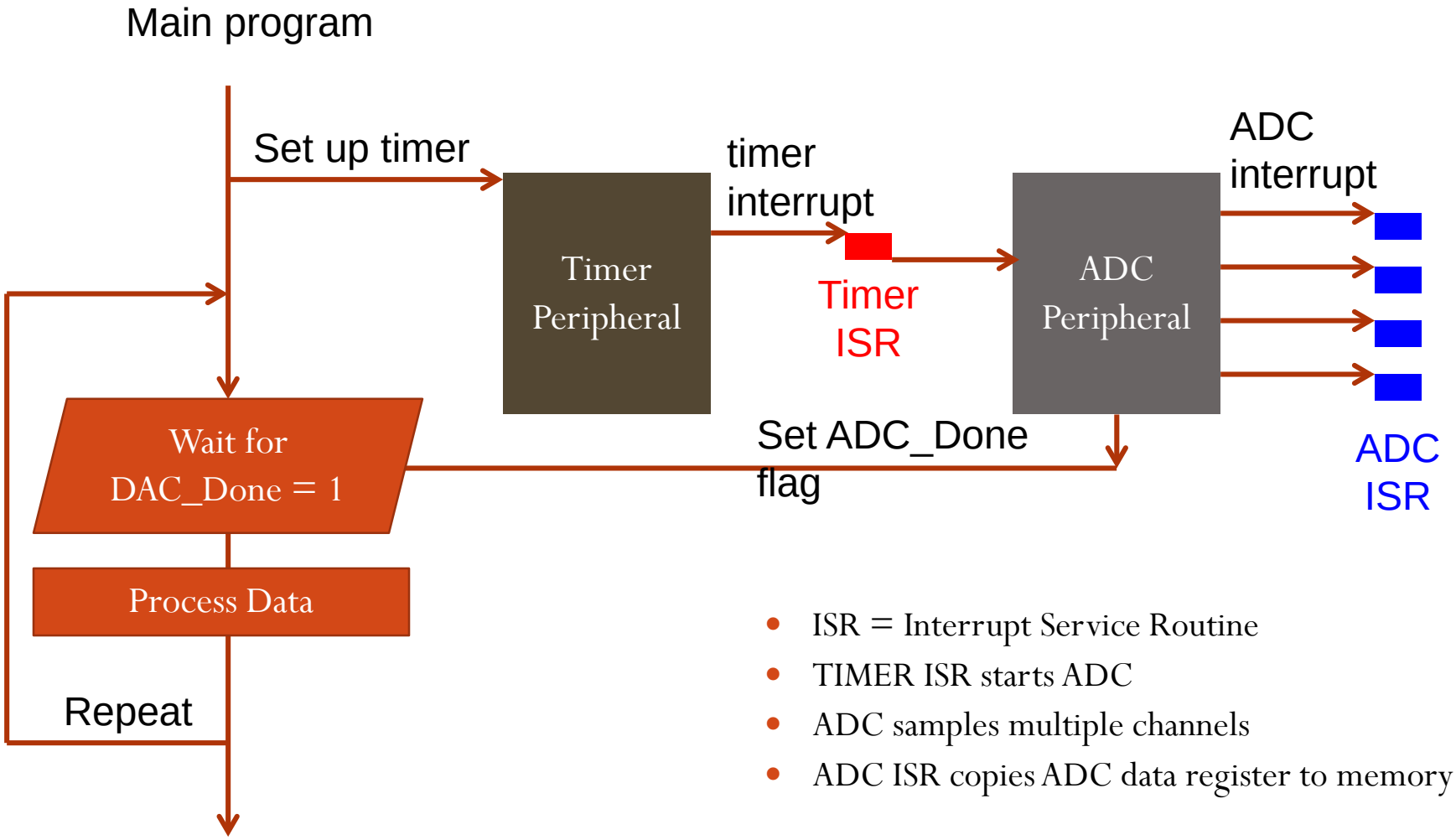
Common Control Register

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
DMA[1:0]		DDS	Res.	DELAY[3:0]				Reserved			MULTI[4:0]				
rw	rw	rw		rw	rw	rw	rw				rw	rw	rw	rw	rw

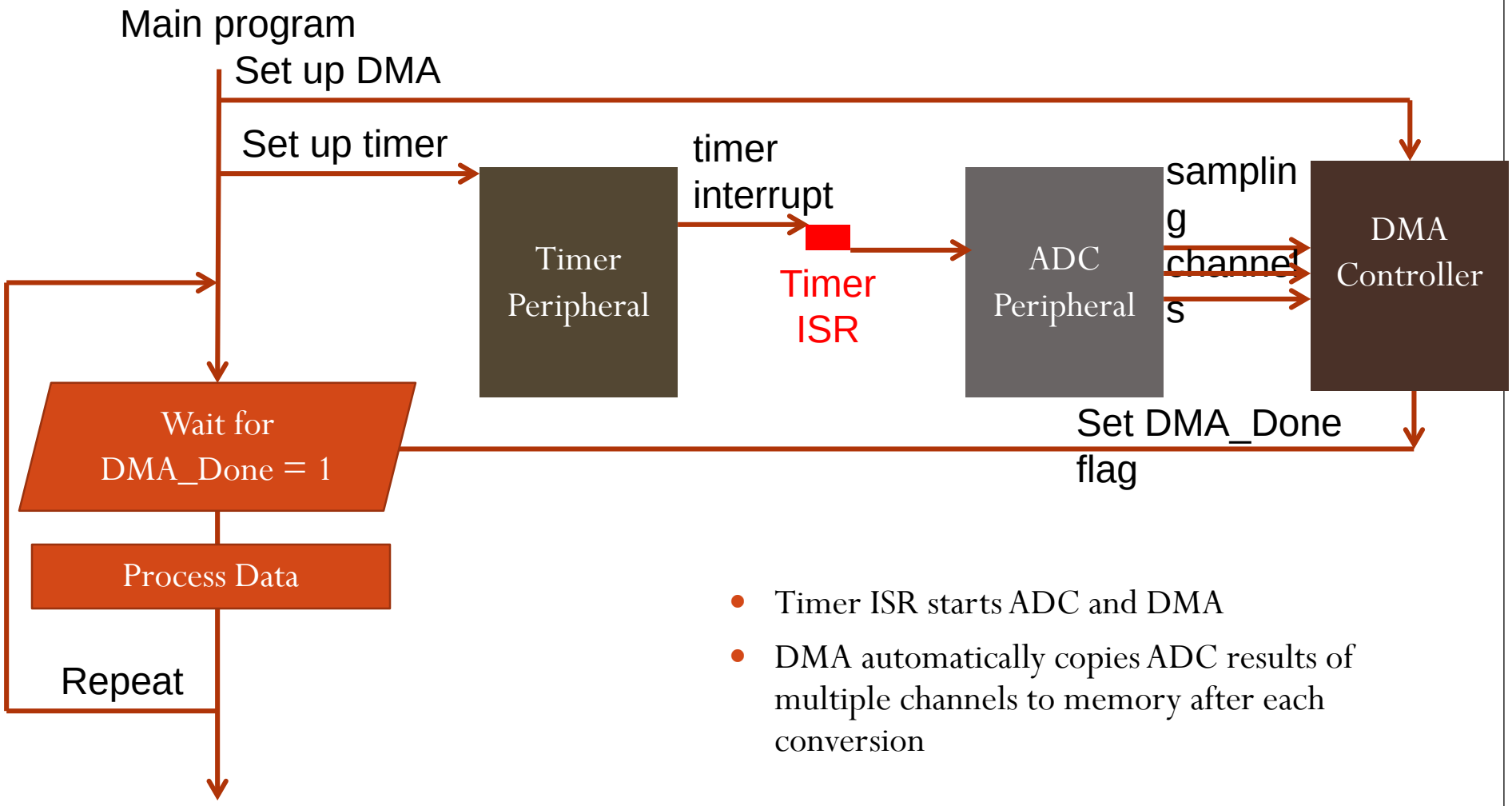
- Select different modes by writing to MULTI [4:0] bits
- Prescale the clock by writing to ADCPRE bits
- Enable the V_{BAT} or the temperature sensor by setting VBATE or TSVREFE
- Decide the delay between to sampling phases by writing to DELAY bits

Example: ADC with Timer Interrupts



- ISR = Interrupt Service Routine
- TIMER ISR starts ADC
- ADC samples multiple channels
- ADC ISR copies ADC data register to memory

Example: ADC with Timer and DMA



- Timer ISR starts ADC and DMA
- DMA automatically copies ADC results of multiple channels to memory after each conversion

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: Temperature Sensor

- ADC1 Channel 16
- The minimum ADC sampling time for the temperature channel is 10 microseconds
- Sampling cycles at least 110
- T(°C) = {(Vsense-V25)/Avg_Slope}+25
  - V25=Vsense value for 25°C(typical value:0.78V)
  - Avg_Slope=average slope of the temperature vs. Vsense curve(typical value:1.3mv/°C)
  - Vsense=DR×3/4096 (If Vref+=3v, Vref-=0v, 12-bit format)
  - Statics really vary from board to board!

- Use your finger to press the chip in the center of the board, the temperature will go high.