Lab 10. Speed Control of a D.C. motor

Speed Measurement:

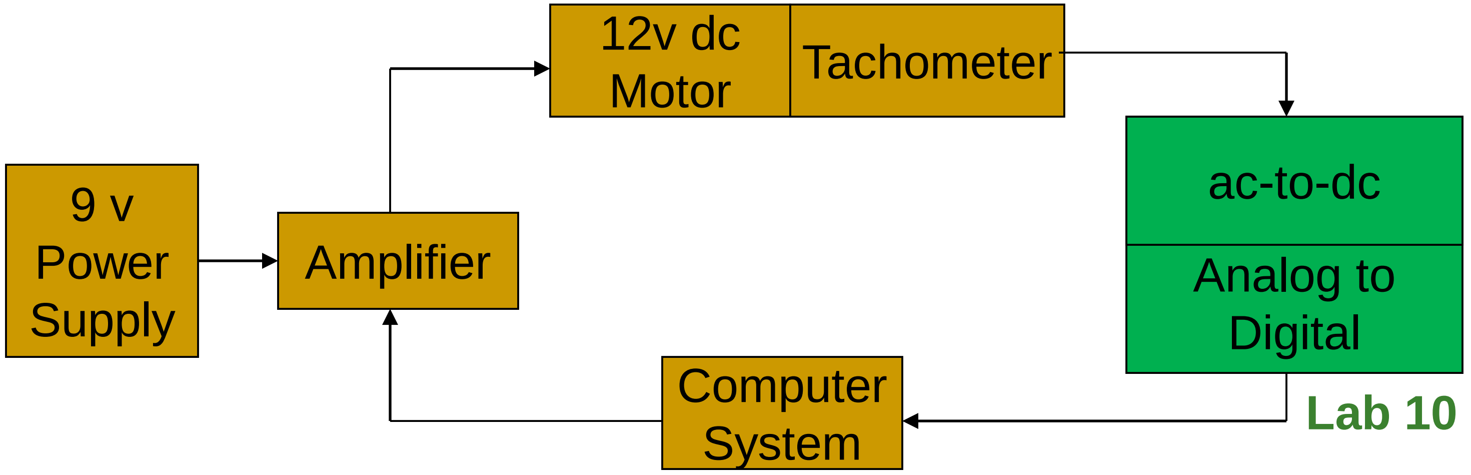
Tach Amplitude Method

References: STM32L100 Data Sheet (pin definitions)

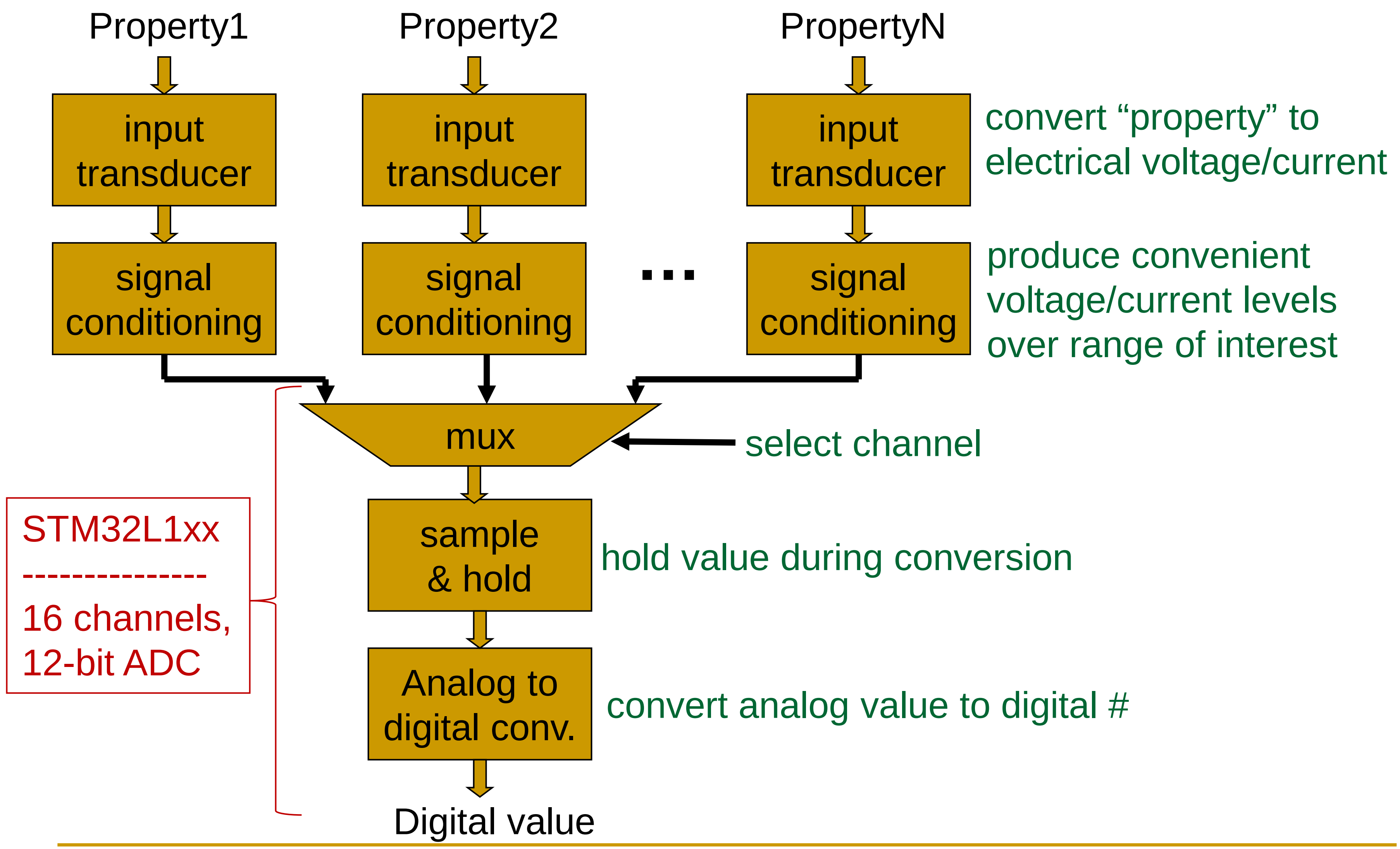
STM32L100 Ref. Manual (ADC, GPIO, Clocks)

Motor Speed Control Project

1. Generate PWM waveform
2. Amplify the waveform to drive the motor
3. Measure motor speed
4. Measure motor parameters
5. Control speed with a PID controller



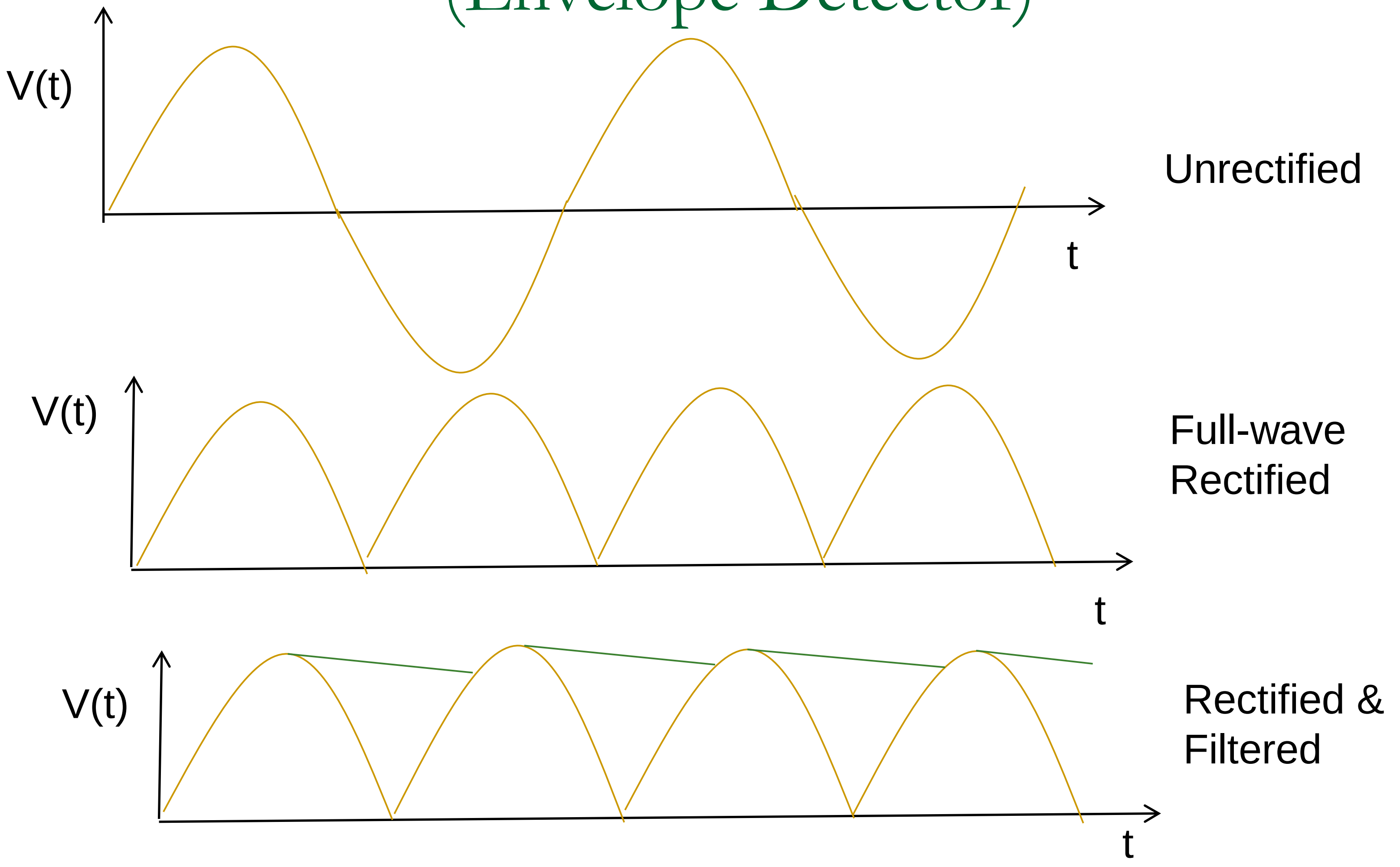
Typical analog input subsystem



Signal conditioning

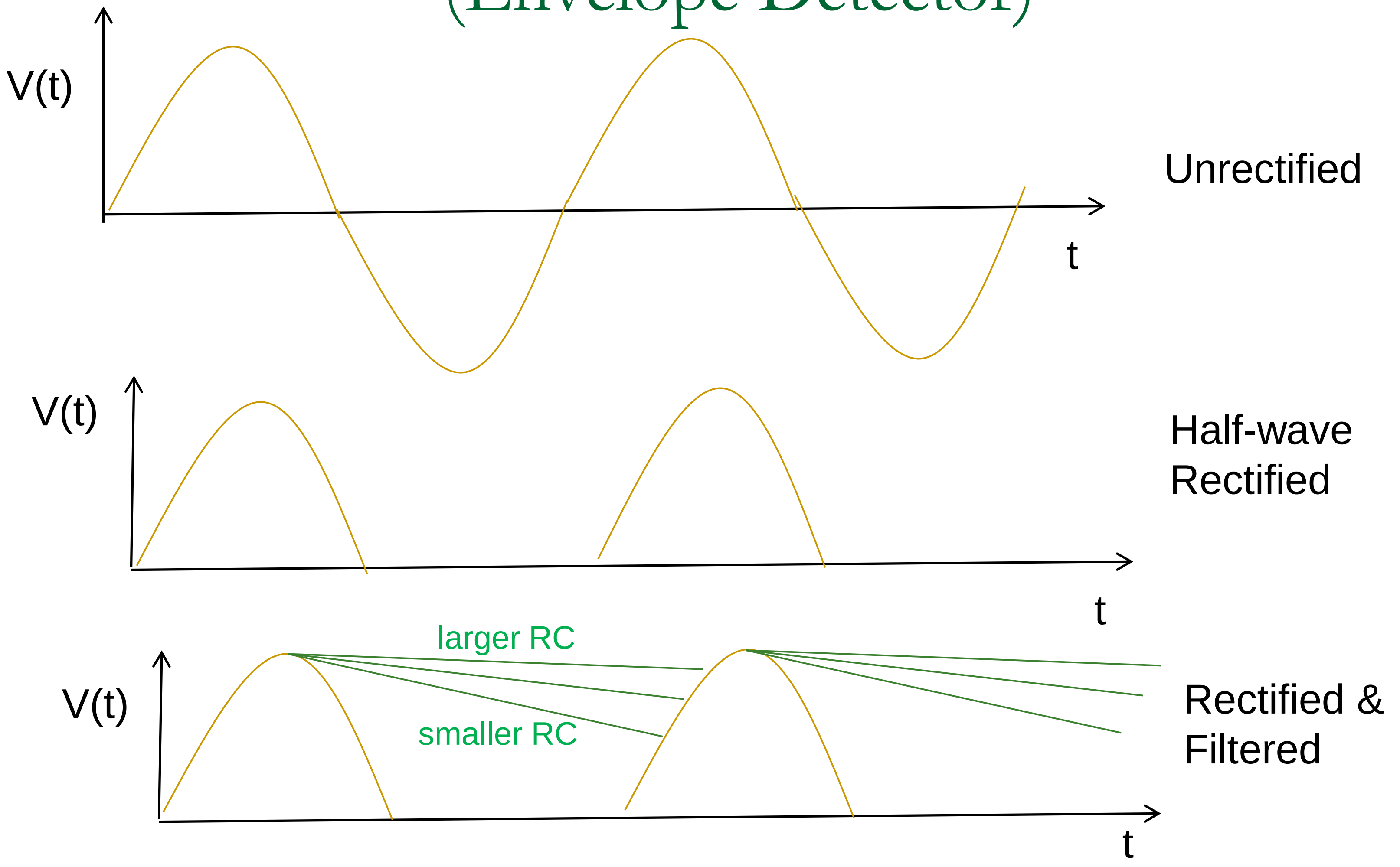
- Produce noise-free signal over A/D converter input range STM32L100-Discovery : [0v to 3v]
  - **Convert AC signal to DC form**
    - *Needed in this lab to measure tachometer signal amplitude*
  - **Amplify/attenuate voltage/current levels**
    - *Tachometer voltage might be > max ADC input range*
  - Bias (shift levels to desired range)
  - Filter to remove noise
  - Common mode rejection for differential signals
  - Isolation/protection (optical/transformer)

Example: AC to DC conversion
(Envelope Detector)

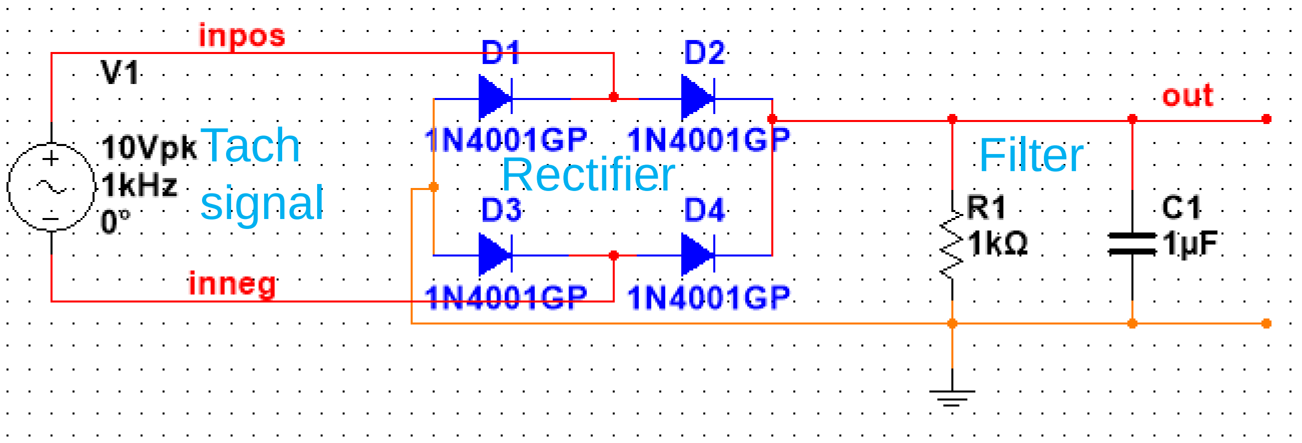


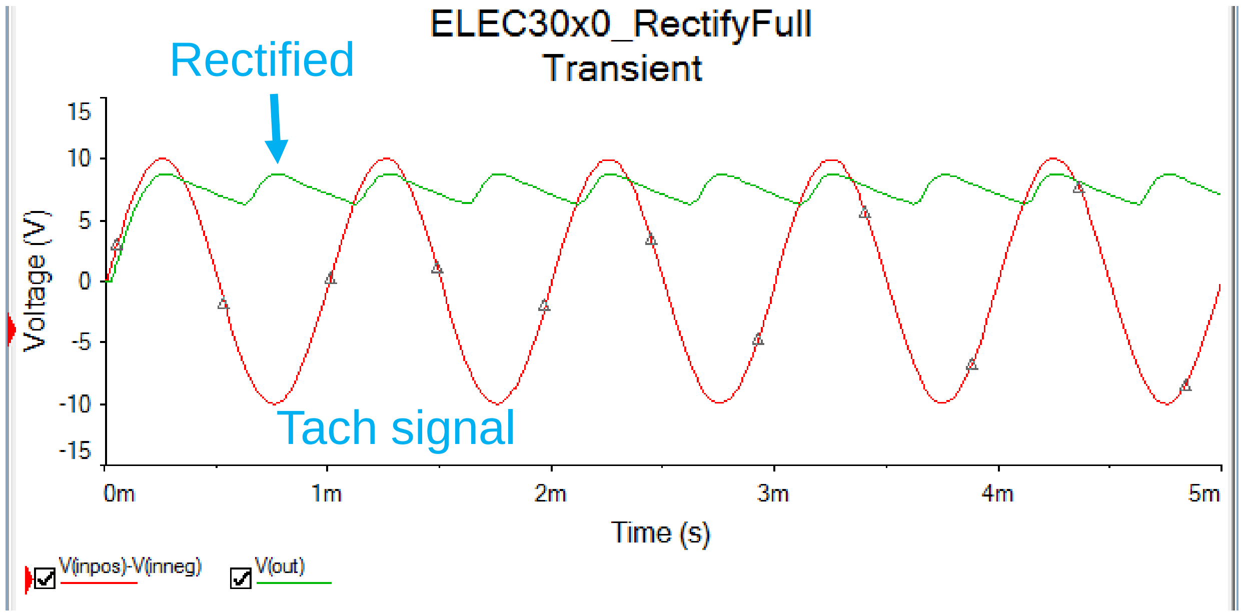
(May also choose half-wave rectified form – consider "ripple" in DC level)

Example: AC to DC conversion
(Envelope Detector)



Signal conditioning to measure tach signal amplitude



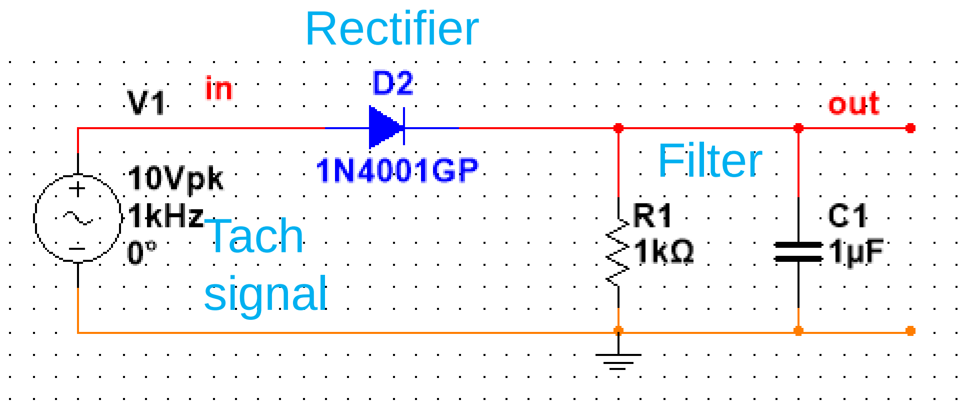


Rectified signal:

Is output "ripple" acceptable?

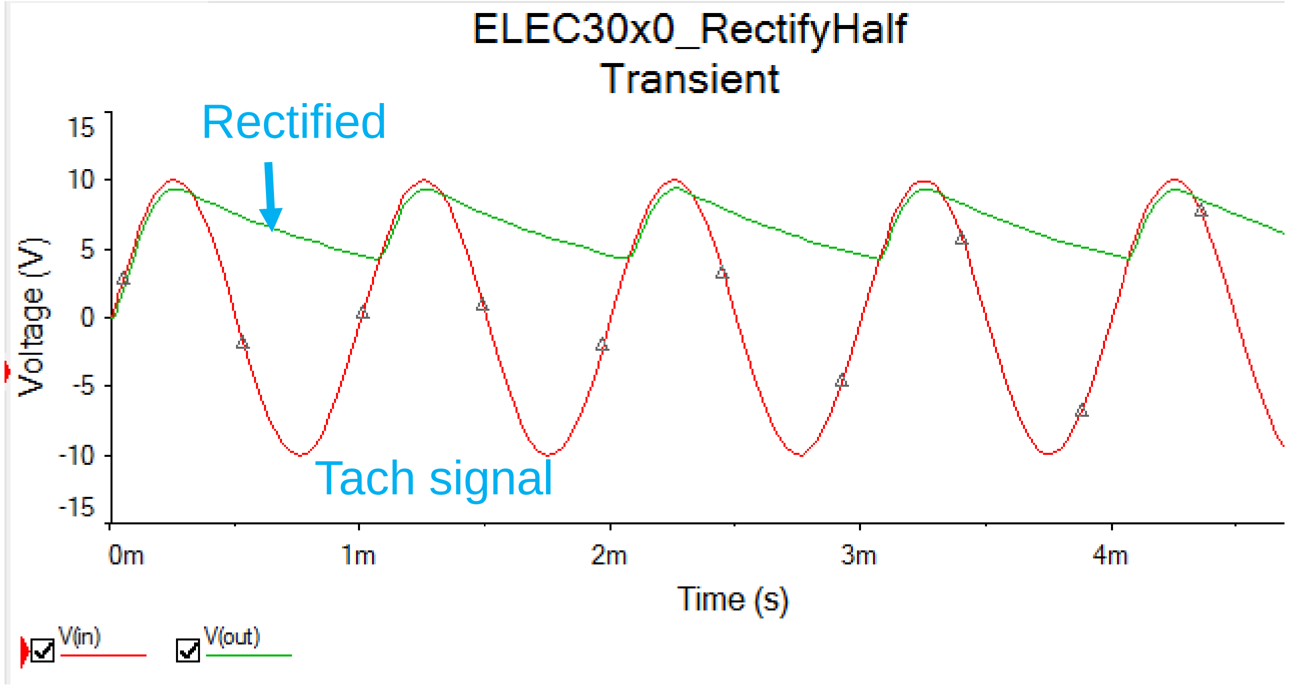
Is the DC level acceptable?

Signal conditioning to measure tach signal amplitude



Rectified signal:
Is output "ripple" acceptable?

Is the DC level acceptable?



Analog to digital conversion

- Given: continuous-time electrical signal

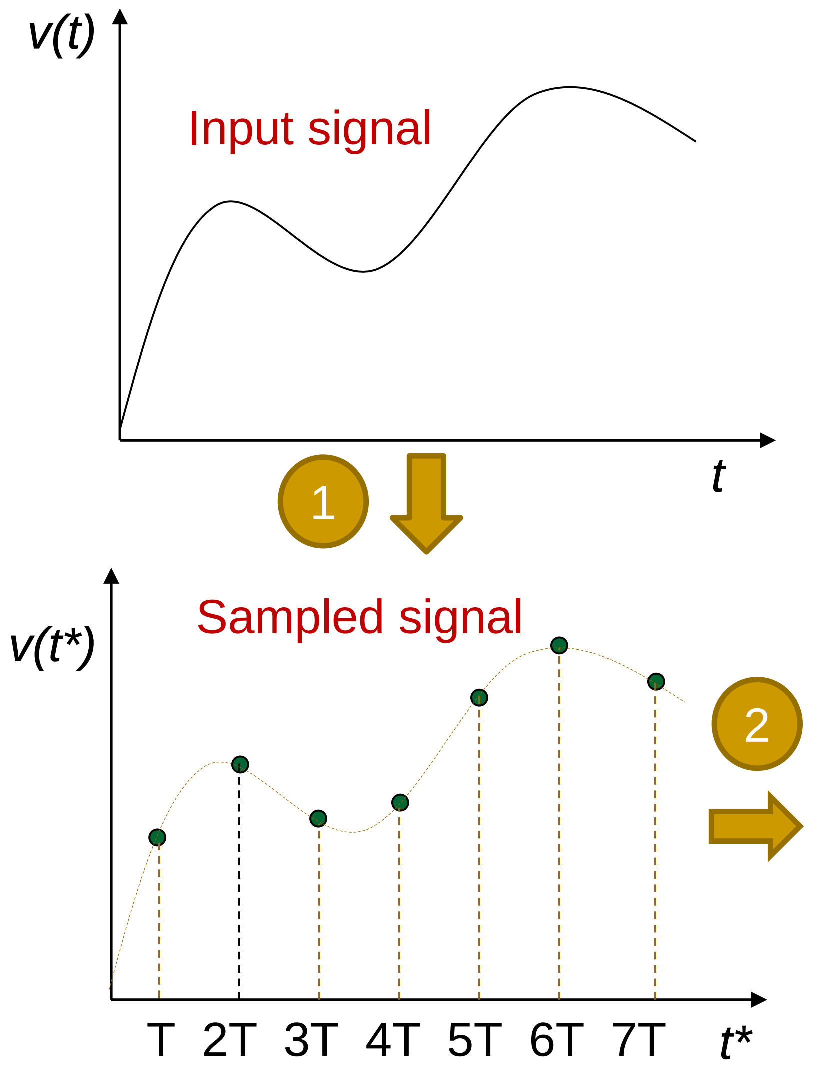
  $v(t),\ t >= 0$

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

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

  - $v(nT) = v(t)$ value measured at the n^{th} sample time
    - Quantized to one of 2^k discrete levels
    - Produces a k-bit number
  - T = “sampling time”
    - v(t) is “sampled” every T seconds
    - Sampling frequency $F_{sample} = 1/T$

A/D conversion process



Sampled data sequence:

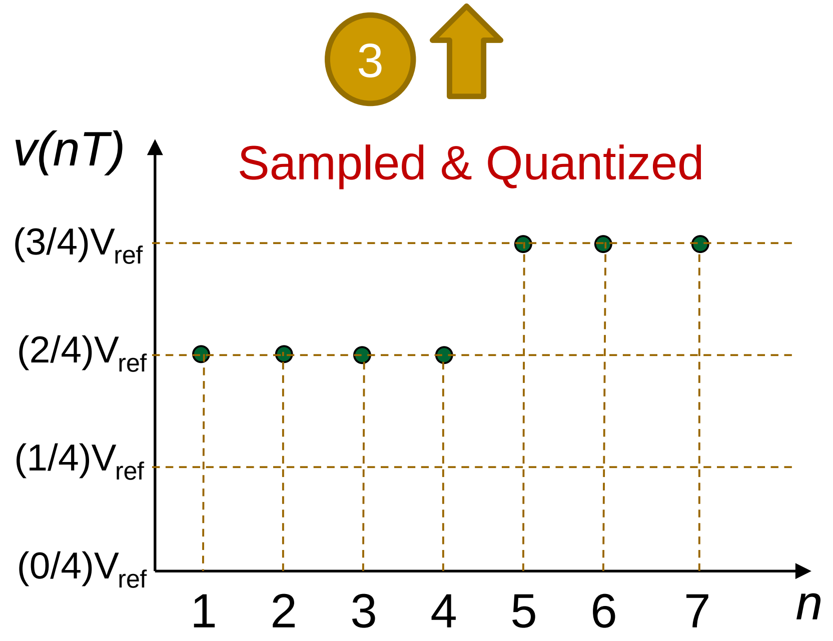
n= 1 2 3 4 5 6 7

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

$$\underbrace{10, 10, 10, 10, 11, 11, 11}$$

binary data

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



A/D conversion parameters

- Sampling rate, F (sampling interval T = 1/F)
  - *Nyquist rate* ≥ 2 x (highest frequency in the signal)
    - to reproduce sampled signals
    - CD-quality music sampled at 44.1KHz
      (ear can hear up to about 20-22KHz)
    - Voice in digital telephone sampled at 8KHz

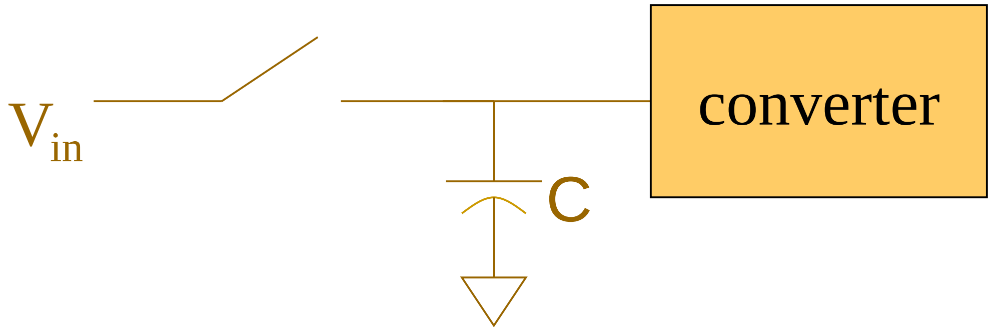
- Precision (# bits in sample value)
  - k = # of bits used to represent sample values
  - "precision" = step size between values = $(1/2^k) \times V_{range}$

    Ex. Temperatures [-20^{O}C…+60^{O}C]:

    If k=8, precision = 80^{O}C/256 = 0.3125^{O}C

- Accuracy = degree to which converter discerns proper level (error when rounding to nearest level)

Sample-and-hold



- Required if A/D conversion is slow relative to frequency of signal:
  - Close switch to “sample” V_{in} (charge capacitor C to V_{in})
    - Aperture (sampling) time = duration of switch closure
  - Open switch to “hold” V_{in} on C
  - Sample time often programmable.

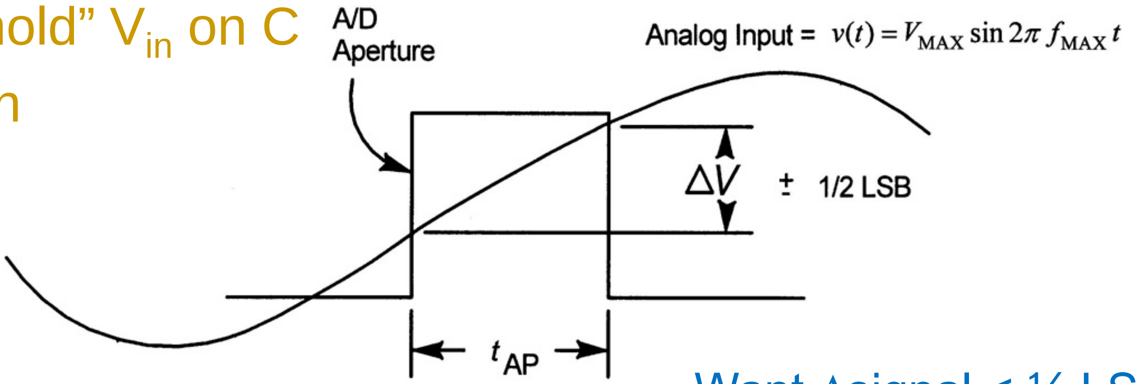


Figure 17-4 Aperture time error.

Want Δsignal < ½ LSB

Digital to analog conversion

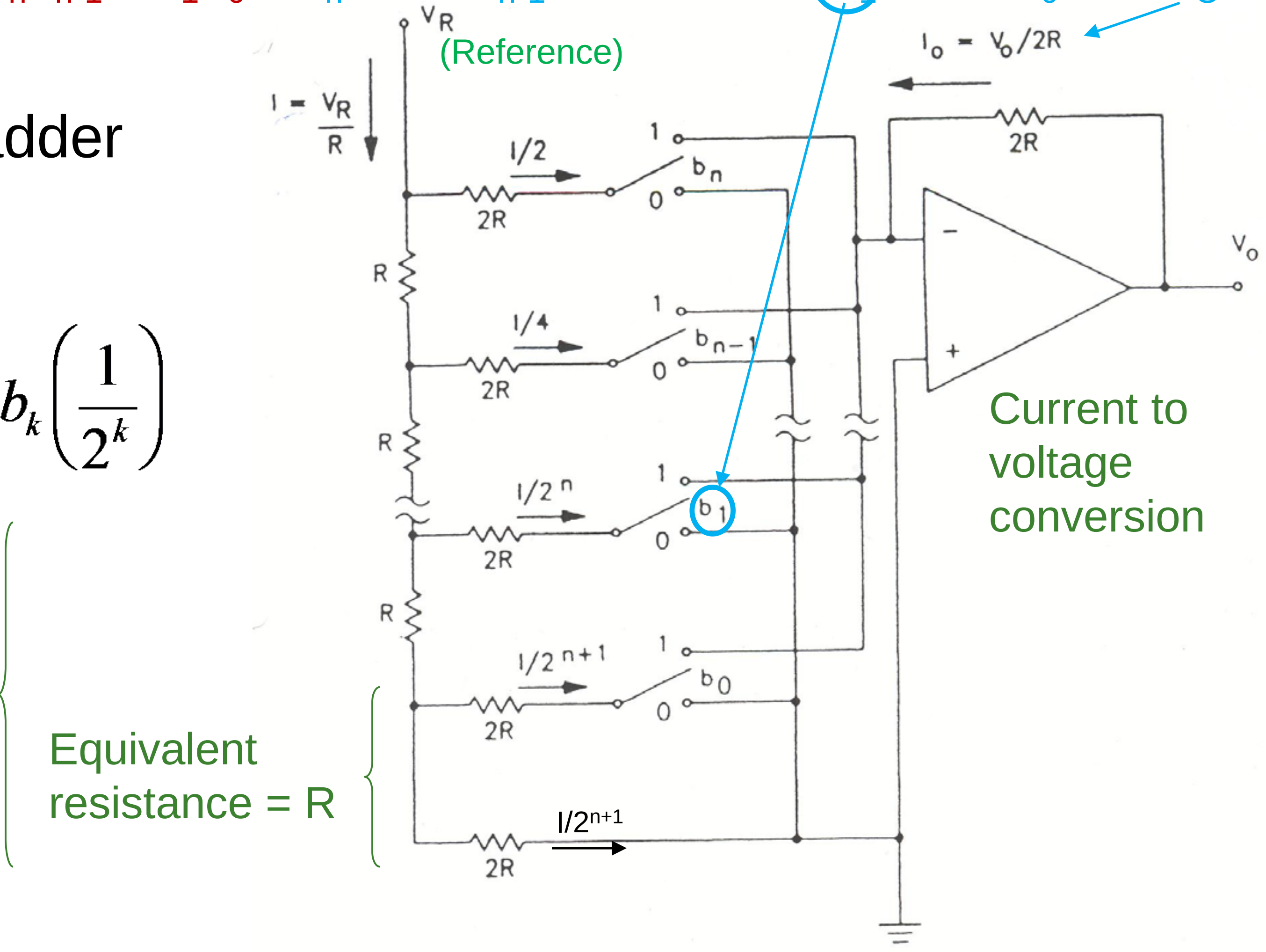
Number = $b_n b_{n-1} \ldots b_1 b_0 = b_n{*}2^n + b_{n-1}{*}2^{n-1} + \ldots + b_1{*}2^1 + b_0{*}2^0 = I_O$

R-2R Ladder Network

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

Equivalent resistance = R

Equivalent resistance = R



Current to voltage conversion

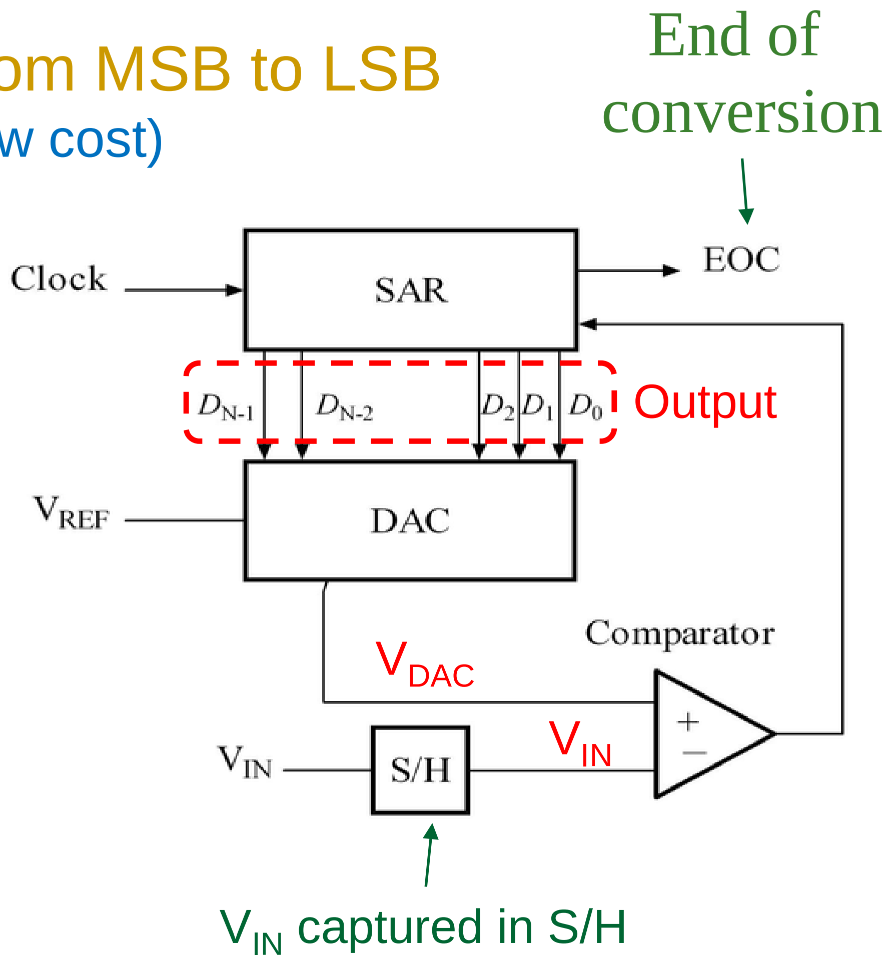
Successive approximation analog to digital converter (ADC)

- Determine one bit at a time, from MSB to LSB

  Used in most microcontrollers (low cost)

1. *Successive Approximation Register* (SAR) sets $D_{N-1} = 1$
2. *SAR* outputs $D_{N-1} \dots D_0$, converted by *DAC* to analog V_{DAC}
3. V_{DAC} is compared to V_{IN}
4. *Comparator* output resets D_{N-1} to 0 in SAR if $V_{DAC} < V_{IN}$
5. Repeat 1-4 for $D_{N-2} \dots D_0$ (one clock period per bit)

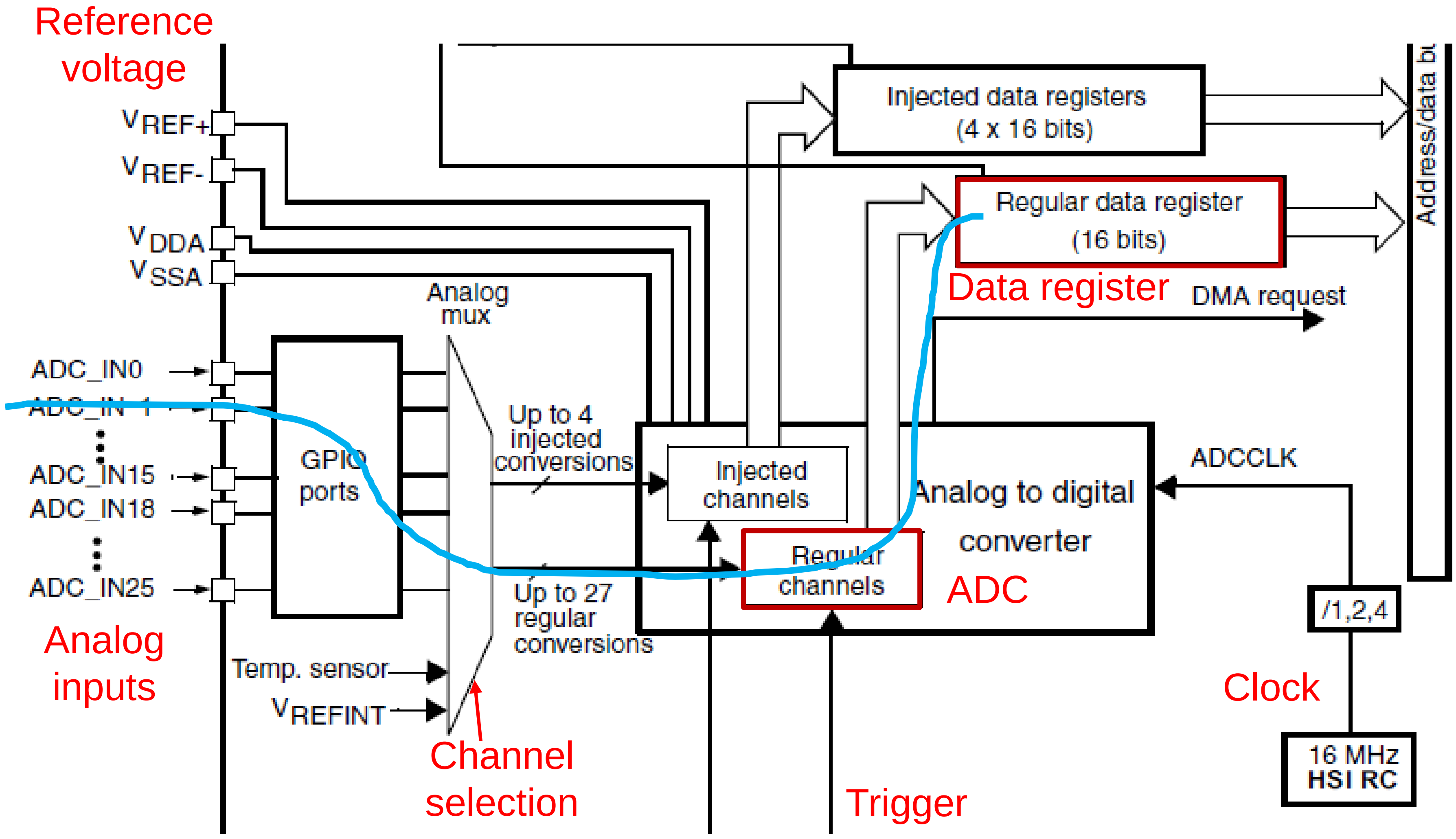
- Final SAR value $D_{N-1} \dots D_0$ is digital representation of V_{IN}



STM32L100RC Analog to Digital Converter

- Successive approximation ADC
- Input range: $V_{REF-} \leq V_{IN} \leq V_{REF+}$ (3.6 v max)
  - Discovery Board: V_{REF+} = VDDA pin, hard-wired to VDD (+3v)
    V_{REF-} = VSSA pin, hard-wired to GND (0v)
- Selectable resolution: 12, 10, 8, or 6 bits (default=12)
  - $T_{CONVERT}$ = 12, 11, 9, 7 clock cycles, respectively
- Programmable sampling time T_{SAMPLE} = 4 to 384 clock cycles
- Minimum conversion time 1µs ($T_{SAMPLE} + T_{CONVERT}$)
- 22 input channels
  - GPIO pins: ADC_IN[0:15], ADC_IN[18:21] - on designated GPIO pins
  - Temperature sensor & voltage reference: ADC_IN[16:17]
- Single or continuous conversions
- Scan mode for conversion of multiple channels
- Interrupt at end of conversion or end of sequence
- Trigger conversions with software or hardware (timers/EXTI)

ADC System Components



Using the ADC

- Setup
  - Connect voltage reference (hard-wired to 3v on Discovery)
  - Configure GPIO pin (select **analog mode** in MODER)
  - Enable HSI **clock** (for ADC conversion)
  - Enable ADC digital interface **clock**
  - Enable ADC
  - Select data format
  - Select sample trigger source
  - Select input channel(s)
  - Select conversion mode
- Trigger conversion
- Read results
- Adjust results as needed (calibrate, average, etc.)

ADC initialization

- Configure GPIO pin(s) as **analog** signals
  - Refer to *Pin Definition Table* in STM32L100 data sheet to determine which GPIO pins correspond to ADC_IN0..ADC_IN21
    - Example: GPIOA pins PA0 - PA7 = ADC_IN0 - ADC_IN7, respectively
  - Enable GPIOx clock in **RCC->AHB1ENR**
  - Select analog mode in **GPIOx->MODER** (disables pull-up/down resistors)
- Enable HSI clock in **RCC->CR**, which runs ADC conversions
  - RCC->CR |= RCC_CR_HSION; //HSION = bit 0 of RCC->CR
- Enable ADC1 clock in **RCC->APB2ENR** (for ADC digital interface)
- Power up the ADC
  - Set ADON bit in **ADC1->CR2** (control register 2)
  - Wait until ADONS = 1 in **ADC1->SR** (takes 3.5 - 4 µsec)
  - For power efficiency, shut off the ADC when not used
- Configure ADC options (data format, conversion mode, etc.)

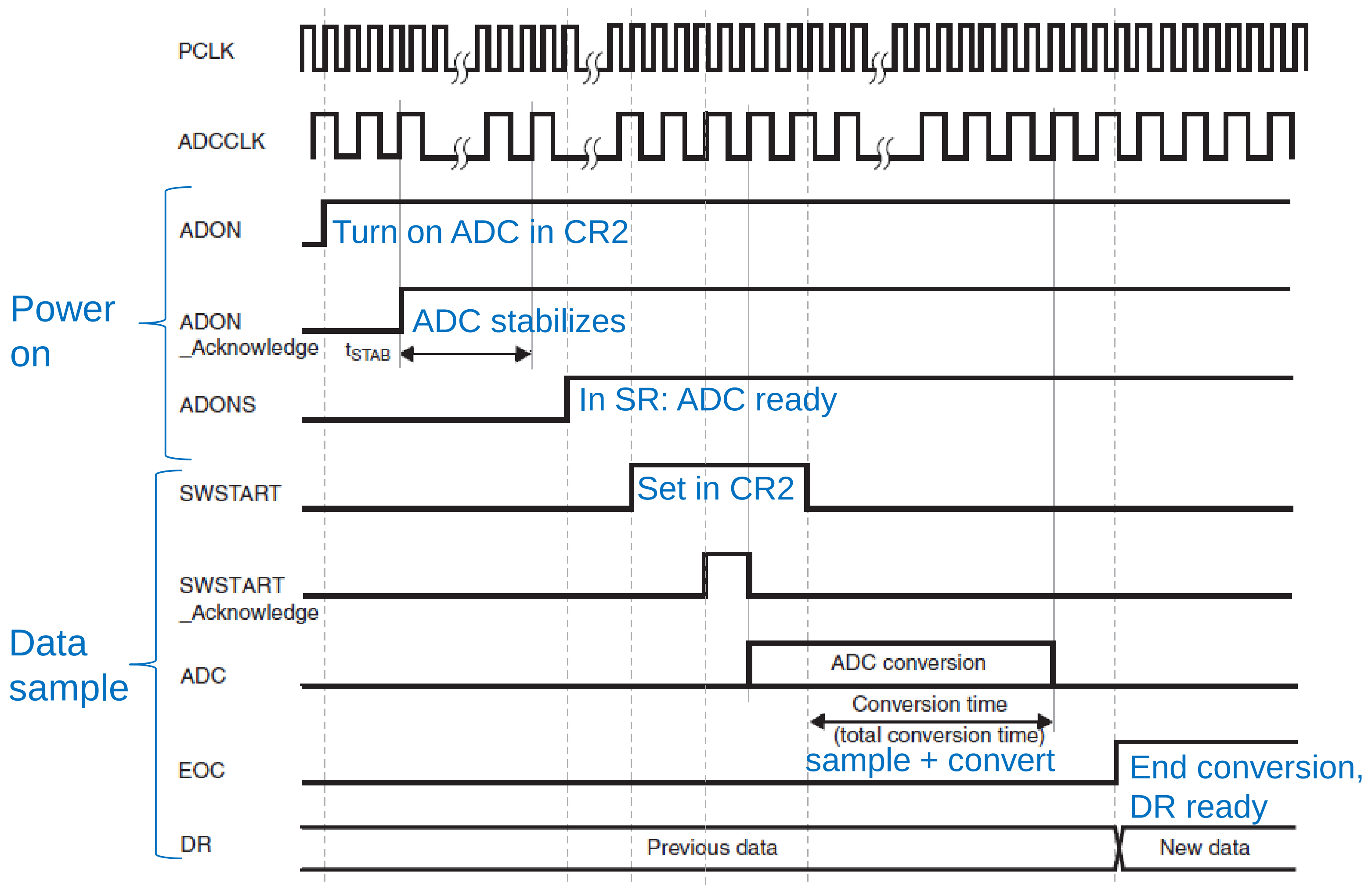
Data format

- 16-bit data register: ADC1->DR
  - Read as a 16-bit unsigned variable
- Data resolution can be 12, 10, 8 or 6 bits
  - Select via RES bits in **ADC1->CR1** (default RES = 00 => 12 bits)
  - 12-10-8-6 bits take 12-11-9-7 clock cycles, respectively, to convert (trade off resolution for speed)
- Data can be left or right-aligned within the data register
  - Select via ALIGN bit in **ADC1->CR2**
  - Right alignment (ALIGN = 0): 0000dddddddddddd (default)
  - Left alignment (ALIGN = 1): dddddddddddd0000

Conversion modes

- **Single conversion** (default: SCAN=0 in CR1, CONT=0 in CR2)
  - Select an input channel (SQ1 field in in **ADC1->SQR5**)
  - Start the conversion (software start or hardware trigger)
  - EOC sets when conversion is complete
  - Read the result in the DR
- **Scan mode** (enable with SCAN=1 in CR1)
  - Perform a **sequence** of conversions of designated input channels
    - Define sequence length in **ADC1->SQR1**
    - Select channels in **ADC1->SQR1**…**ADC1->SQR5** (channels can be in any order)
  - Start the conversion *sequence* (software start or hardware trigger)
  - EOC sets after each conversion (EOCS = 0) or after the entire sequence is complete (EOCS = 1). (EOCS in **ADC1->CR2**)
- **Continuous mode** (enable with CONT=1 in CR2)
  - Start 1st conversion/sequence (software start or hardware trigger)
  - Next conversion/sequence starts automatically after a conversion/sequence completes

ADC timing


PCLK
ADCCLK
ADON
Turn on ADC in CR2
ADON _Acknowledge
ADC stabilizes
t_{STAB}
ADONS
In SR: ADC ready
Power on
SWSTART
Set in CR2
SWSTART _Acknowledge
ADC
ADC conversion
Conversion time (total conversion time)
sample + convert
EOC
End conversion, DR ready
DR
Previous data
New data
Data sample

ADC conversion time

- Total conversion time = $\mathbf{T_{sampling}}$ + $\mathbf{T_{conversion}}$
  - $\mathbf{T_{conversion}}$ = 12/11/9/7 cycles for 12/10/8/6 bit resolution
  - $\mathbf{T_{sampling}}$ = sampling time, ***specified for each channel***
    - Options: 4, 9, 16, 24, 48, 96, 192, 384 clock cycles
    - 3-bit value SMPn[2:0] sets sample time for channel n
      - ADC1->SMPR3 configures channels 9-0
      - ADC1->SMPR2 configures channels 19-10
      - ADC1->SMPR1 configures channels 29-20

Example: Fastest conversion rate for 12-bit data

f_{ADCCLK} = 16MHz; min $T_{sampling}$ = 4; max res. = 12 bits

T_{total} = (4 + 12)/16MHz = 1us (1 Msamples/sec)

Example: Set $T_{sampling}$ = 24 clock cycles for ADC_IN8

ADC1->SMPR3 (reset value = 0x00000000)

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
SMP5[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

SMPx[2:0]: *Channel x Sample time selection (# clock cycles)*

000: 4 cycles, 010: 16 cycles, 100: 48 cycles, 110: 192 cycles
010: 9 cycles, 011: 24 cycles, 101: 96 cycles, 111: 384 cycles

Default is 4 cycles for each channel

```
// Set sample time for ADC_IN8 to 24 cycles
ADC1->SMPR3 &= ~ADC_SMPR3_SMP8;   //Clear SMP8 bits*
ADC1->SMPR3 |=   0x03000000;      //SMP8 = 3
```

**ADC_SMPR3_SMP8 = 0x03000000*

ADC control register 1 (ADC1->CR1)

Reset value = 0x0000 0000 (bold values below)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved					OVRIE	RES[1:0]		AWDEN	JAWDEN	Reserved				PDI	PDD
					rw	rw	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

RES[1:0]: resolution

00: 12-bit, T_{CONV} = 12 cycles
01: 10-bit, T_{CONV} = 11 cycles
10: 8-bit, T_{CONV} = 9 cycles
11: 6-bit, T_{CONV} = 7 cycles

EOCIE: end of conversion interrupt enable:

0: disable the interrupt
1: enable ADC interrupt when EOC sets

SCAN: enable scan mode

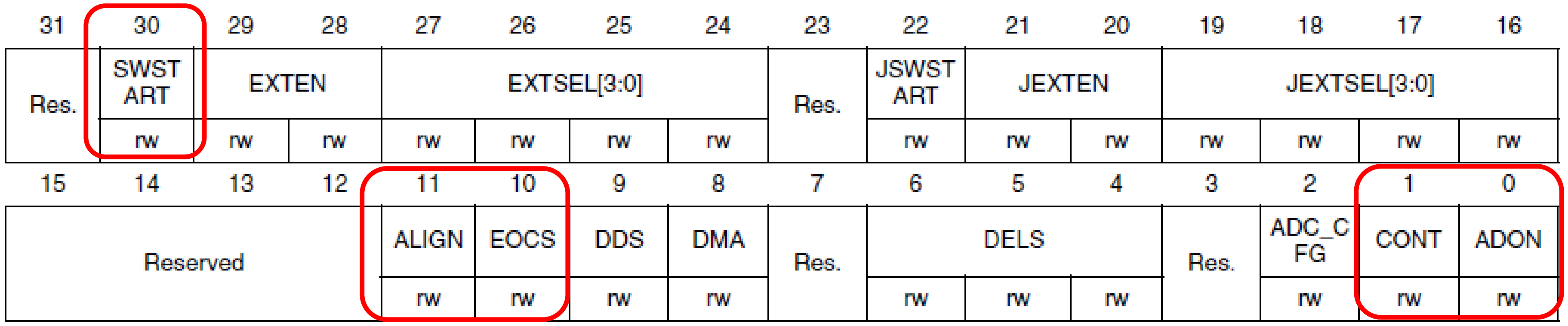
0: disable Scan mode
1: enable Scan mode
(convert inputs selected in ADC_SQRx)

Default setup:

- 12-bit sample
- single channel (no scan)
- no interrupt

ADC control register 2 (ADC1->CR2)

Reset value = 0x0000 0000



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SWSTART	EXTEN		EXTSEL[3:0]				Res.	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	Res.	DELS			Res.	ADC_CFG	CONT	ADON
				rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

SWSTART: Software “start” signal

Write 1 to start conversion

Resets when conversion starts

ALIGN: Data alignment in 16-bit result register

0: Right alignment (upper bits = 0)

1: Left alignment (lower bits = 0)

EOCS: End of conversion selection

0: EOC bit set at end of conversion *sequence*

1: EOC bit set at end of *each conversion*

ADON: Turn ADC on/off

0: Disable ADC

1: Enable ADC

```
//Turn on ADC
ADC1->CR2 |= 1;
```

CONT:

0: single conversion mode

1: continuous conversion mode

ADC status register (ADC1->SR)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						JCNR	RCNR	Res.	ADONS	OVR	STRT	JSTRT	JEOC	EOC	AWD
						r	r		r	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

ADONS: ADC ON state
1 = ADC ready to convert
Set/cleared by HW

EOC: End of Conversion
1 = Conversion complete (if EOCS = 0)
Sequence complete (if EOCS = 1)
Set by HW.
Clear by SW or by reading DR

```
//Wait for end of conversion (EOC=1)
while ((ADC1->SR & 0x02) == 0);
```

Other status bits:

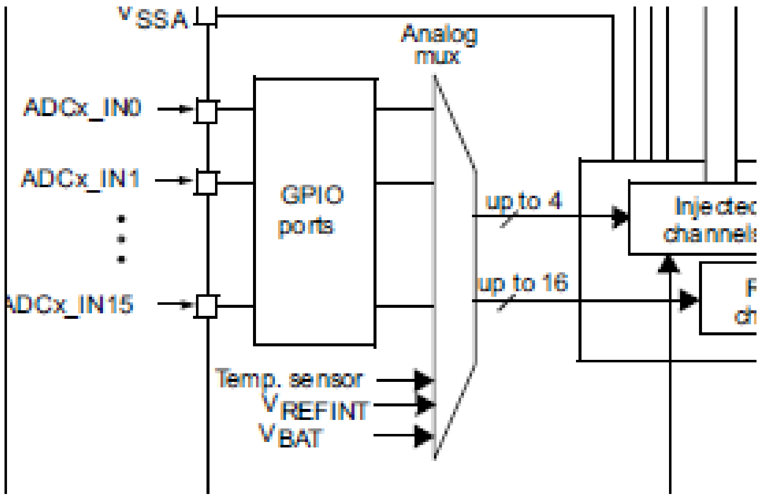
RCNR: Regular Channel Not Ready
1 = Regular conversion can be done
HW sets/clears

STRT: Start status
1 = Regular channel conversion has started
HW sets/clears

OVR: Overrun detected
1 = Regular conversion data lost (DR overwritten before read)
Clear in SW

Channel Selection

- **Sequence** of conversions (up to 28) can be done on any channel(s), in any order
- Specify #conversions in sequence via L[4:0] bits in the **ADC1->SQR1** register
  - Default is one conversion (L = 0)
- Specify channels in seq. by configuring the **ADC1_SQRx** sequence registers
  - ADC1->SQR5: conversions 1-6
  - ADC1->SQR4: conversions 7-12
  - ADC1->SQR3: conversions 13-18
  - ADC1->SQR2: conversions 19-24
  - ADC1->SQR1: conversions 25-28
  - In these registers, bits SQn[4:0] select channel # for the nth conversion in the sequence.
- For single-channel conversion, specify channel # in SQ1 of **ADC1->SQR5**

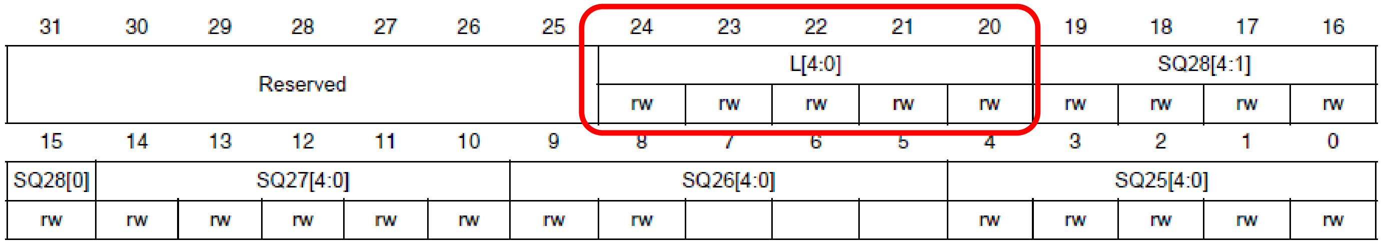


- ADC_IN16 connected to internal temperature sensor
- ADC_IN17 connected to internal reference voltage VREFINT

Example on next slide

Example: Select single conversion of ADC_IN8

ADC1->SQR1   (reset value = 0x00000000)



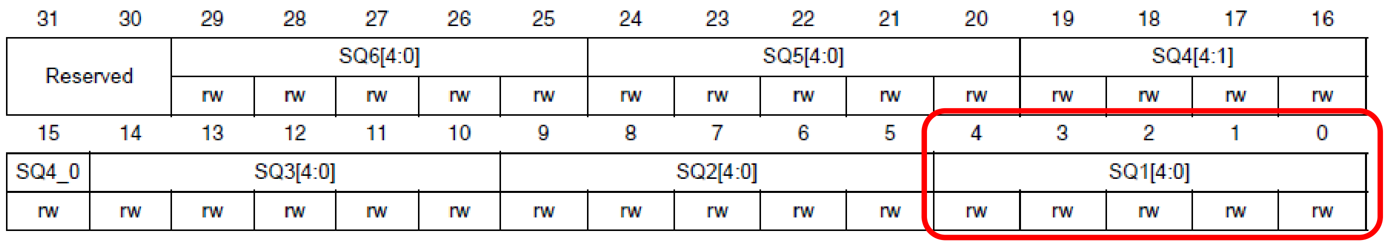
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved							L[4:0]					SQ28[4:1]			
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SQ28[0]	SQ27[4:0]					SQ26[4:0]					SQ25[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

#conversions in the sequence = L+1 (default L=0 selects a <u>single</u> conversion)

ADC1->SQR1 &= ~ADC_SQR1_L;   //set L=0   *(ADC_SQR1_L = 0x01F00000)*

(but - not really necessary, since L=0 is the default)

ADC1->SQR5   (reset value = 0x00000000)



31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved		SQ6[4:0]					SQ5[4:0]					SQ4[4:1]			
		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**
SQ4_0	SQ3[4:0]					SQ2[4:0]					SQ1[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

SQ1=x selects ADC_INx as first channel in a sequence

ADC1->SQR5 &= ~ADC_SQR5_SQ1; //clear SQ1 bits *(ADC_SQR5_SQ1 = 0x0000001F)*

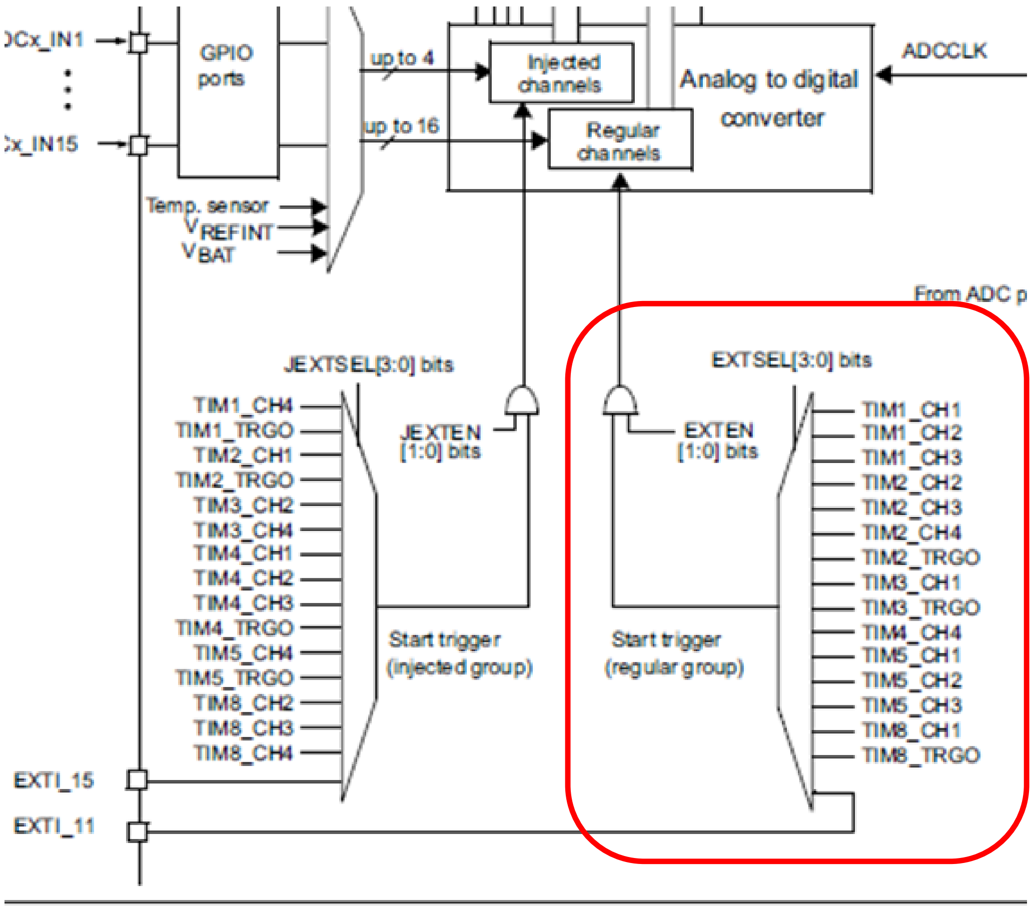
ADC1->SQR5 |=   0x00000008;   //SQ1=8 for ADC_IN8

Conversion Trigger Selection
(configure in ADC1->CR2)

- Software trigger:

//Set SWSTART bit to 1
ADC->CR2 |= ADC_CR2_SWSTART;
(0x40000000)

- External trigger
  - Select trigger detection mode via EXTEN bits (rising and/or falling edge of trigger)
  - Specify the trigger source via EXTSEL bits
    Ex. EXTSEL = 0111 for TIM3_CC1 event



Using the ADC (summary)

- Setup
  - Connect voltage reference (hard-wired to 3v on Discovery)
  - Configure GPIO pin (select analog mode in MODER, turn on GPIO clk)
  - Enable HSI clock for ADC conversion (RCC->CR)
  - Enable ADC digital interface clock (RCC-<APB2ENR)
  - Power on ADC (CR2)
  - Select data format (CR1,CR2)
  - Select conversion mode (CR1,CR2)
  - Select sample time (SMPR3)
  - Select input channel(s) (SQR1,SQR5)
- Trigger conversion (Software: CR2, or External: CR2)
- Read results (DR)
- Adjust results as needed (calibrate, average, etc.)

Working with ADC data samples

$$ADC1 \rightarrow DR = \frac{Vin}{Vref} \times 2^{\#bits}$$

- Several conversions may be needed
  - Average several samples of N to filter out noise
  - Compute approximate input voltage *Vin* from N
  - For a sensor, use sensor's transfer function to compute the physical parameter value (e.g. pressure) from *Vin*
  - Do additional computations based on this physical parameter (e.g. compute depth based on pressure)
  - Convert data to some other form (eg. ASCII characters, to send to a display)
- Numeric considerations
  - Consider resolution of measured data (eg. 12 bits) vs resolution of other data involved in calculations
  - Consider measurement errors in the ADC
  - Some program data may be in floating-point format

Lab Procedure

- Design and incorporate a rectifier & filter into your circuit to convert the tachometer output to a DC voltage level
  - Model in PSPICE to verify design.
  - Measure tachometer signal and rectifier/filter output with o'scope.
  - EEBoard waveform generator can be used to test the circuit without the motor.

- Modify software to add ADC initialization function and ADC input function (trigger conversion, wait for EOC, read result)
  - Consider averaging some # of samples
  - EEBoard waveform generator can be used to test the circuit without the motor.

- Measure: tachometer signal amplitude, rectifier/filter voltage output & ADC value for each of the 10 speed settings

- Plot:
  - ADC value vs tachometer output amplitude
  - ADC value vs. ATD input voltage (rectifier/filter output)
  - ADC value vs. PWM signal duty cycle