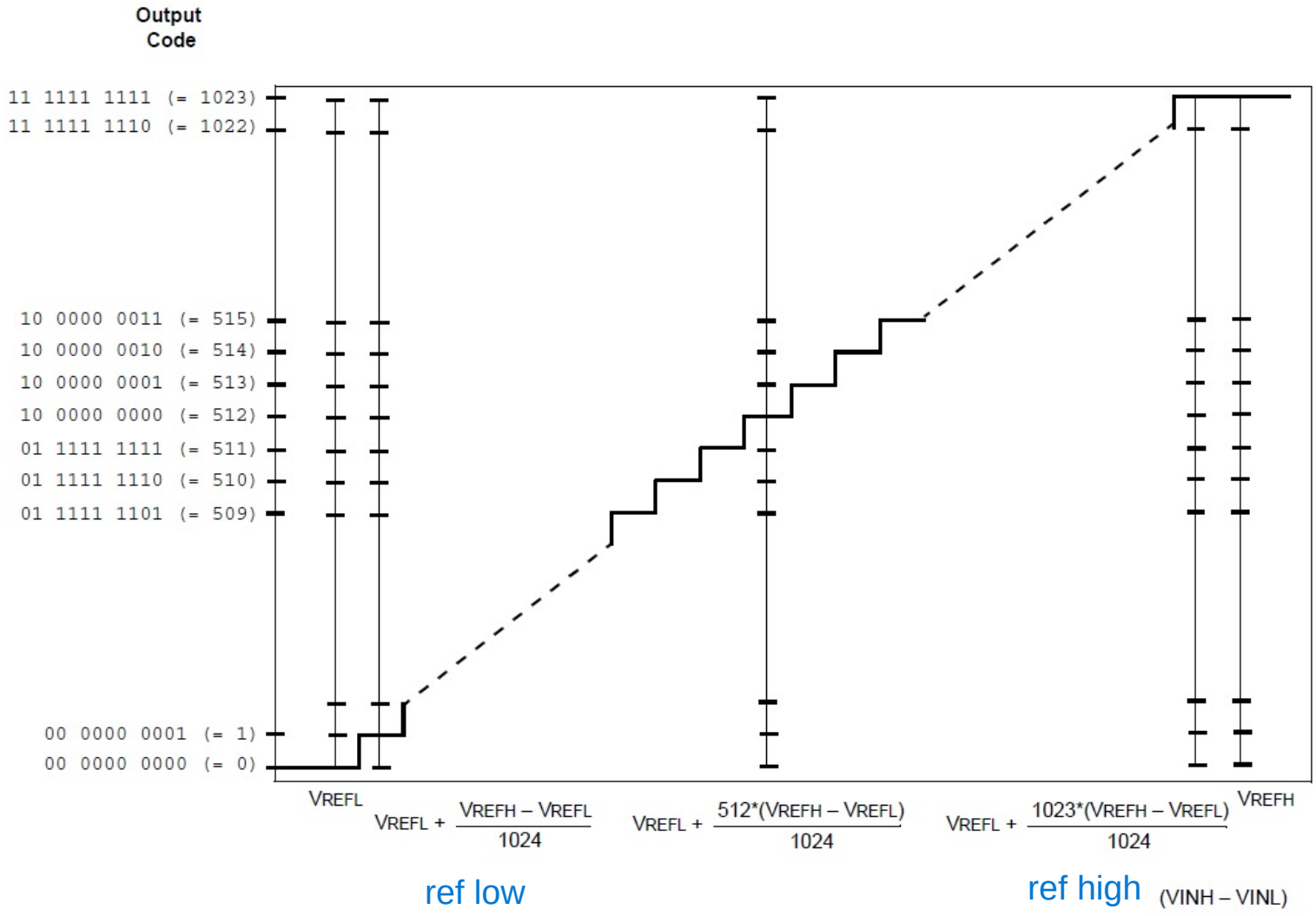
Embedded Systems
3.6 - ADC (Clicker 2)

Enrico Simetti

ISME - Interuniversity Research Center on Integrated Systems for Marine Enviroment
DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering
University of Genova, Italy
enrico.simetti@unige.it

What sampling means



depending on the diff, we get a qauntization? error, half the err is boh

dsPIC33 ADC Modules I

- Two ADC modules
- Up to 1 Msps
- 10-bit or 12-bit operation modes
- Up to 32 analog pins (ADC 1)
- Up to 16 analog pins (ADC 2)
- 4 differential S/H amplifiers (channels)
- 1 converter — single ADC model -> single converter always
- Channel scan mode
- 16 word buffer for conversion results — All results stored in this 16 word buffer
- Selectable conversion trigger source

selectable trigger: not necessarily manually but also via interrupt

dsPIC33 ADC Modules II

only channel that can be connected to all inputs

X : the 4 different channels

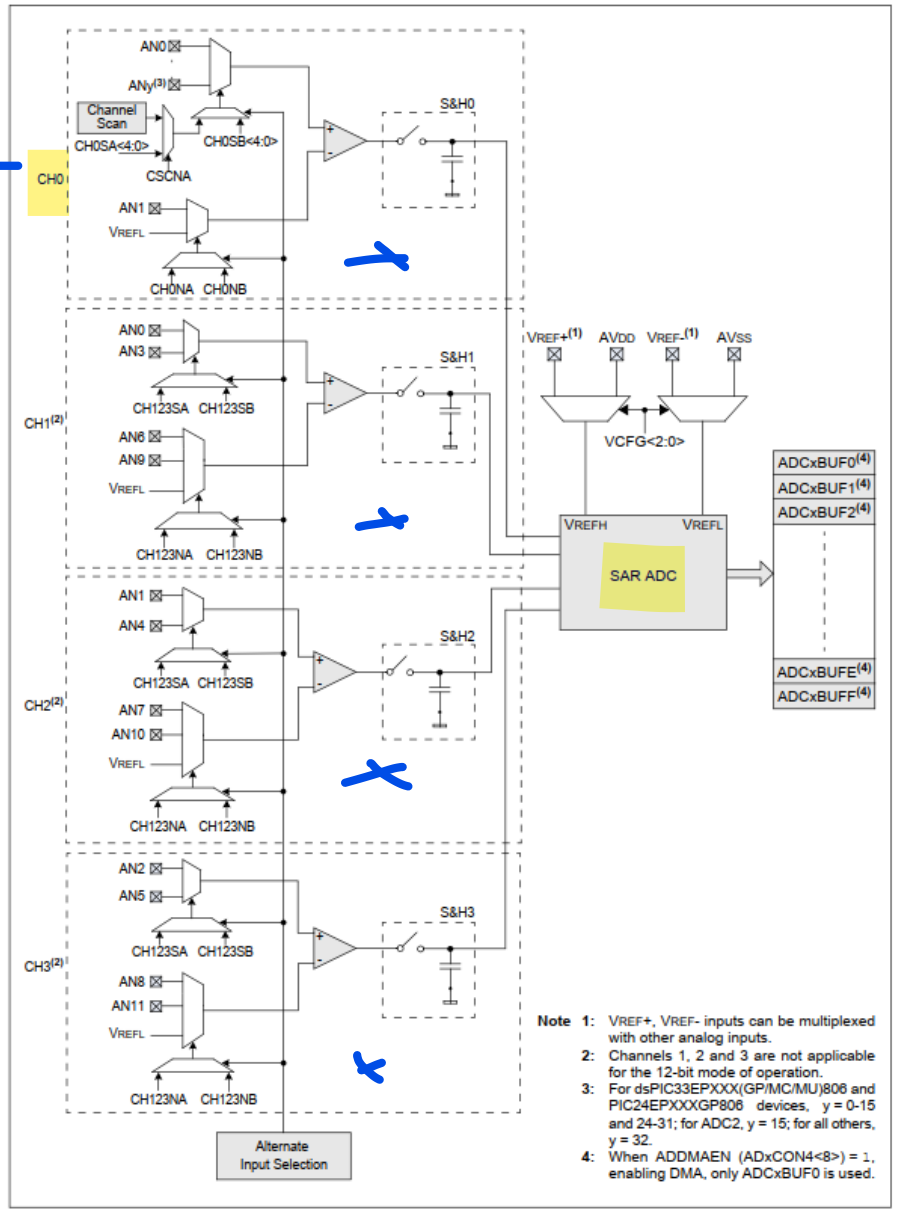
- connected to ground
+ connected to the analog signal we want to signal
output: analog signal

but can be changed

channel 1-3 have limited configurability

ch0: scan to get inputs from 1 to 5

FIGURE 23-1: ADCx MODULE BLOCK DIAGRAM



alter value: charge a capacitor and keep on following the signal.
when we hold the capacitor part stays constant, then charge it again if we want to follow signal

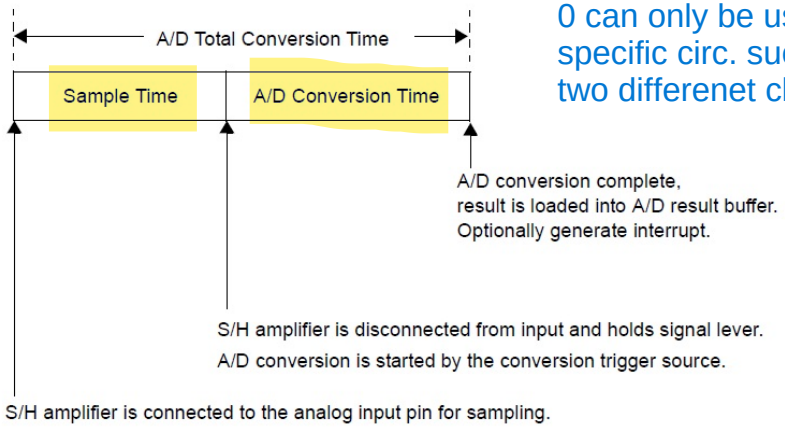
converter -> constant
beacause adc's higher bit is 1 only if converter > signal

Sampling and Conversion

you need to let the capacitor charge: sampling time needed (wait)

specificable via adc internal clock (multiple of TCy)

- Tad is ADC time period
- Tad duration is user defined [0.5 Tcy – 32 Tcy]
- Conversion time duration (Tconv): *fixed to 12 Tad*
- Sample time duration (Tsamp) can be
  - fixed (specified in number of Tads [0 Tad - 31 Tad])
  - determined by user software (by clearing SAMP bit)
- Sampling rate = 1/(Tconv+Tsamp)



0 can only be used in specific circ. such having two differenet channels

min time: 154 ms, if we configure it to go faster -> not gonna work

For correct A/D conversions, the A/D conversion clock (Tad) must be selected to ensure a minimum Tad time of 154 nsec (for VDD = 5V)

ADC triggers I

Sampling period start

- automatic (at the end of previous conversion)
- manual (by setting SAMP bit in software)

Sampling period end (=conversion time start)

- automatic (if Tsamp is fixed)
- manual (by setting SAMP bit in software)
- (driven by other events)

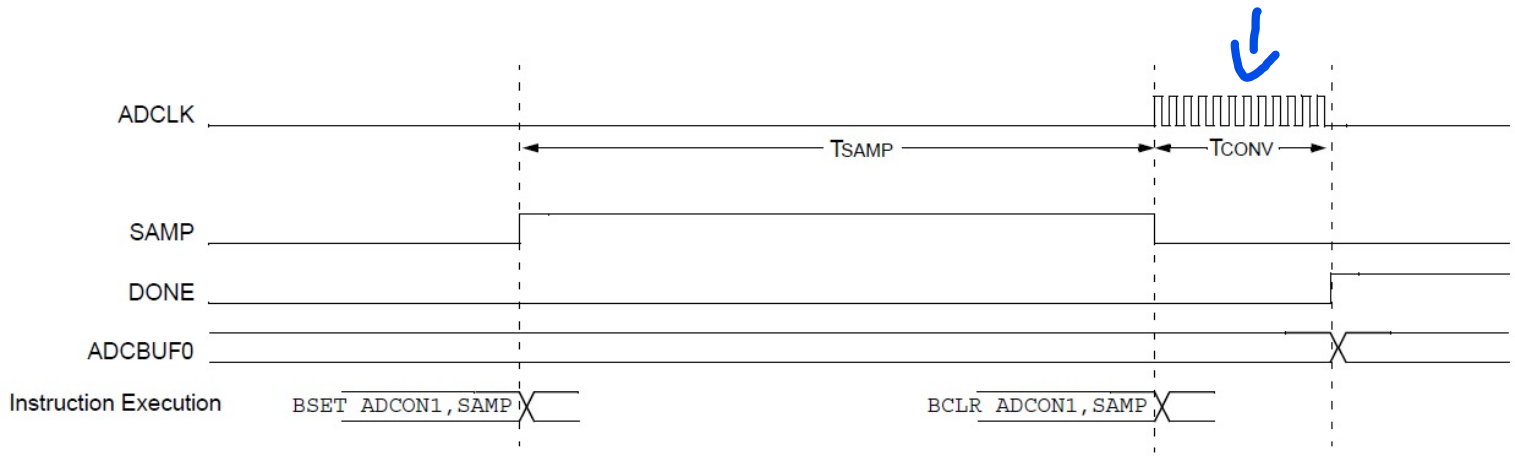
ADC triggers II

when to begin sampling
when to stop sampling before conversion

manual config

manual start - manual end

12 clock signals: conversion

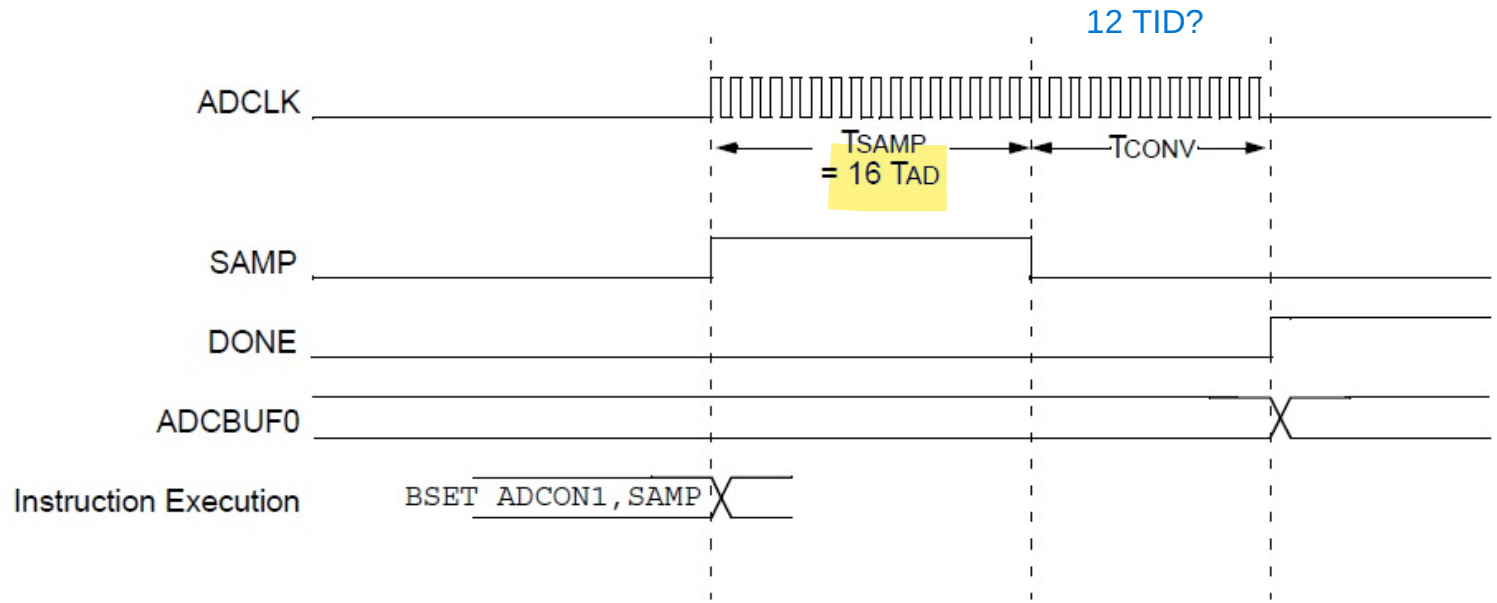


ASAM = Automatic SAMpling

```
ADxCON1bits.ASAM = 0;   // x = 1 or 2, specifies ADC module
ADxCON1bits.SSRC = 0;
```

ADC triggers III

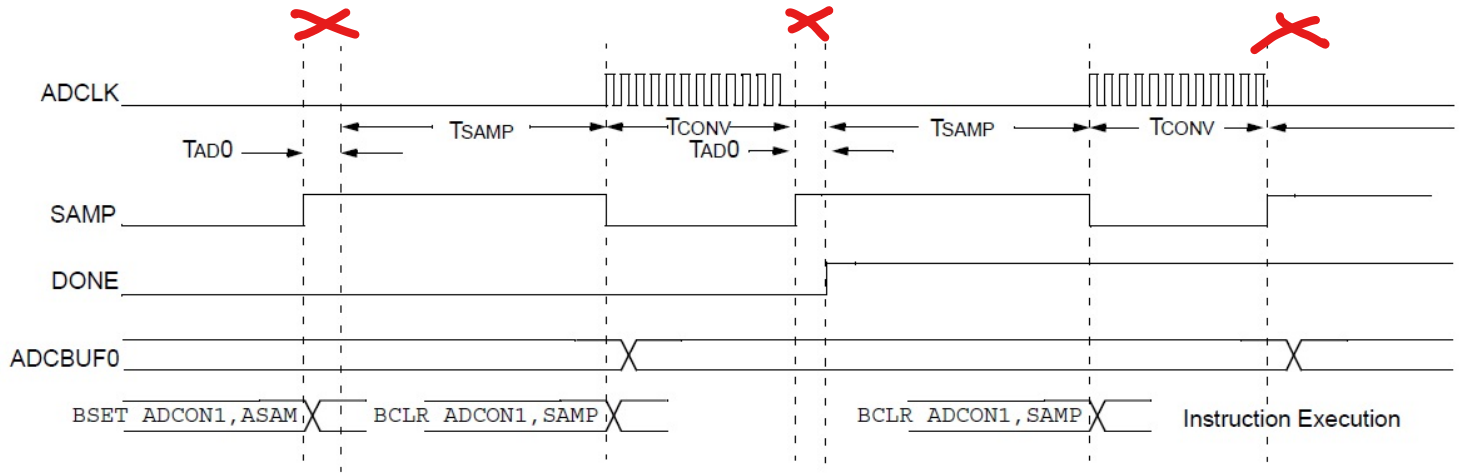
manual start - automatic end




```
ADxCON1bits.ASAM = 0;   off -> manual sampling
ADxCON3bits.SAMC = 16; // sample time 16 Tad  fixed duration
ADxCON1bits.SSRC = 7; // conversion starts after time specified
    by SAMC
```

ADC triggers IV

automatic start - manual end



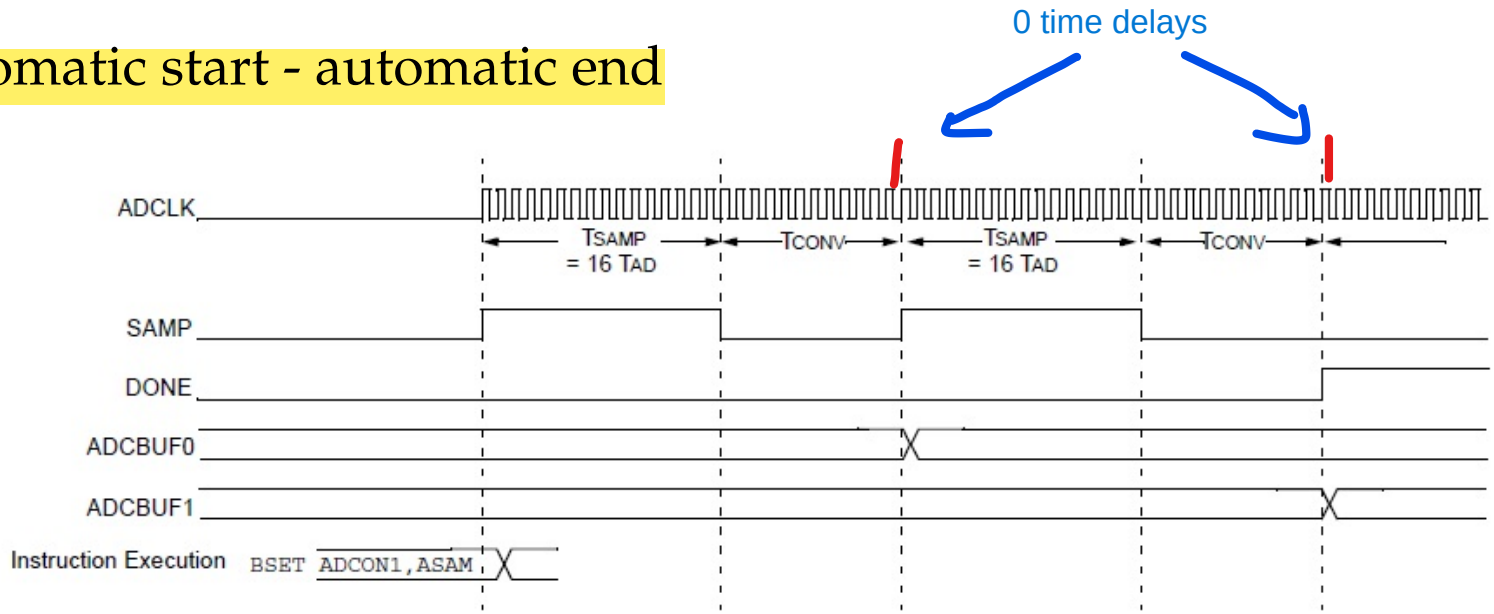

```
ADxCON1bits.ASAM = 1;
ADxCON1bits.SSRC = 0;
```

At the end of previous conversion, automatically start sampling

off -> manual end

ADC triggers V

automatic start - automatic end

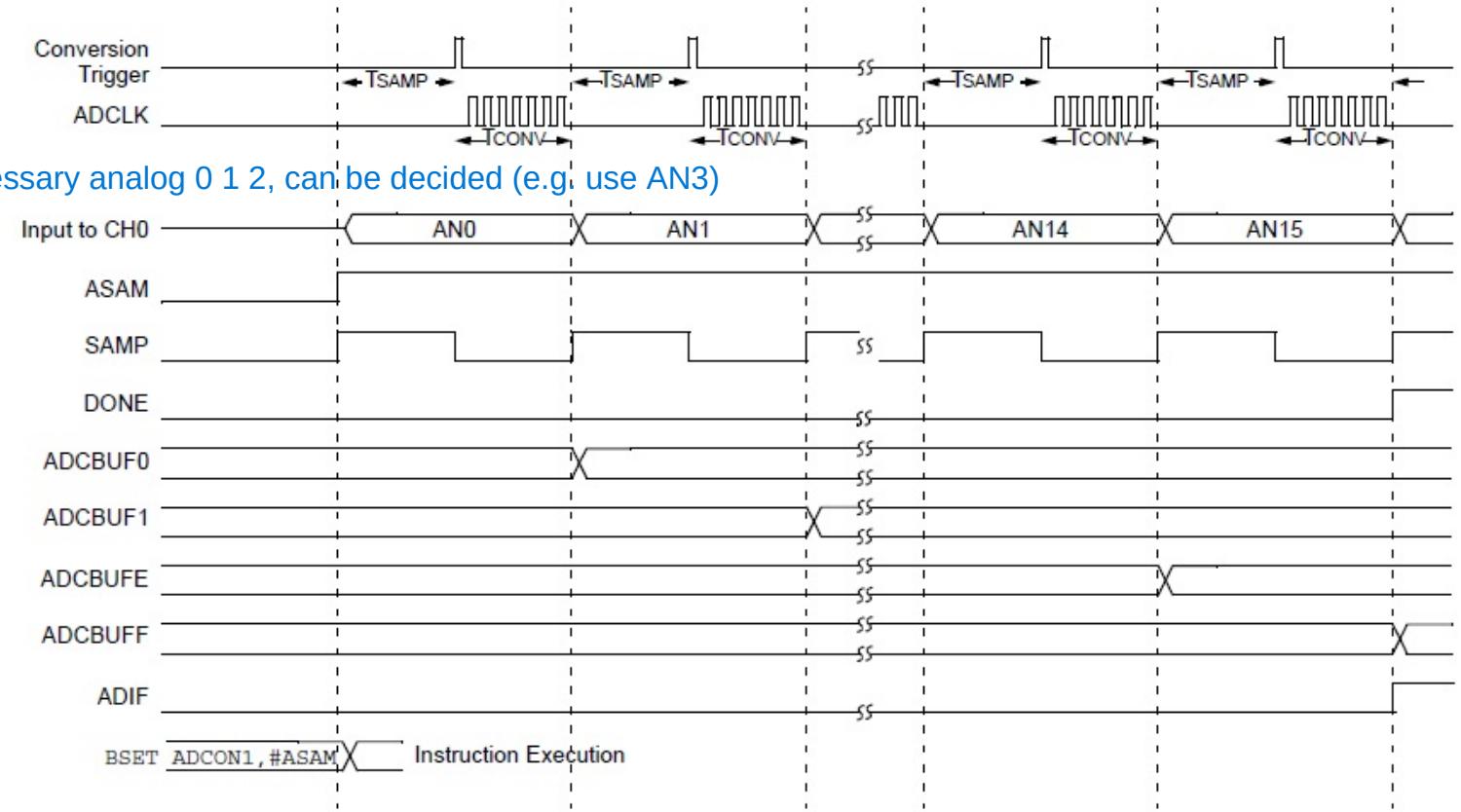


sampling begins at the end of conversion and it's fixed -> full troughput (no extra delays)

```
ADxCON1bits.ASAM = 1;
ADxCON3bits.SAMC = 16; // sample time 16 Tad
ADxCON1bits.SSRC = 7; // conversion starts after time specified
    by SAMC
```

ADC Scan mode

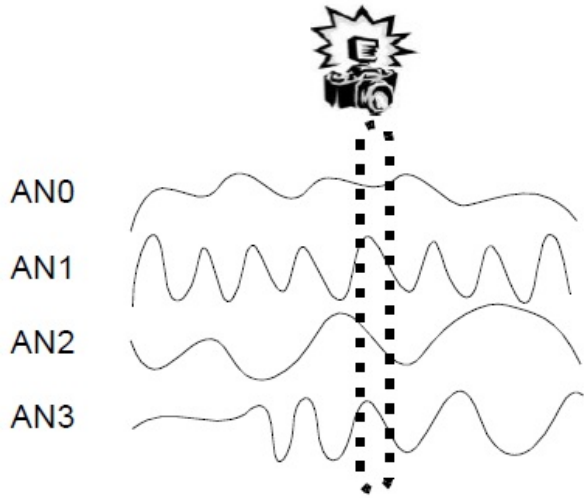
(analog from 0 to 7)



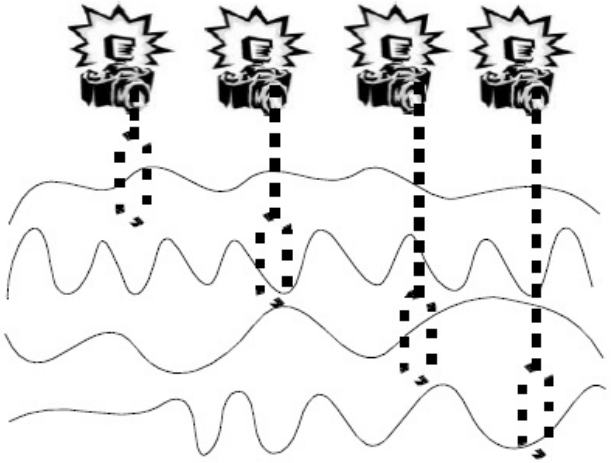

```
ADxCON2bits.CSCNA = 1; // scan mode
```

ADC Multiple Channels

why are there 4 channels?
multiple channels allow us to sample simultaneously and not sequentially
conversion is still sequentual -> what is the benefits?
at the end of the conversion if i did a simultaneous sampling, the 4 signals are referred to the same time period (on hold at the same time)
ptherwise i have 4 diff instance of time -> int this csae asyncronous programming
depends on the situation, if signals not related or slow time rate it doesn't matter to be sincronous



Simultaneous Sampling



Sequential Sampling

1 at the very end of conversion

REGISTER 23-1: ADxCON1: ADCx CONTROL REGISTER 1

R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
ADON	—	ADSIDL	ADDMABM	—	AD12B(1)	FORM<1:0>	
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0, HSC	R/C-0, HSC
SSRC<2:0>			SSRCG	SIMSAM	ASAM(3)	SAMP	DONE(3)
bit 7							bit 0

these 5 are the one we are going to use

ADxCON1: ADC Control register 1.

- *ADON*: turns on the ADC module
- *SSRC*: selects how the conversion should start (0 = manual, 7 = internal counter) vedi esempi
- *ASAM*: selects how the sampling should start
- *SAMP*: sample enable bit
- *DONE*: tells when the A/D conversion is done

ADC related registers II

REGISTER 23-2: AD1CON2: ADC1 CONTROL REGISTER 2

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0
VCFG<2:0>			—	—	CSCNA	CHPS<1:0>	
bit 15							bit 8

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	SMPI<4:0>					BUFM	ALTS
bit 7							bit 0

AD1CON2: ADC Control register 2.

- *CSCNA*: selects if inputs should be scanned — activates scan mode
- *CHPS*: selects the channels — select the channels we want to use (CH0 - 1 - ecc)
- *SMPI*: after how many conversions when the IF should be set to 1

  trigger a flag
  we are working at 1 microsecond time

ADC related registers III

REGISTER 23-4: ADxCON3: ADCx CONTROL REGISTER 3

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	—	—	SAMC<4:0>(1)				
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS<7:0>(2,3)							
bit 7							bit 0

ADxCON3: ADC Control register 3.

- *SAMC*: how long the sampling should last [0 Tad - 31 Tad] (matters *only* if SSRC = 7)
- *ADCS*: selects how long is one Tad [1/2 Tcy - 32 Tcy]

Time of ADC as a function of Tcy

ADC related registers IV

2 possibilities: CH0 -> S positive, N -> Negative
S dictate what is the positive output and viceversa
A - B: cionfiguration of ADC that can be switch thanks to that bit??

ADxCHS0: selects the inputs to channel 0

REGISTER 23-7: ADxCHS0: ADCx INPUT CHANNEL 0 SELECT REGISTER

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB	—	—	CH0SB<4:0>(1)				
bit 15							bit 8

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA	—	—	CH0SA<4:0>(1)				
bit 7							bit 0

positive input for channel

negative input for channel

ADC related registers V

ADxCHS123: selects the inputs to channels 1, 2 and 3

REGISTER 23-6: ADxCHS123: ADCx INPUT CHANNEL 1, 2, 3 SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	CH123NB<1:0>		CH123SB
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	CH123NA<1:0>		CH123SA
bit 7							bit 0

Results will be found in registers *ADCBUF0* through *ADCBUFF*

ANSELx: sets pins as analog (1) or digital (0). As a *thumb rule*, always set each ANSELx to 0 and then toggle to 1 only the pins required to be analog

↳ Analog selector x: A, B, C ... ??

these registers are important, failing to configure them will result in errors during the conversion

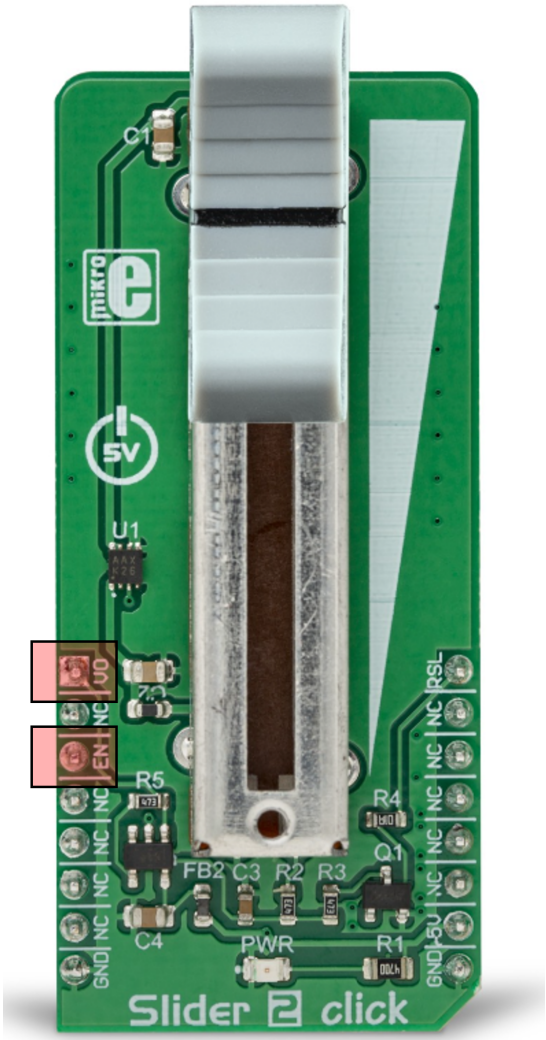
put all reg to 0 (everything is digital) then we specify which ones need to be analog signals, since we will use only 1 or 2 analog signals

Configuring the ADC

1. select your T_{AD} ADC time and set ADxCON3bits.ADCS (suggested value is 8 for our board)
2. select how sampling begins
   - 2.1 manual: ADxCON1bits.ASAM = 0;
   - 2.2 automatic: ADxCON1bits.ASAM = 1;
3. select how sampling ends and conversion begins
   - 3.1 manual: ADxCON1bits.SSRC = 0;
   - 3.2 automatic: ADxCON1bits.SSRC = 7; and then choose how long sampling lasts by setting ADxCON3bits.SAMC;
4. choose how many channels you want to use setting ADxCON2bits.CHPS
5. choose the positive input to the channels by setting ADxCHS0bits.CH0SA; and ADxCH123Sbits.CH123SA;
6. set the input pins as analog (value 0) in ANSELx and the rest as digital (value 1)
7. if more than 1 channel is used, choose sequential or simultaneous sampling using AxDCON1bits.SIMSAM
8. if sequential set ADxCON2bits.SMPI; to number of inputs - 1; if simultaneous take into account the number of channels used
9. finally, turn ADC on with ADxCON1bits.ADON = 1;

Negative usually connedcted to ground, in most application it won't need to be changed

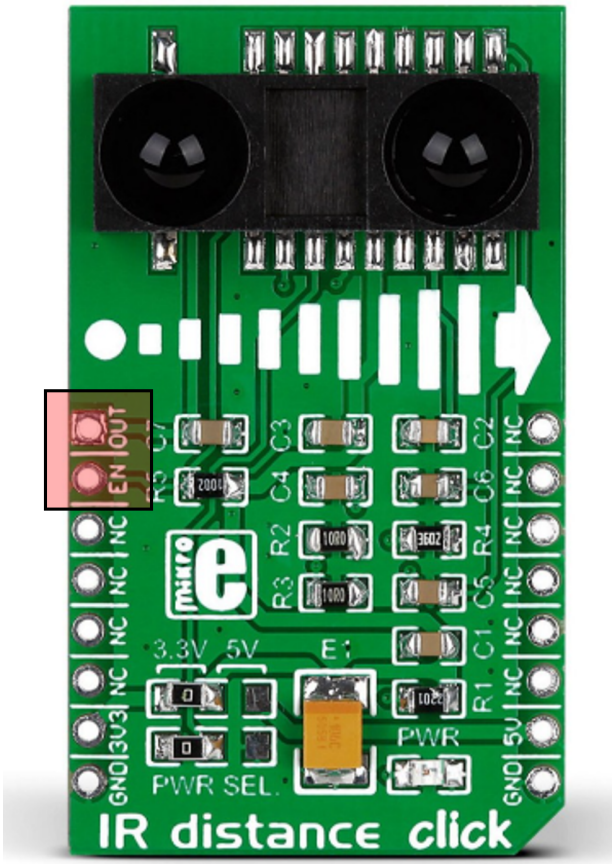
Slider 2 Click Specs



Enable board, then we can sample and get output

- Linear potentiometer, to be plugged in one of the MikroBUS sockets.
- The output voltage is on 'VO' pin.
- The EN pin is required to be high for the device to work properly
- The output voltage range is between 0 and 4.096 V.

IR Distance Sensor Specs



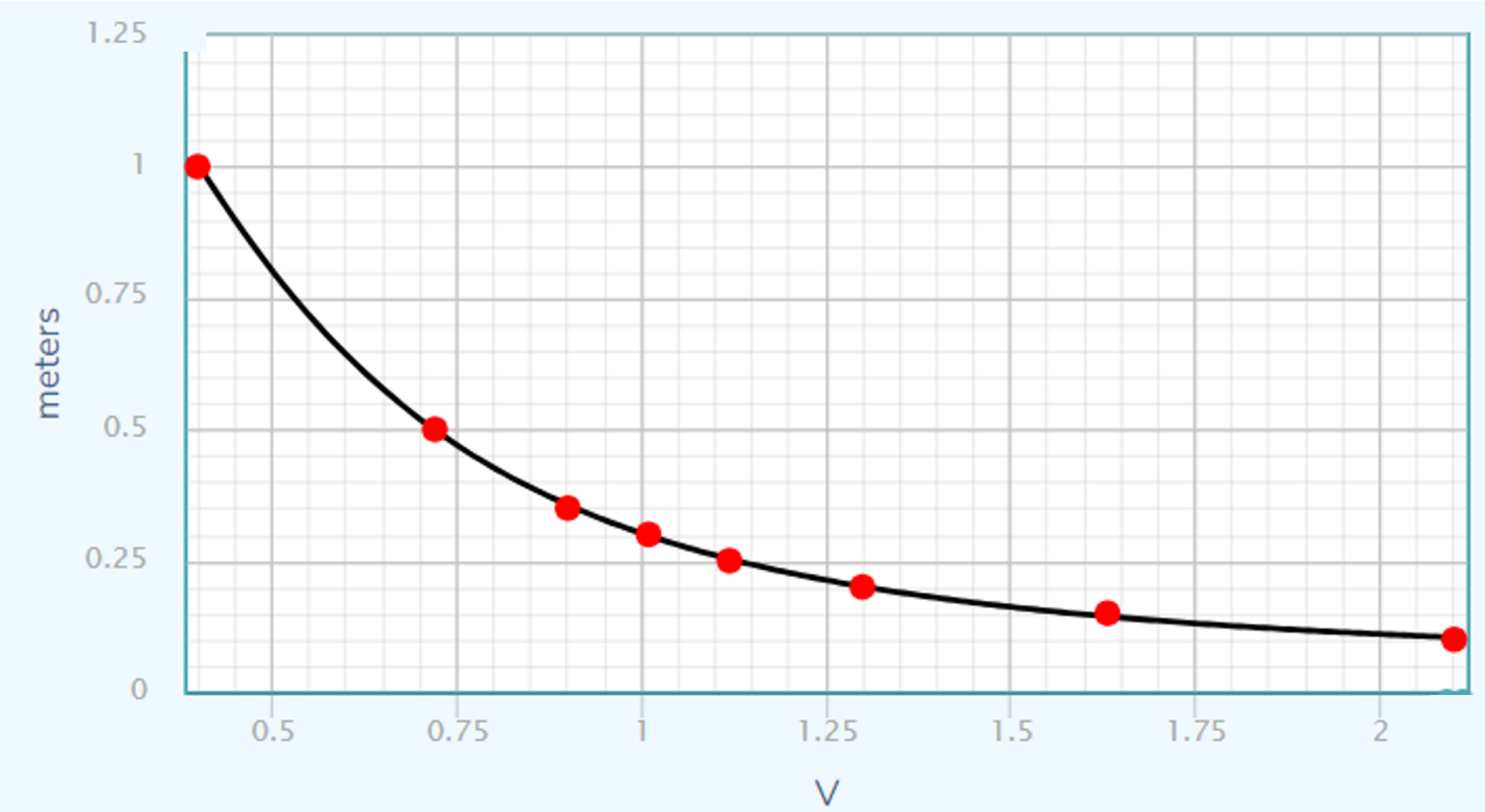
distance measurement

- Infra-red distance sensor has a valid range of 10 cm to 1.5 m.
- Output voltage (pin is inversely proportional to distance.
- The EN pin is required to be high for the device to work properly
- Measured voltage at 10 cm is (approximately) 2.1 V, at 1 m is 0.4 V.
- Sensor response is *nonlinear*.

between 0 and 1023: value in bits for voltage, need to be converted
careful about overflow and underflow

IR Distance Sensor Response

It can be approximated well with a *fourth order polynomial*

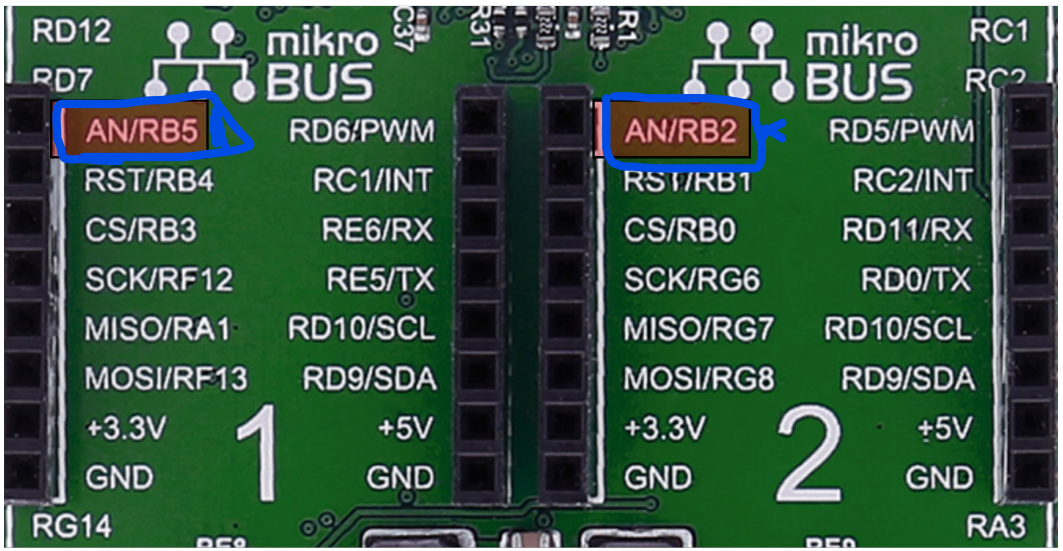


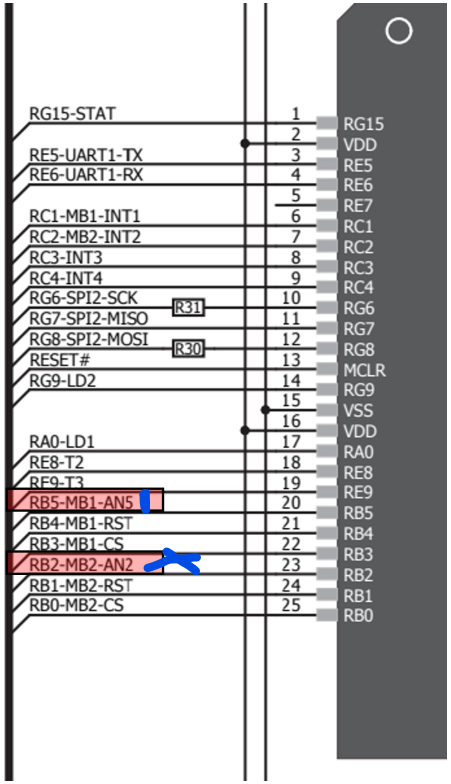
Given a voltage reading V, the distance measured by the sensor can be computed as:

$$y = 2.34 - 4.74V + 4.06V^2 - 1.60V^3 + 0.24V^4$$

Analog Pins on dspic33

The MIKROobus sockets on the Clicker 2 board are connected to RB5 and RB2, corresponding to AN5 and AN2 on the dspic33 layout.





via this we also determine which enable we need to activate

Assignment: ADC

to check if it works see the stream of bits from uart???

1. Plug the *Slider* and the RS232 UART shield in the MIKROBus sockets of the Clicker board. Read the value of the potentiometer and send it on the UART. Use manual sampling and manual conversion.
2. Plug the *IR Distance Sensor* and the RS232 UART shield in the MIKROBus sockets of the Clicker board. Read the value of the distance sensor and send it on the UART. Use manual sampling and automatic conversion. distance sensor will be used for final project
3. Plug the *IR Distance Sensor* and the RS232 UART shield in the MIKROBus sockets of the Clicker board. Use the interrupt of button E8 to trigger an ISR that performs manual sampling and conversion, then send the result on the UART. Hint: check the Interrupt and UART tutorials for the Clicker 2 board.