AURIX Analog-to-Digital Converters

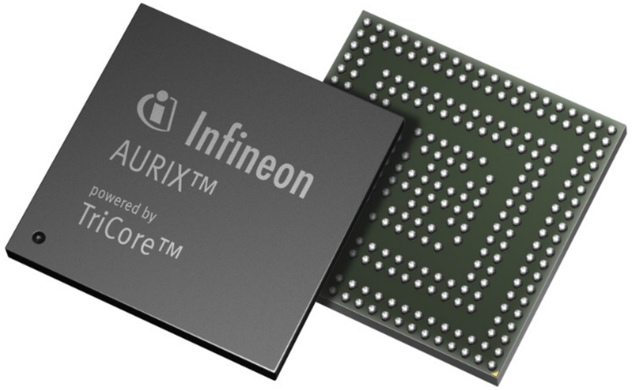
Agenda

Introduction to ADC

ADC Theory
- Sampling
- Resolution
- Nonlinearities
- Dynamic range
- SAR ADC example

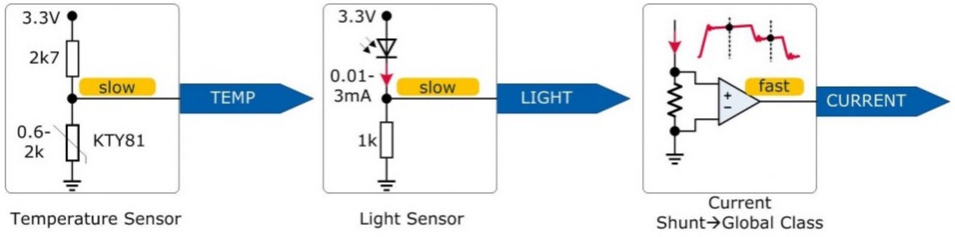
AURIX ADC
- AURIX ADC Modules
- EVADC Groups & Channels
- EVADC Conversion Queue
- EVADC Initialisation Sequence

Hands-on session



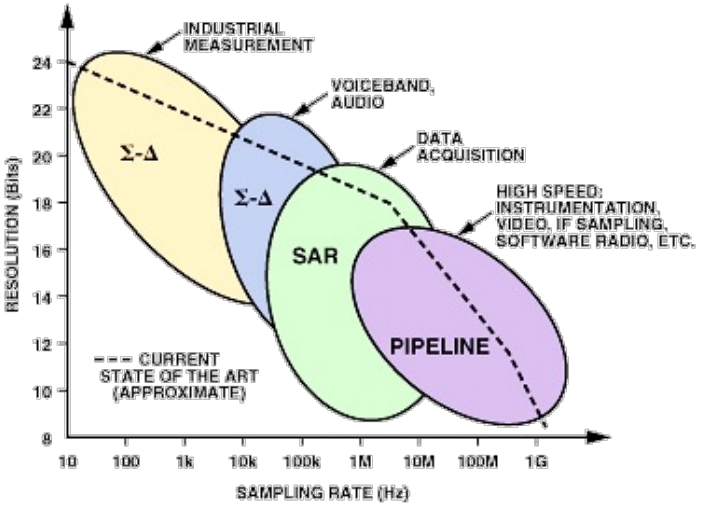
Introduction to ADC

ADC is a circuit that converts an analog signal (voltage variation over time) into discrete form (digital values) that can be processed by HW or SW



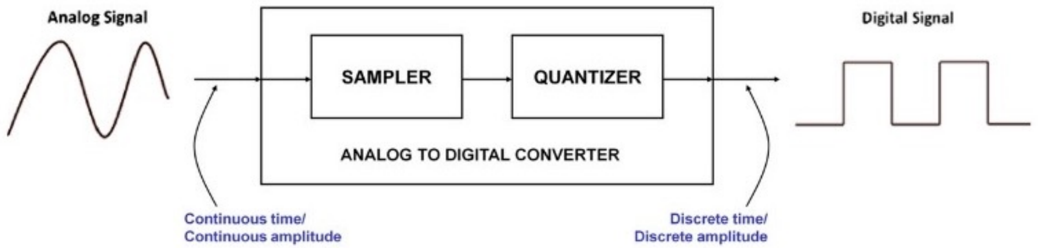
<u>Types of ADC</u>:

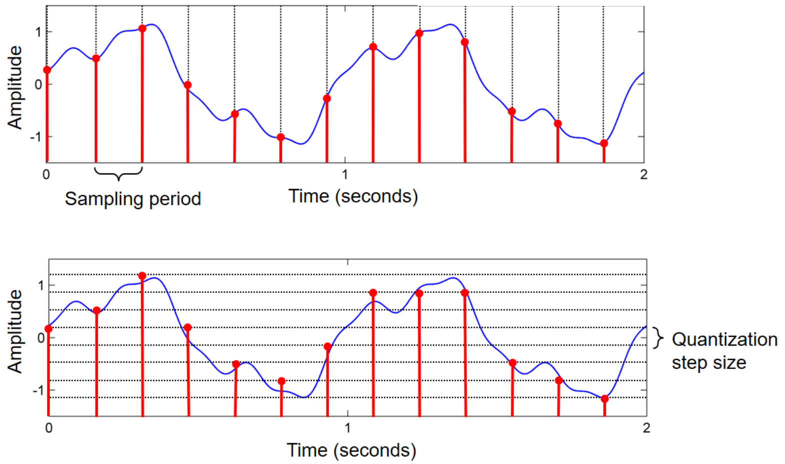
- **SAR** – Successive Approximation Register ADC
- **DSADC** – Delta-Sigma (Sigma-Delta) ADC
- FLASH-ADC
- Pipeline ADC
- Dual-Slope



ADC characteristics

Conversion pipeline



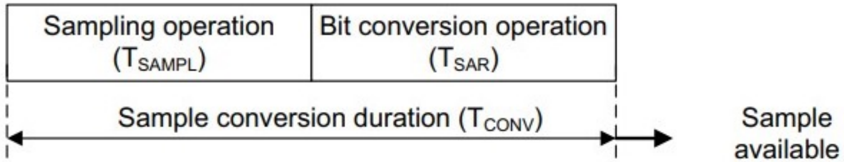


Key Characteristics

- *Sampling Rate*
- *Resolution*
- *DC & AC nonlinearities*
- *Dynamic range*
- *Supply level, consumption*
- *Input types*
- *Output format*

ADC Sampling

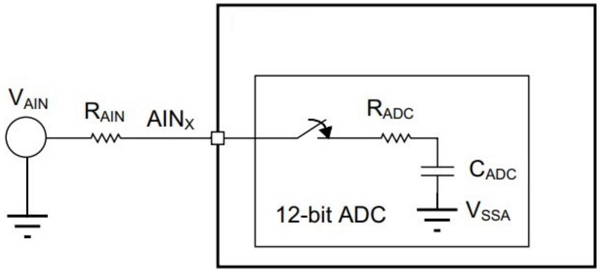
The **sample rate** or sampling frequency is the maximum rate at which an ADC can convert the analog signal into a digital data

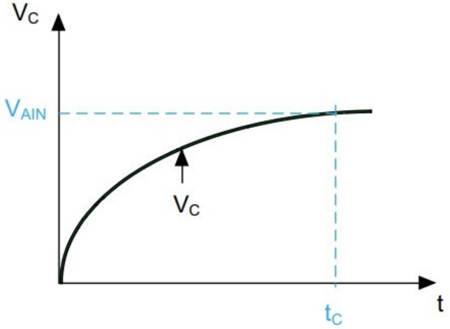


ADC sample conversion time $(T_{CONV}) = $ Sampling time $(T_{SMPL}) +$ Bit conversion time (T_{SAR})

ADC sample rate $= 1/T_{CONV} = 1/(T_{SMPL} + T_{SAR})$

The sampling interval is a part of the sampling-conversion period at which an ADC stores and holds (SH) an instantaneous value of the input signal

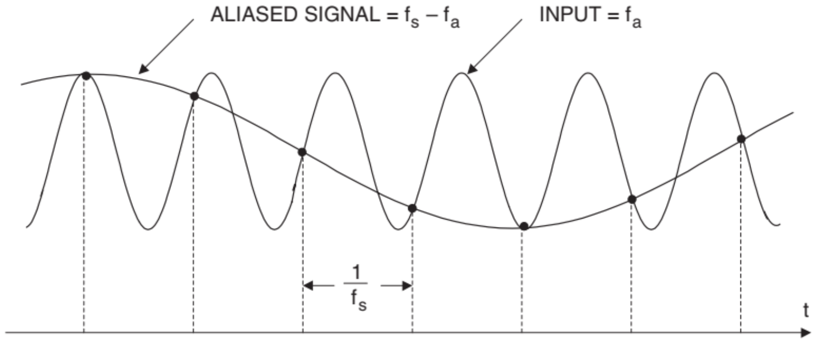




The selection of the Sampling rate depends on the input signal's highest frequency component ($ƒa$) and is defined by the **Nyquist frequency** ($ƒs$):

$$fs \geq fa * 2$$

Understanding the input signals properties e.g. highest frequency content is an important part of getting accurate measurements and avoiding Aliasing



ADC Resolution

The resolution determines the minimum change in the input signal that makes the output change by one count

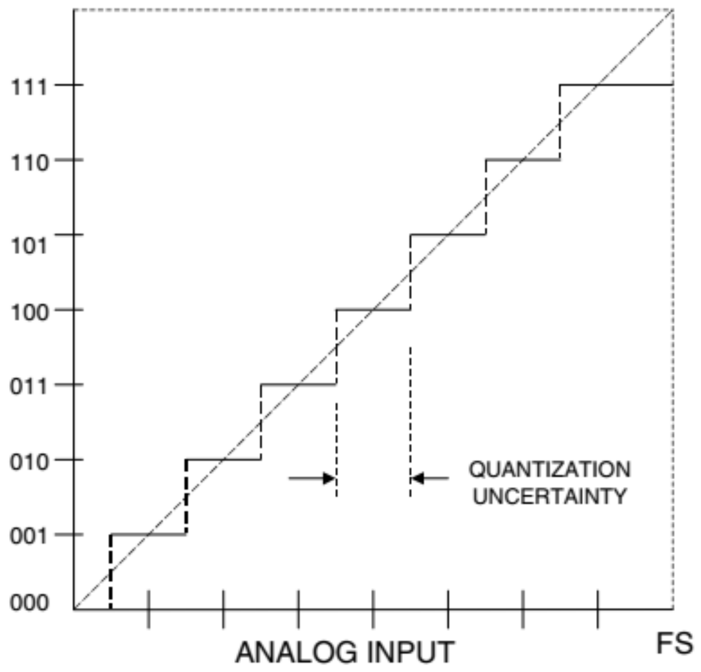
The resolution is expressed as a number of output bits. The smallest increment in the signal value that can be recognized by an ADC is defined as *least significant bit* (LSB):

$$1LSB = \frac{Vref}{(2^N - 1)}$$

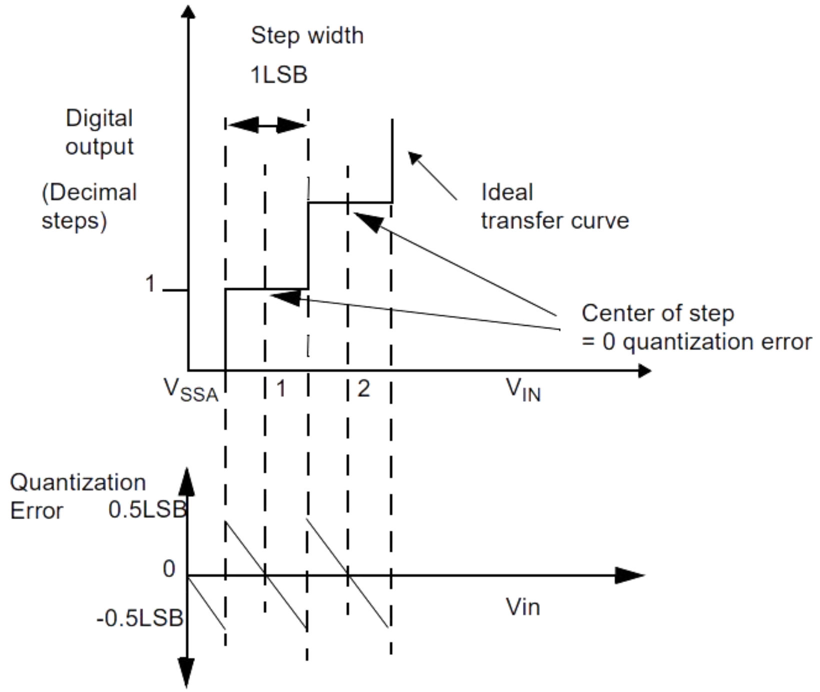
Resolution, N	2^N	LSB
8bit	255	10 mV
10bit	1024	5 mV
12bit	4096	1.22 mV
14bit	16384	0.3 mV
16bit	65536	0.075 mV

Vref = 5V

Ideal transfer function

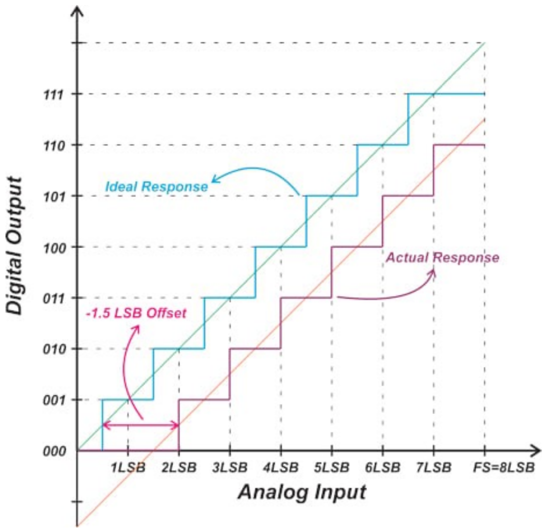


Quantization error

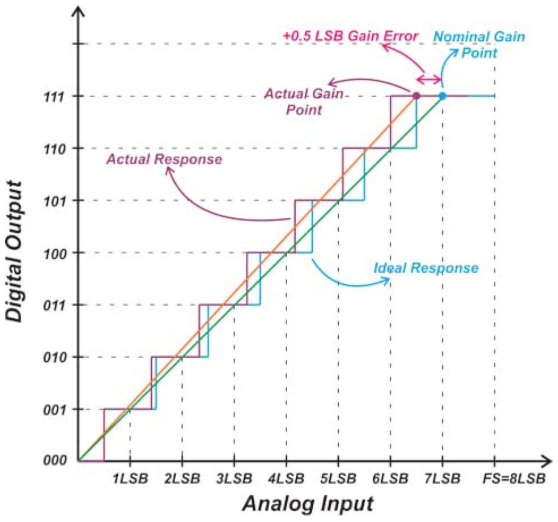


ADC Nonlinearities

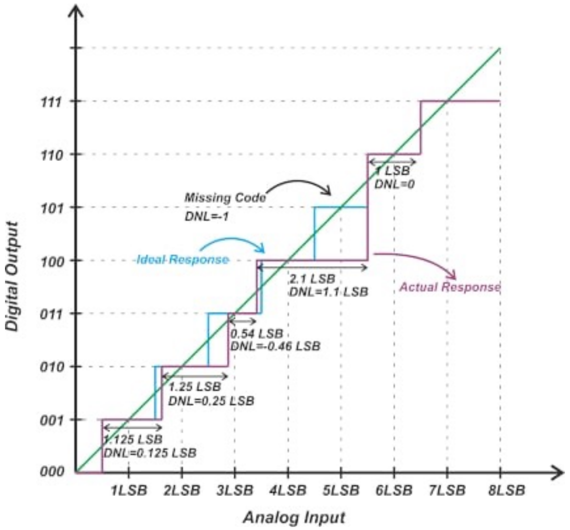
Transfer function nonlinearities in a real ADC



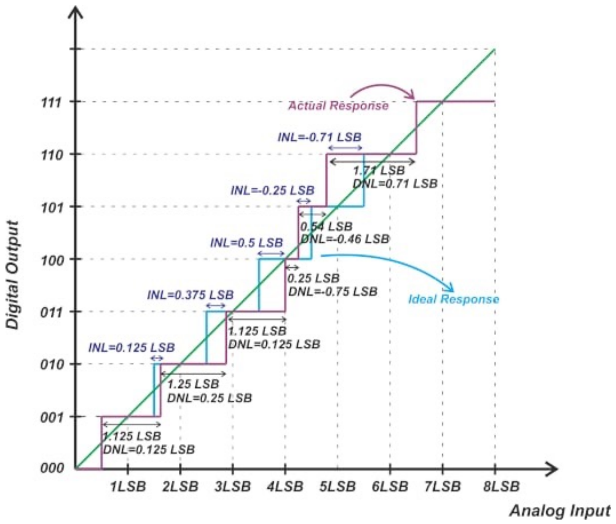
Offset



Gain



DNL



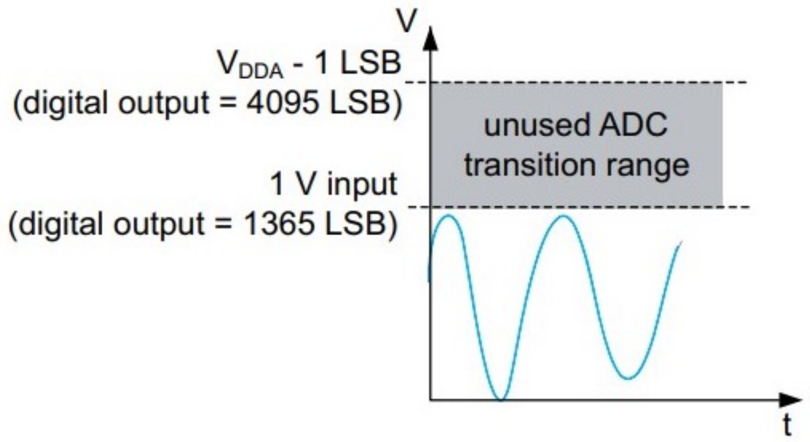
INL

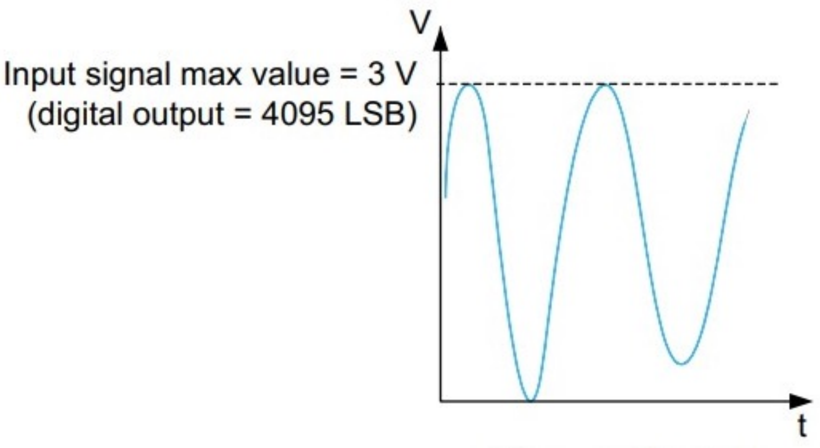
Total Unadjusted Error (**TUE**) defines the maximum deviation (in LSBs) from the ideal transfer curve

$$TUE = \sqrt{Offset^2 + Gain^2 + DNL^2 + INL^2}$$

ADC Dynamic range

Dynamic range defines the ratio between the minimum and the maximum input values that an ADC can reliably convert





The achieve maximum conversion precision the input signal has to match to the dynamic range of an ADC. The signal amplification or attenuation might be required

Data Conversion

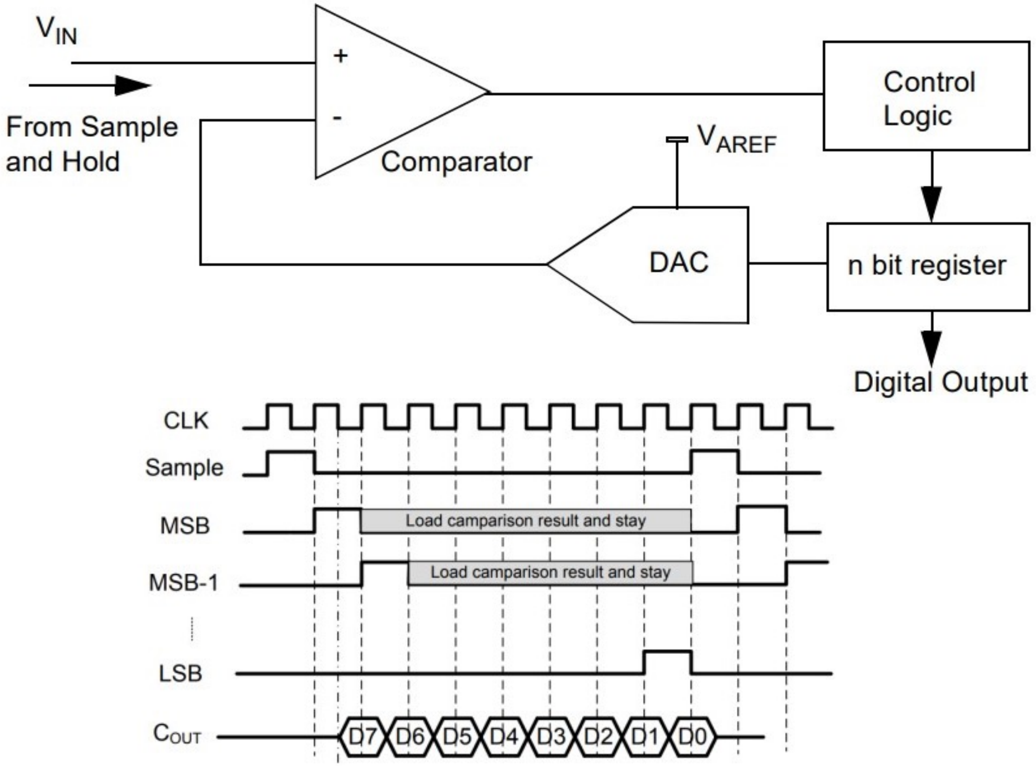
$$\text{Digital\_value} = \frac{\text{Vin}}{\text{Vref}} \times (2^N - 1)$$

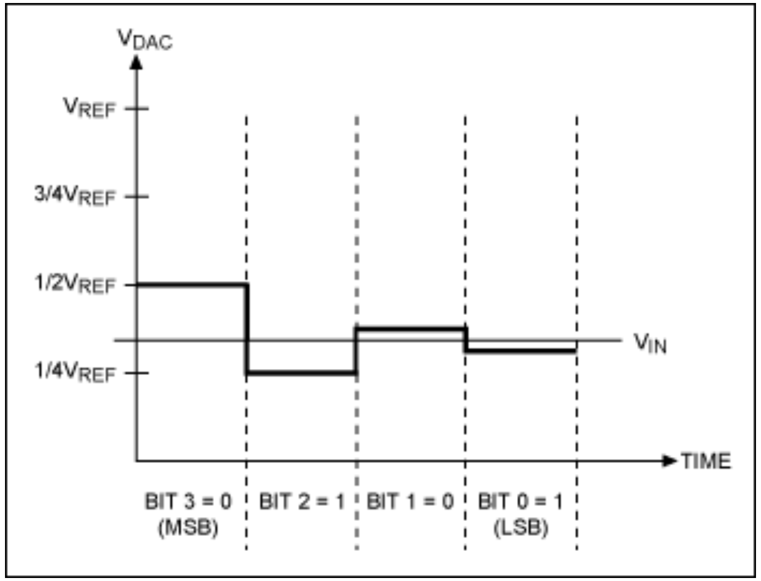
$$\text{Vin} = \frac{\text{Digital\_value}}{(2^N - 1)} \times \text{Vref}$$

SAR ADC

This sampled input from Sample & Hold (SH) capacitor is fed into a comparator along with the input from an internal DAC, the output of which is adjusted in binary increments to get as close as possible to the sampled value

SAR ADC employs a binary search algorithm to match an input voltage with a reference value





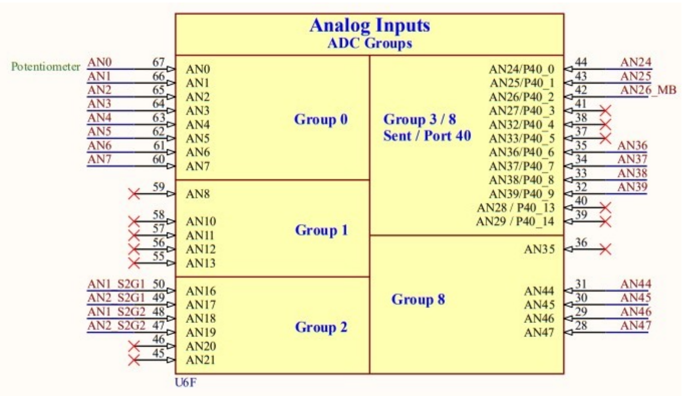
4-bit conversion example

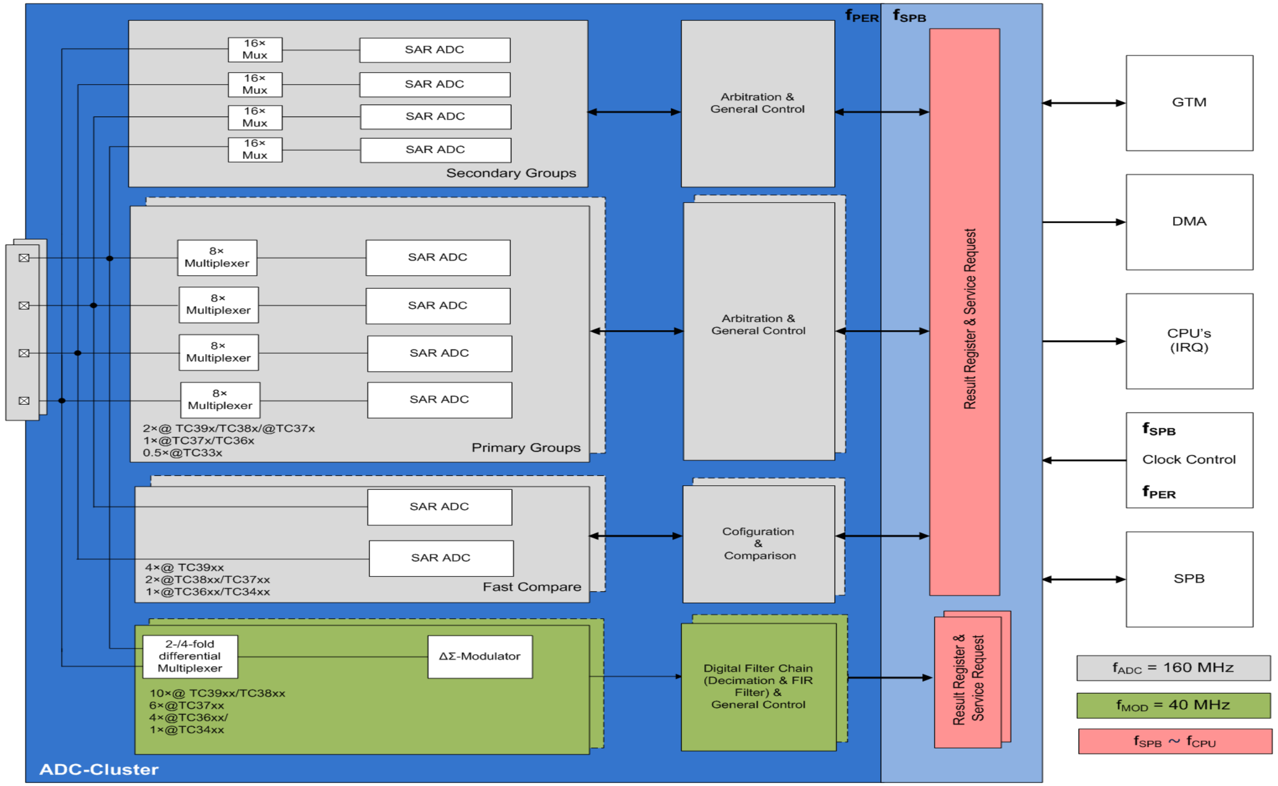
AURIX TC3x ADC

Three ADC types in AURIX

- **SAR** x 12 by 8/16 channels
  - **EVADC** Primary 12 bit, ≤ 2.5MS/s
  - **EVADC** Secondary 12 bit, ≤ 1.4MS/s
- **Fast Compare** x8 10 bit, ≤ 5MS/s
- **EDSADC** x14 16 bit, ≤ 200KS/s

TC375: 8 x **EVADC**, 4 x **FCC**, 6 x **EDSADC**

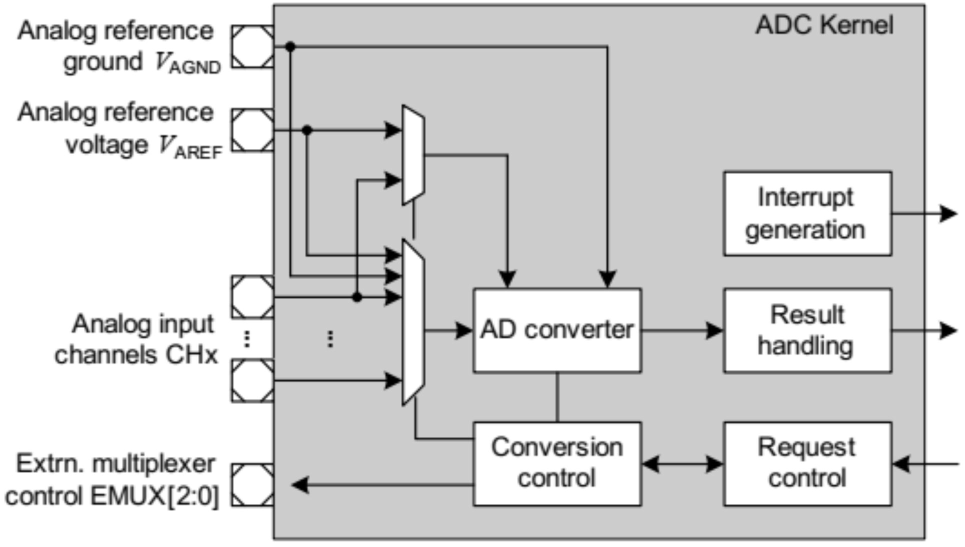




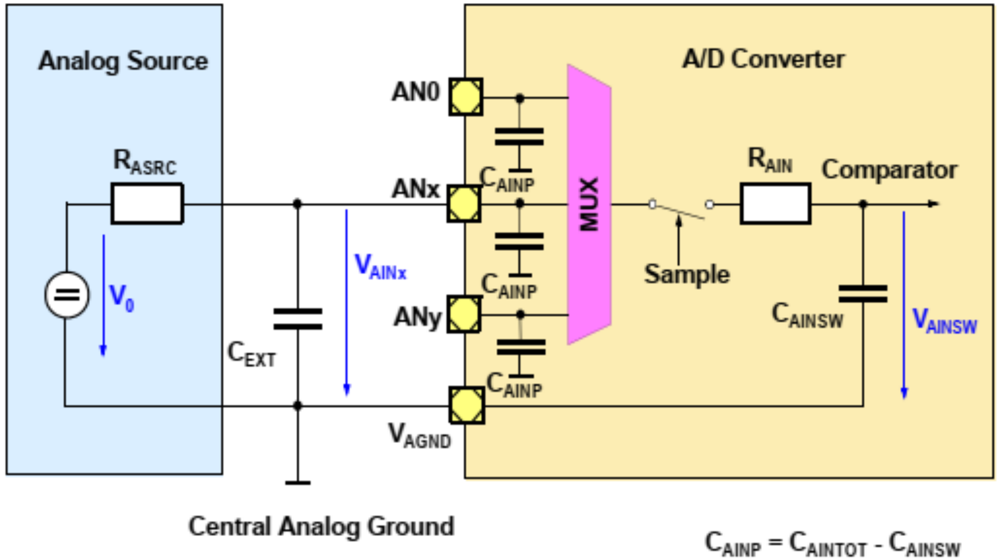
EVADC Group & Channels

Each **EVADC Group** is an independent SAR converter that consists of 8 (or 16) input channels, Multiplexer, Converter, Control Logic, Request control and Result handling

EVADC input channels are multiplexed to connect the corresponding signal source to the converter one at a time.
For each channel, the sample time can be controlled individually



MC_ADC_KERNEL



$C_{AINP} = C_{AINTOT} - C_{AINSW}$

EVADC Conversion

EVADC is designed to execute complex sequences of conversions by filling up *Queues*

Conversion Request

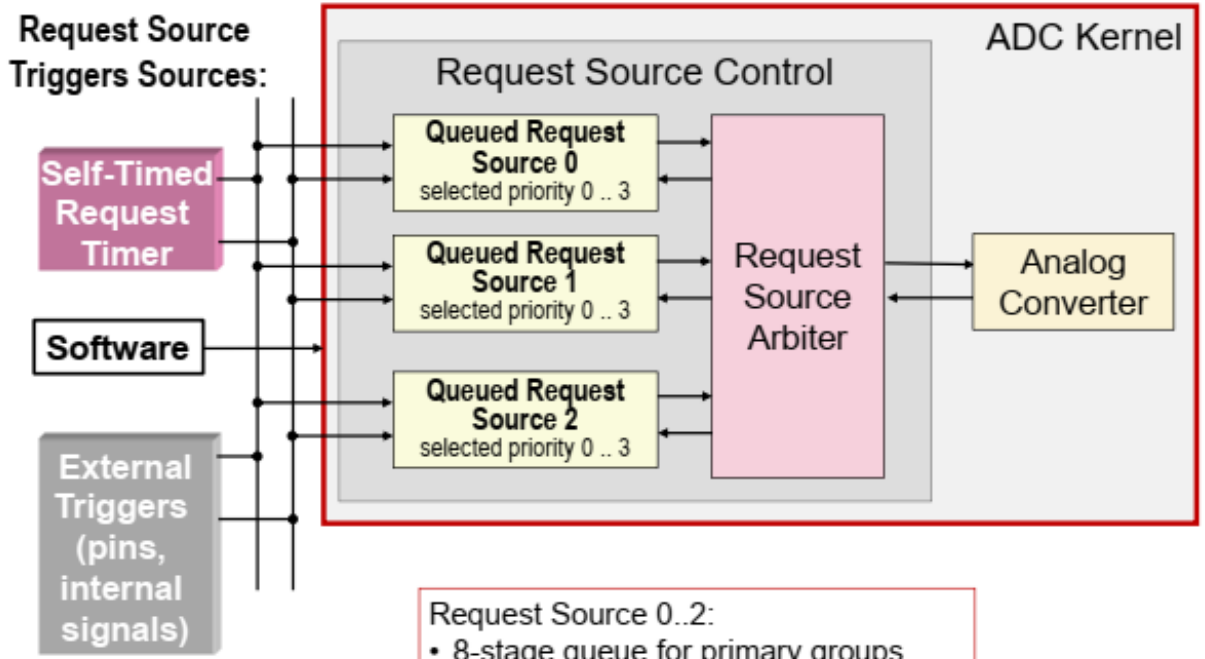
Conversion sequence can be started (requested) by 3 different sources:

- Software Trigger
- Self-Timed Trigger,
- External Trigger e.g. GPIO, GTM

The requested conversion can be executed once or repeatedly after trigger

Arbiter

When multiple conversion requests are used arbitration process defines which conversion is executed next based on assigned priorities



Request Source 0..2:
- 8-stage queue for primary groups
- 16-stage queue for secondary groups

EVADC initialization

Enable a primary/secondary group and prepare it for operation

```
EVADC_GxANCFG  = 0x00300000   ;Analog clock frequency is 160 MHz / 4 = 40 MHz (example)
                              ;CALSTC = 00
EVADC_GxARBCFG= 0x00000003    ;Enable analog block
WAIT                          ;Pause for extended wakeup time (≥ 5 µs)
EVADC_GLOBCFG  = 0x80000000   ;Begin start-up calibration
                              ;(other operations can be executed in the meantime)
EVADC_GxARBPR  = 0x01000000   ;Enable arbitration slot 0
EVADC_GxQMR0   = 0x00000001   ;Enable request source 0
EVADC_GxICLASS0=0x00000002    ;Select 4 clocks for sampletime 4 / 40 MHz = 100 ns
                              ;The default setting stores results in GxRES0,
                              ;service requests are issued on GxSR0
EVADC_GxRCR0   = 0x80000000   ;Enable result service requests, if required
EVADC_GxQINR0  = 0x00000020   ;Request channel 0 in auto-repeat mode
WAIT                          ;Wait for start-up calibration to complete
                              ;(other operations can be executed in the meantime)
                              ;---> This starts continuous conversion of the channel
```

EVADC basic setup sequence

Clk & Analog Block enable

↓

Request source and queue settings

↓

Sample time settings

↓

Start conversion

↓

Wait for the 1st conversion result

Hands-on session

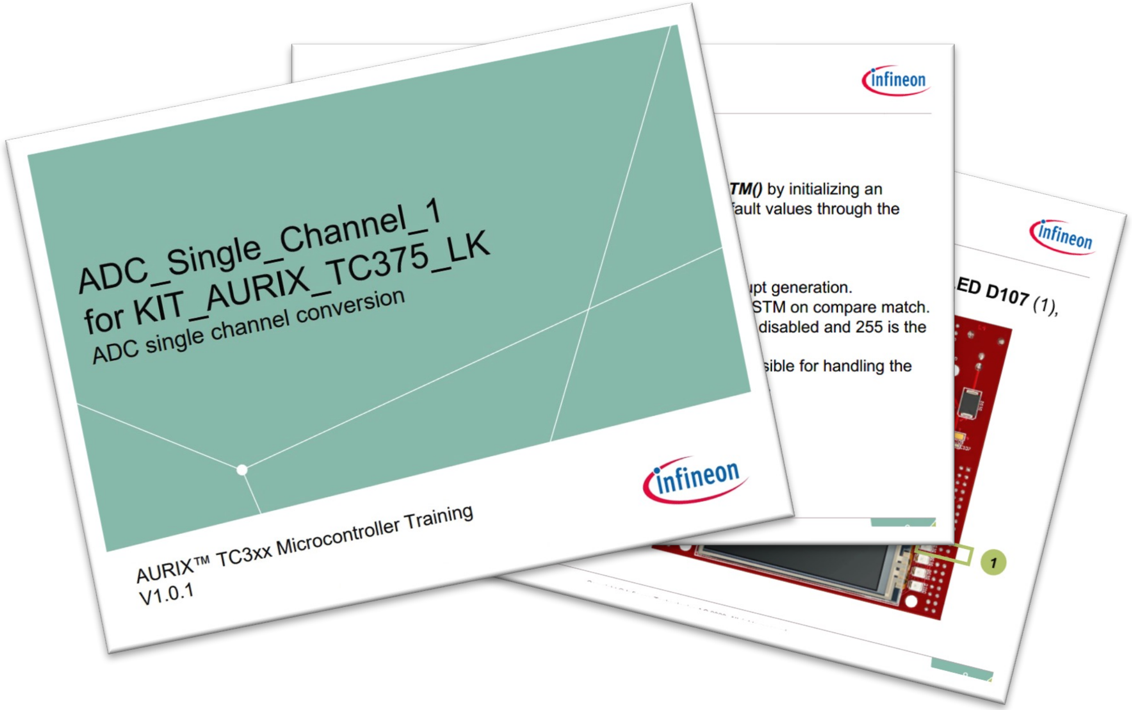
ADC_Single_Channel_1 for KIT_AURIX_TC375_LK

[Code Example]

STM_Interrupt.c

```c
/* Macro to define Interrupt Service Routine.
 * This macro makes following definitions:
 * 1) Define linker section as .intvec_tc<vector number>_<interrupt priority>.
 * 2) define compiler specific attribute for the interrupt functions.
 * 3) define the Interrupt service routine as ISR function.
 *
 * IFX_INTERRUPT(isr, vectabNum, priority)
 *  - isr: Name of the ISR function.
 *  - vectabNum: Vector table number.
 *  - priority: Interrupt priority. Refer Usage of Interrupt Macro for more details.
 */
IFX_INTERRUPT(isrSTM, 0, ISR_PRIORITY_STM);

void isrSTM(void)
{
    /* Update the compare register value that will trigger the next interrupt and toggle the LED */
    IfxStm_increaseCompare(STM, g_STMConf.comparator, TIMER_INT_TIME);
    IfxPort_setPinState(LED, IfxPort_State_toggled);
}

/* Function to initialize the LED */
void initLED(void)
{
    IfxPort_setPinMode(LED, IfxPort_Mode_outputPushPullGeneral);    /* Initialize LED port pin                      */
    IfxPort_setPinState(LED, IfxPort_State_high);                   /* Turn off LED (LED is low-level active)       */
}

/* Function to initialize the STM */
void initSTM(void)
{
    IfxStm_initCompareConfig(&g_STMConf);           /* Initialize the configuration structure with default values   */

    g_STMConf.triggerPriority = ISR_PRIORITY_STM;   /* Set the priority of the interrupt                            */
    g_STMConf.typeOfService = IfxSrc_Tos_cpu0;      /* Set the service provider for the interrupts                  */
    g_STMConf.ticks = TIMER_INT_TIME;               /* Set the number of ticks after which the timer triggers an
                                                     * interrupt for the first time                                 */
    IfxStm_initCompare(STM, &g_STMConf);            /* Initialize the STM with the user configuration               */
}

/* Function to initialize all the peripherals and variables used */
void initPeripherals(void)
{
    initTime();                                     /* Initialize time constants                                    */
    initLED();                                      /* Initialize the port pin to which the LED is connected        */
    initSTM();                                      /* Configure the STM module                                     */
}
```

[Tutorial]



Resources

Books

Walt Kester. 2003. Mixed Signal and DSP Design Techniques, Analog Devices, ISBN 978-0-7506-7611-3

Thomas C. Hayes. 2016. Learning the Art of Electronics: A Hands-On Lab Course 1st Edition, Cambridge U. Press, ISBN 978-0-521-17723-8

Application Notes

AP56003 A Guide to the Analog Part of the A/D Converter

AP32297 A/D Converter Supply and PCB Design Guideline

Code examples & Tutorials

https://github.com/Infineon/AURIX_code_examples/blob/master/code_examples

https://www.infineon.com/cms/en/product/promopages/aurix-expert-training/

Part of your life. Part of tomorrow.