Analog to Digital Conversion

Lecture 8

In These Notes . . .

Analog to Digital Converters

- ADC architectures
- Sampling/Aliasing
- Quantization
- Inputs
- M30262 ADC Peripheral

Reference: M30626 ADC: Hardware Manual, pp. 187-202

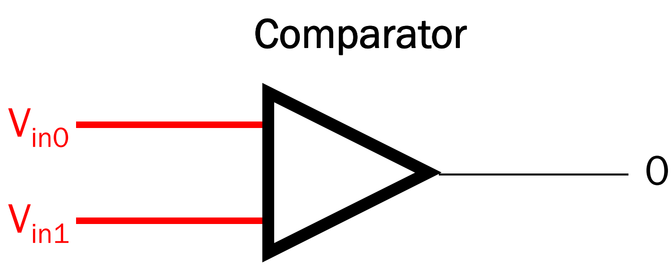
From Analog to Digital

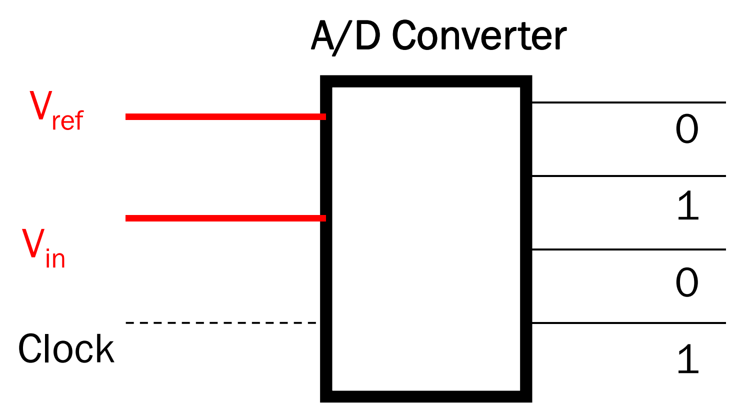
Embedded systems often need to measure values of physical parameters

These parameters are usually continuous (*analog*) and not in a digital form which computers (which operate on discrete data values) can process

A **Comparator** is a circuit which compares an analog input voltage with a reference voltage and determines which is larger, returning a 1-bit number

An **Analog to Digital converter** [AD or ADC] is a circuit which accepts an analog input signal (usually a voltage) and produces a corresponding multi-bit number at the output.





ADC Basic Functionality

n = converted code

V_{in} = sampled input voltage

V_{+ref} = upper end of input voltage range

V_{-ref} = lower end of input voltage range

N = number of bits of resolution in ADC

$$n = \left[\frac{(V_{in} - V_{-ref})(2^N - 1)}{V_{+ref} - V_{-ref}} + 1/2 \right]_{\text{int}}$$

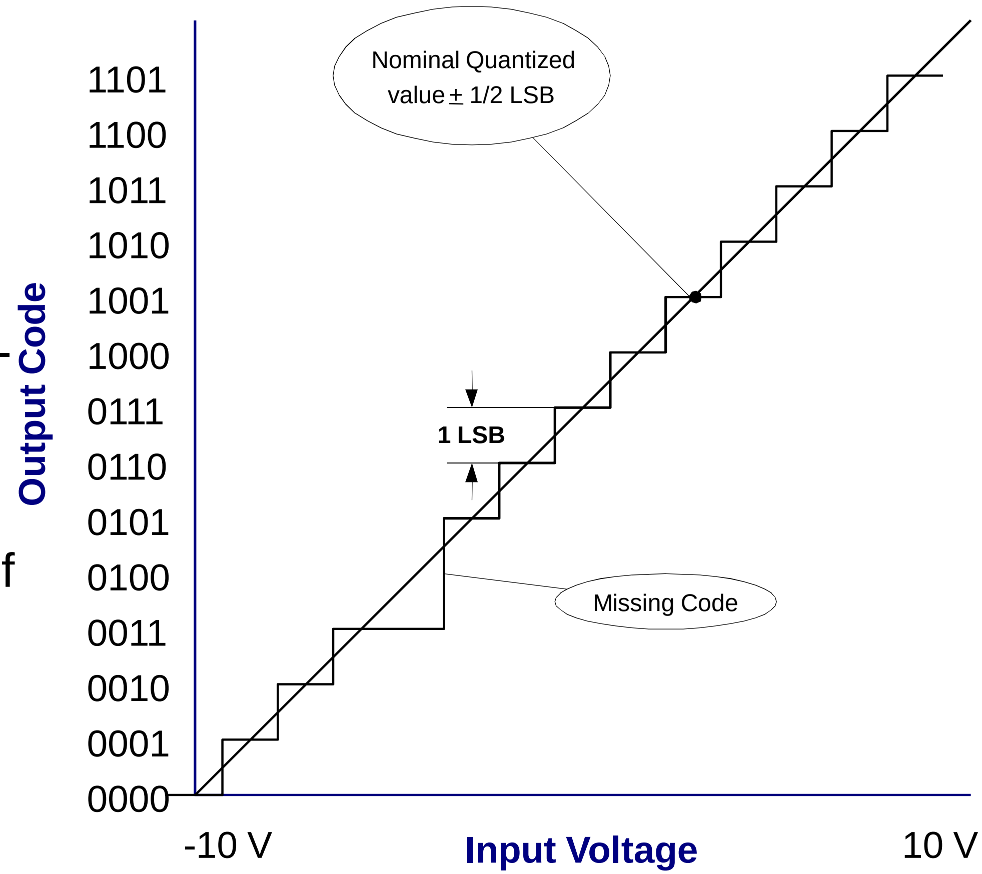
$$n = \left[\frac{(V_{in})(2^N - 1)}{V_{+ref}} + 1/2 \right]_{\text{int}} \qquad \text{if } V_{-ref} = 0\text{v}$$

$$n = \left[\frac{3.30v(2^{10} - 1)}{5v} + 1/2 \right]_{\text{int}} = 675$$

ADC Transfer Function

The ideal output from an A/D converter is a stair-step function (see right)

- Ideal worst case error in conversion is ± 1/2 bit.
- Missing codes or the imperfections where increasing voltage does not result in the next step being output are described as non-monotonicity.
- Errors in A/D conversion may be significant particularly if the full range of the analog signal is significantly less than the range of the analog input of the A/D.

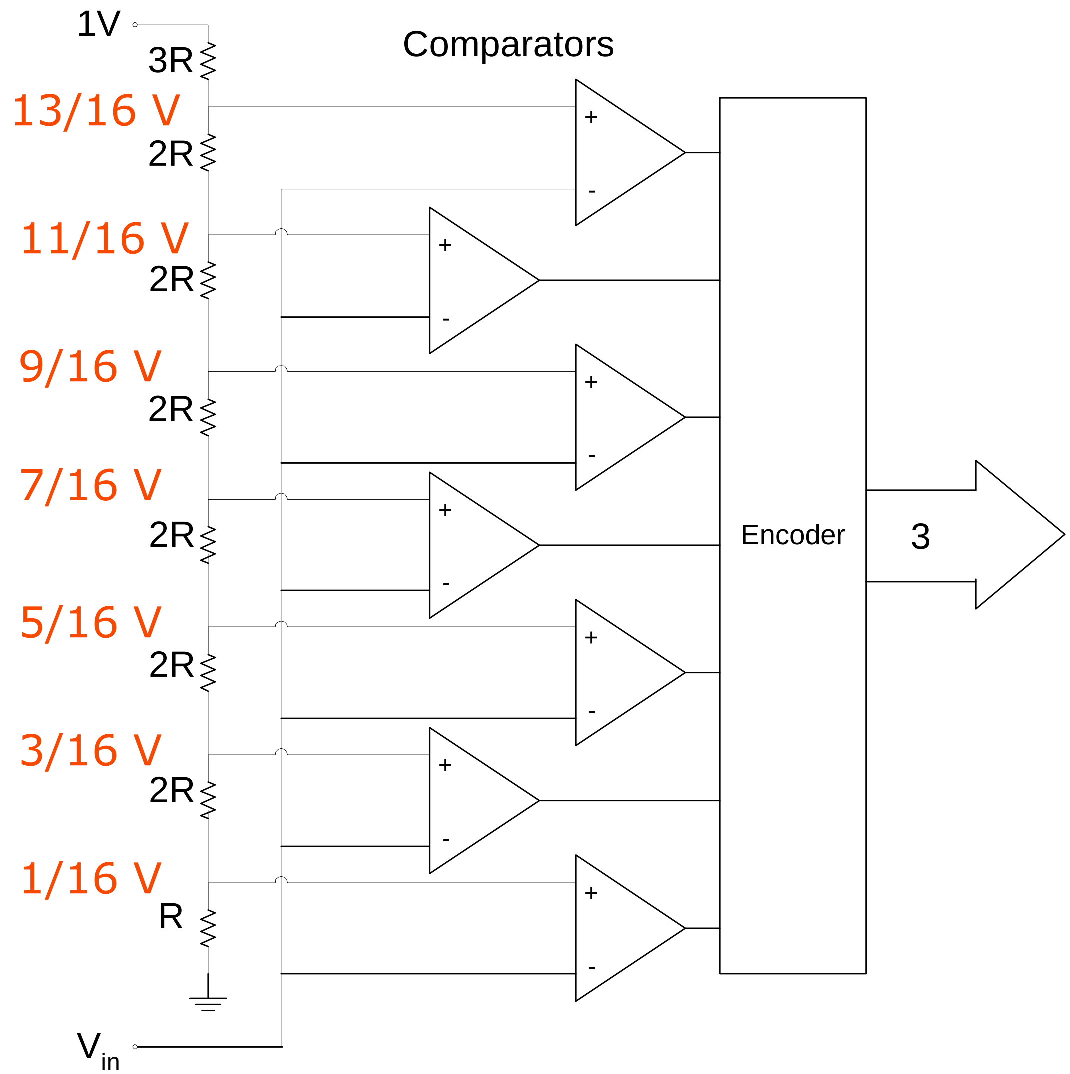


A/D – Flash Conversion

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 (an encoder)



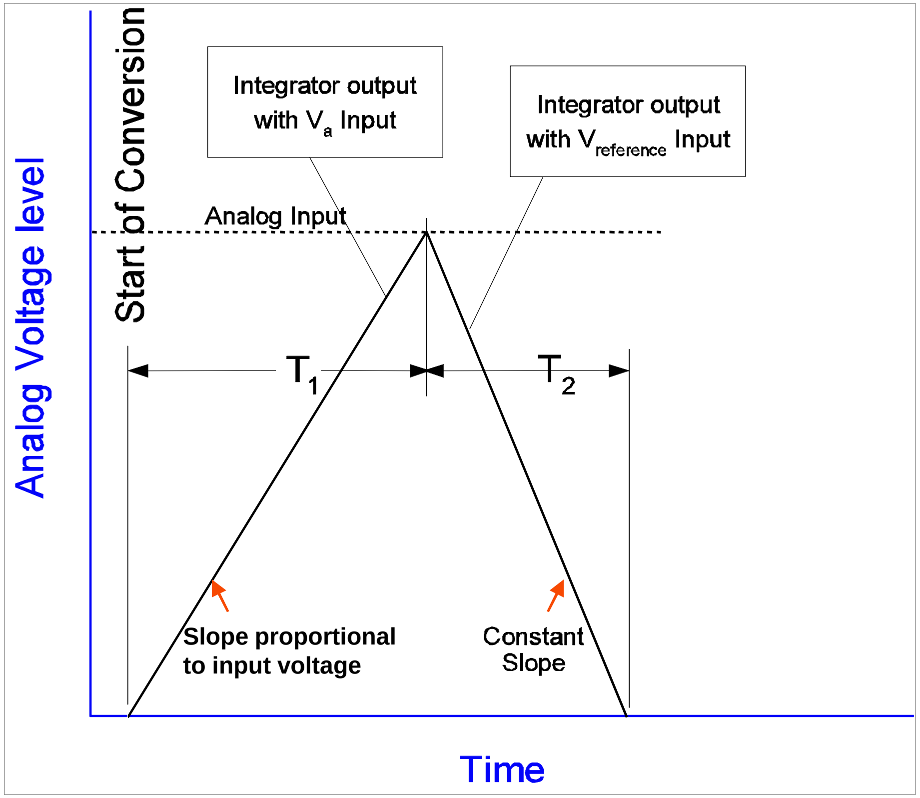
ADC - Dual Slope Integrating

Operation

- Input signal is integrated for a fixed time
- Input is switched to the negative reference and the negative reference is then integrated until the integrator output is zero
- The time required to integrate the signal back to zero is used to compute the value of the signal
- Accuracy dependent on V_{ref} and timing

Characteristics

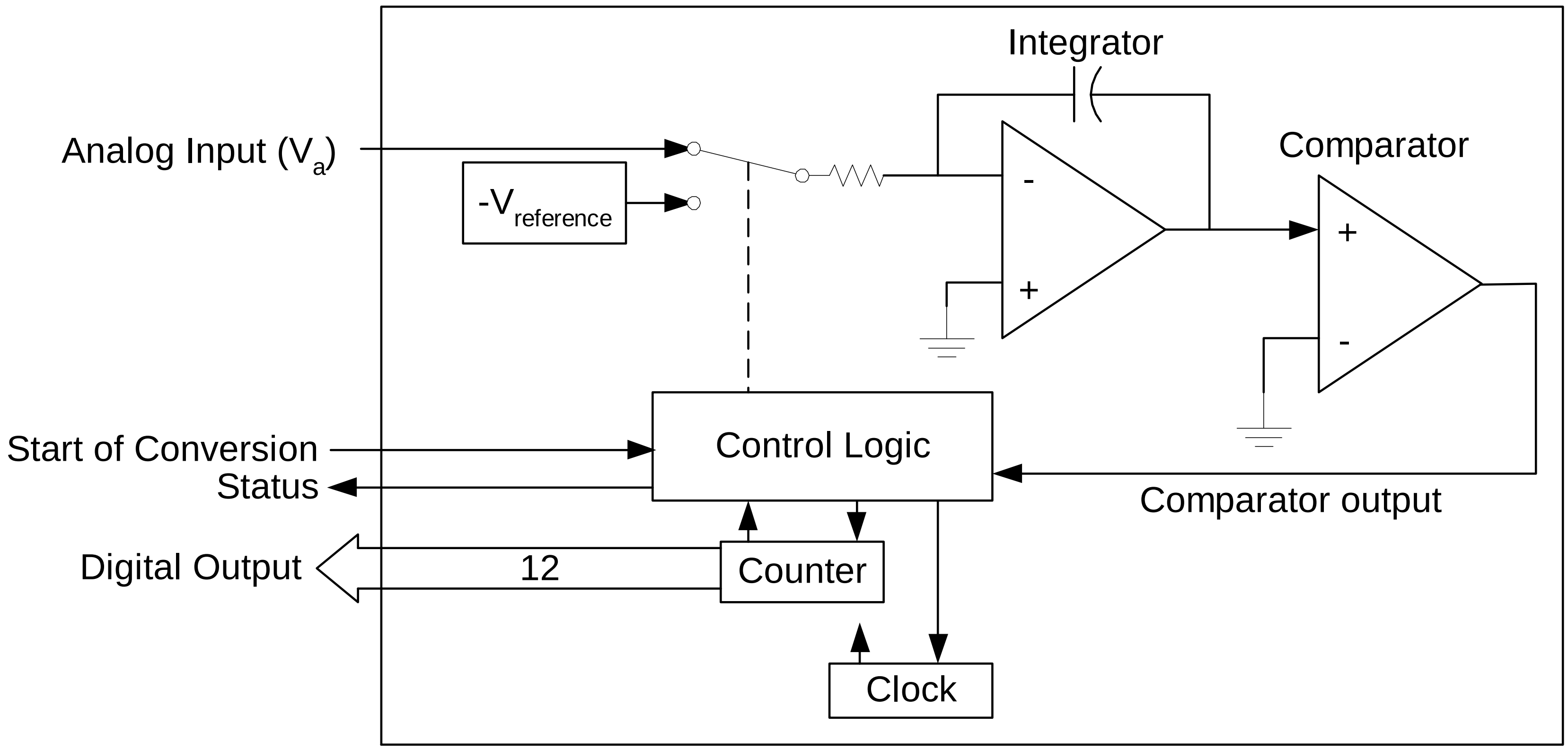
- Noise tolerant (Integrates variations in the input signal during the T_1 phase)
- Typically slow conversion rates (Hz to few kHz)



$$\frac{1}{C}\int_0^{T_1} V_{in}\,dt = -\frac{1}{C}\int_0^{T_2} V_{ref}\,dt$$

$$V_{in} = V_{ref}\frac{T_2}{T_1}$$

ADC - Dual Slope Integrating



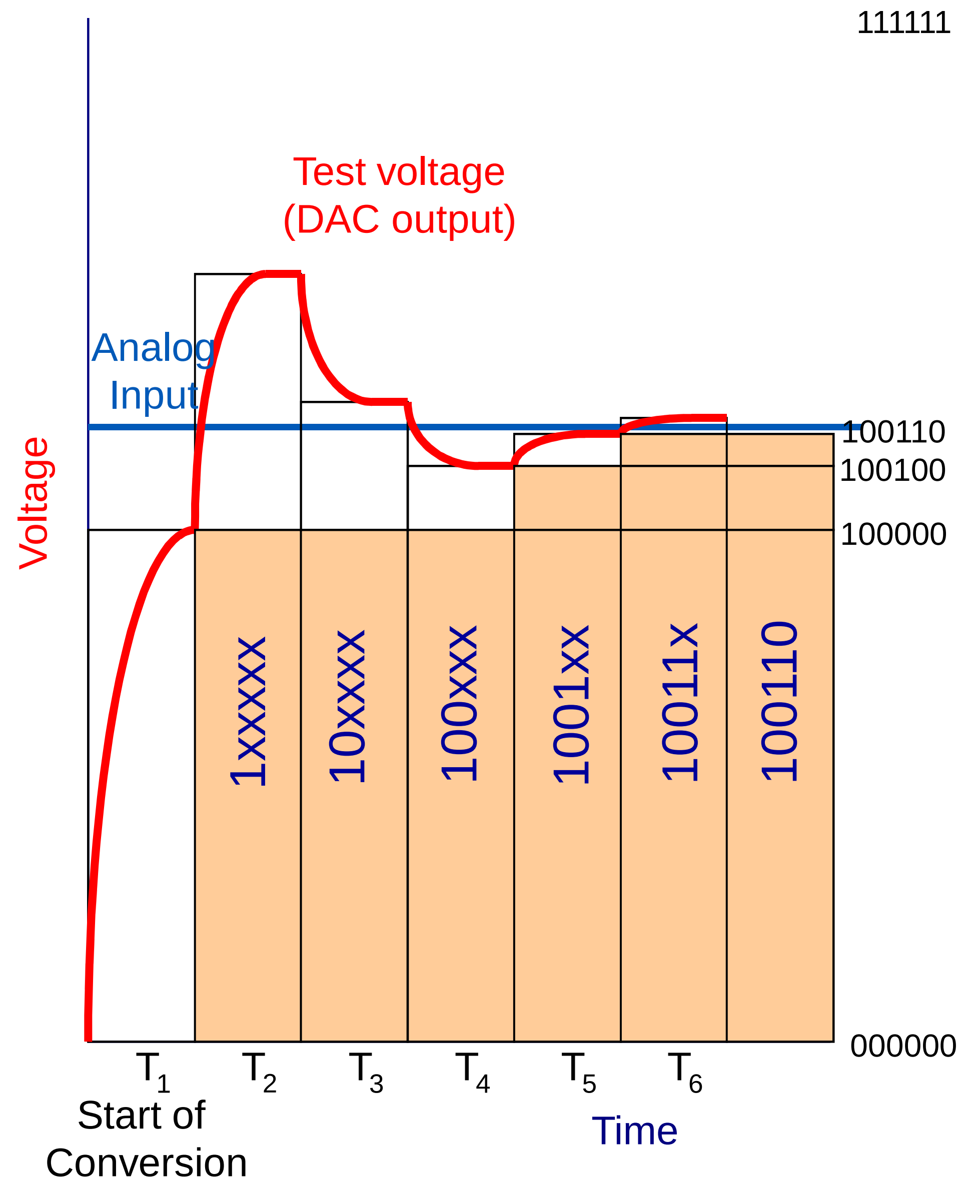
ADC - Successive Approximation Conversion

Successively approximate input voltage by using a binary search and a DAC

SA Register holds current approximation of result

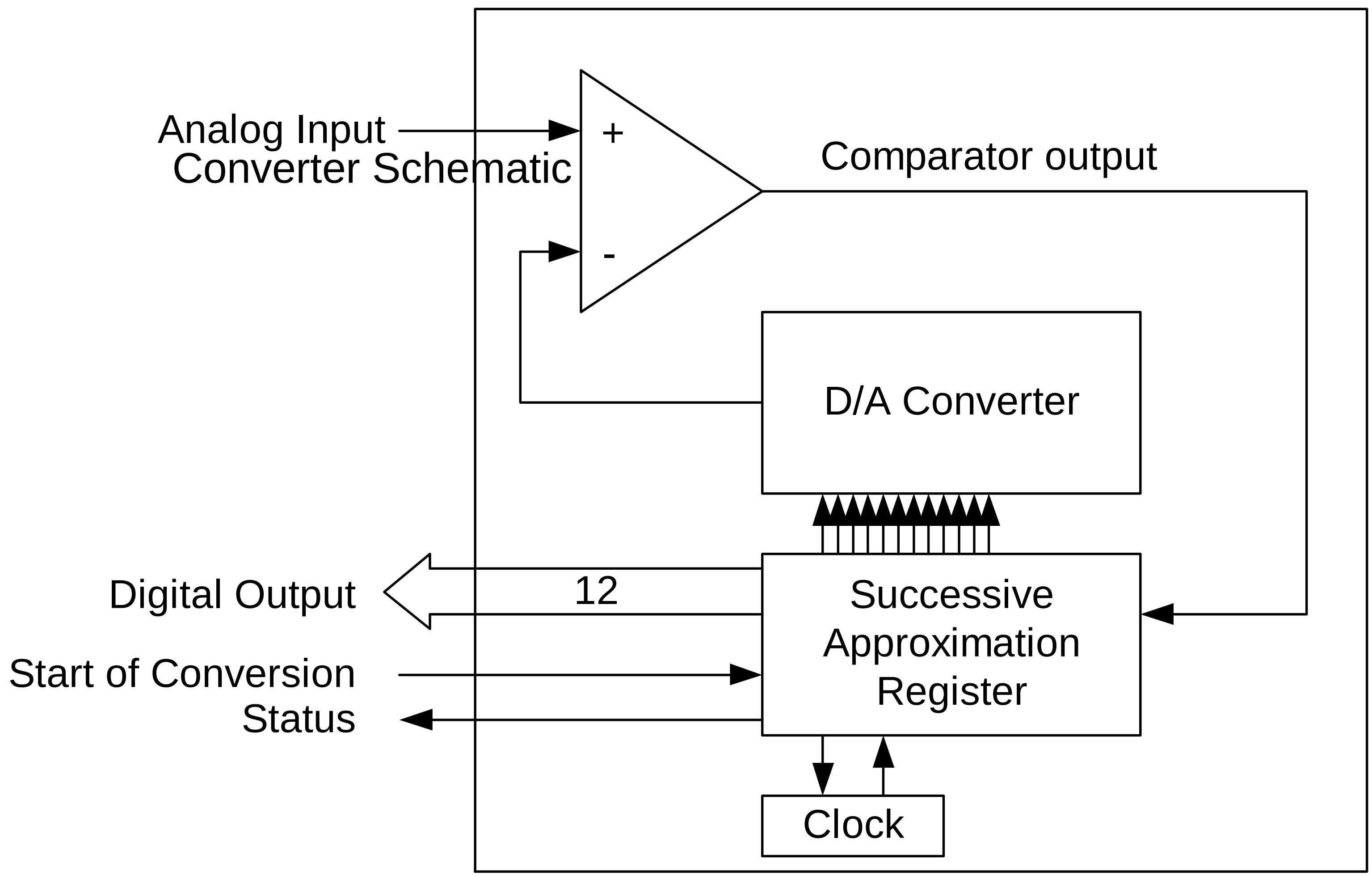
Repeat

- Set next bit input bit for DAC to 1
- Wait for DAC and comparator to stabilize
- If the DAC output (test voltage) is larger than the input then set the current bit to 1, else clear the current bit to 0



A/D - Successive Approximation

Converter Schematic



A/D - Sigma / Delta

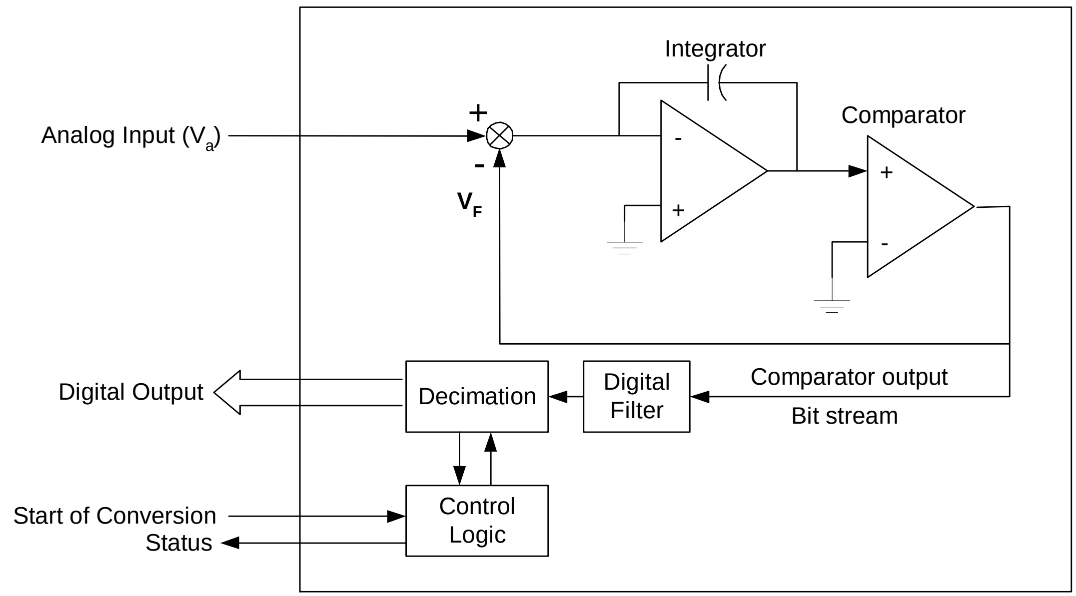
Operation

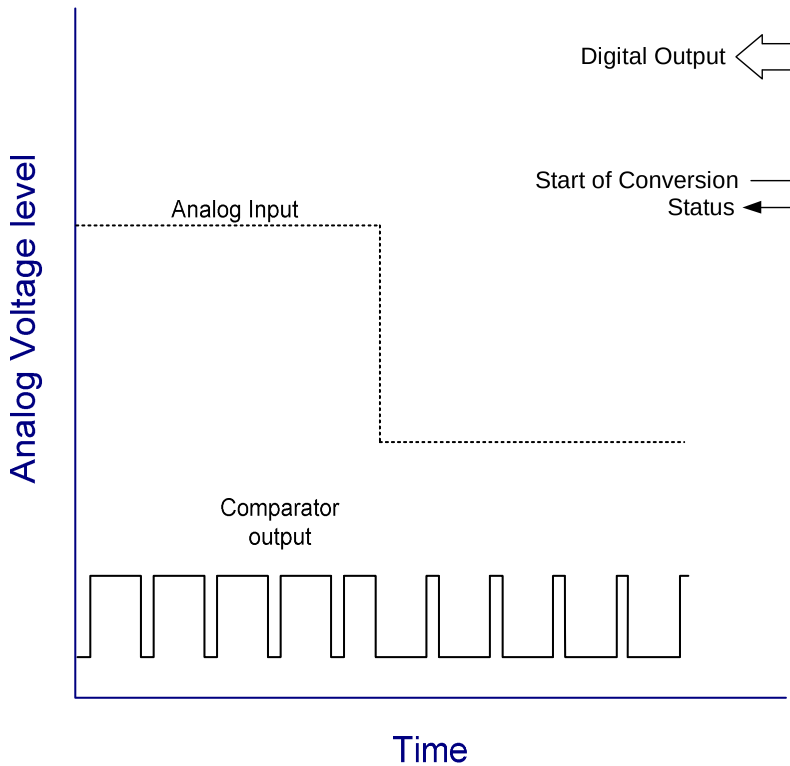
- Comparator feedback signal is subtracted from analog input and the difference is integrated.
- The average value of V_F is forced to equal V_a.
- V_F is a digital pulse stream whose duty cycle is proportional to V_a
- This pulse stream is sampled digitally and averaged numerically (decimation) giving a numerical representation of V_a
- The error in the average or mean is:

$$\sigma_{\bar{\mu}} = \frac{\sigma}{\sqrt{n}}$$

- The greater the number of samples averaged, the greater the accuracy
- The greater the number of samples averaged, the greater the time between the start of gathering samples and the output of the mean (group delay)
- This A/D does not work well if switched from channel to channel because of the delay until a valid result

Sigma / Delta


Integrator
Comparator
Analog Input (V_a)
+
-
V_F
+
-
+
-
Digital Output
Decimation
Digital Filter
Comparator output
Bit stream
Start of Conversion
Status
Control Logic


Analog Voltage level
Analog Input
Comparator output
Time

ADC Performance Metrics

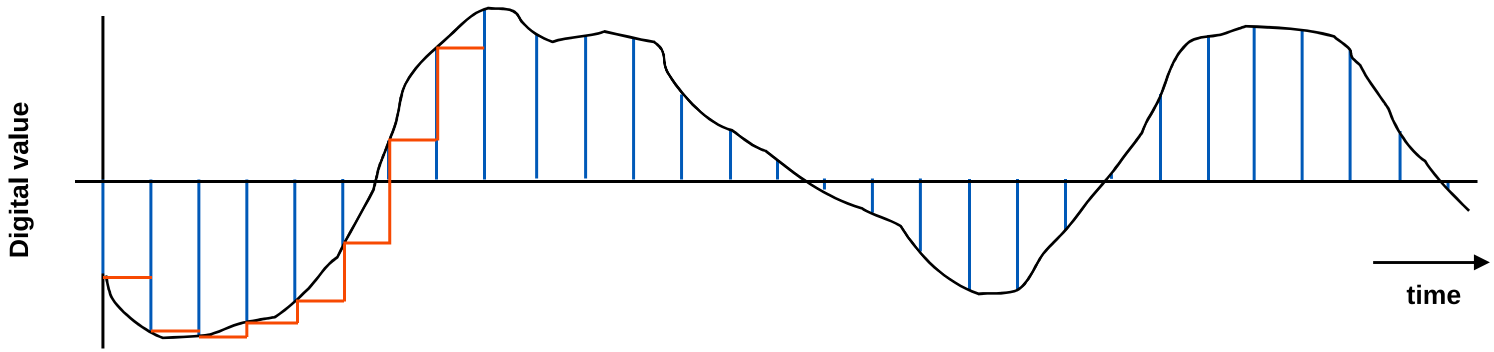
Linearity measures how well the transition voltages lie on a straight line.

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*

Waveform Sampling and Quantization



A waveform is **sampled** at a constant rate – every Δ_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."*

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

Quantization

Quantization: converting an analog value (infinite resolution or range) to a digital value of N bits(finite resolution, 2^N levels can be represented)

Quantization error

- Due to limited resolution of digital representation
- $<= 1/(2*2^N)$
- Acoustic impact can be minimized by dithering (adding noise to input signal)

16 bits…. too much for a generic microcontroller application?

- Consider a 0-5V analog signal to be quantized
- The LSB represents a change of 76 microvolts
- Unless you're very careful with your circuit design, you can expect noise of of at least tens of millivolts to be added in
- 10 mV noise = 131 quantization levels. So $\log_2 131 = 7.03$ bits of 16 are useless!

Inputs

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

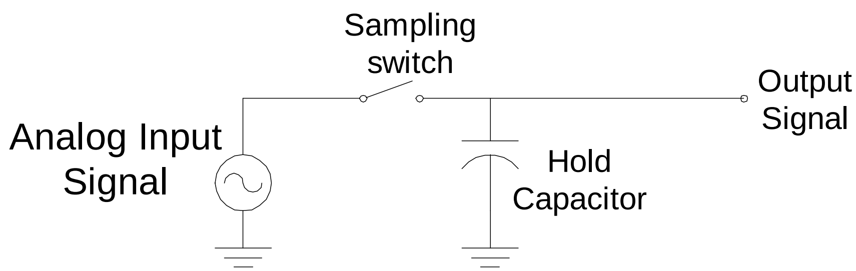
Sample and Hold Devices

Some A/D converters require the input analog signal to be held constant during conversion, (eg. 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 to the right:

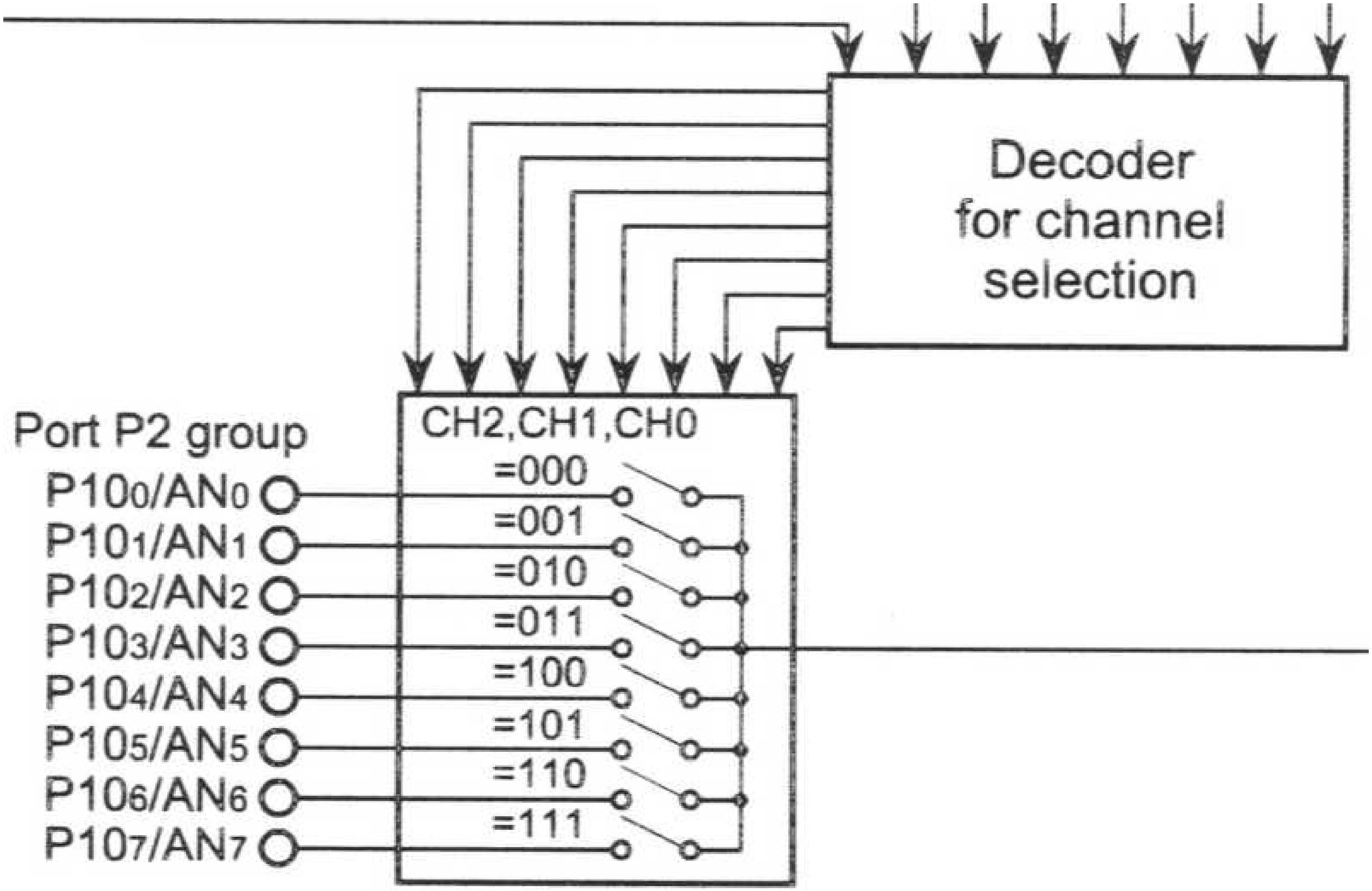
These devices are incorporated into some A/D converters



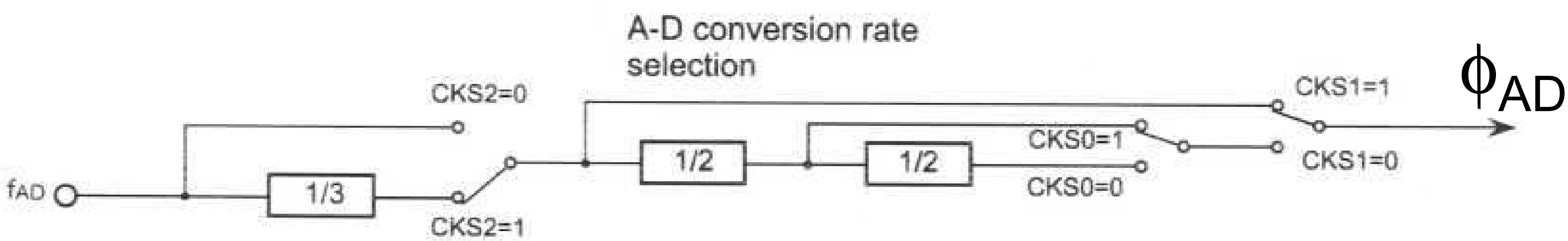
M30626P ADC Peripheral

- 10 bit successive approximation converter, can operate in 8 bit mode
- Input voltage: 0 to V_{CC}
- Reference voltage applied to V_{REF} pin
  - Can be disconnected with VCUT bit to save power
- Input Multiplexer: 8 input channels

Input Mux (262, but 626 similar)

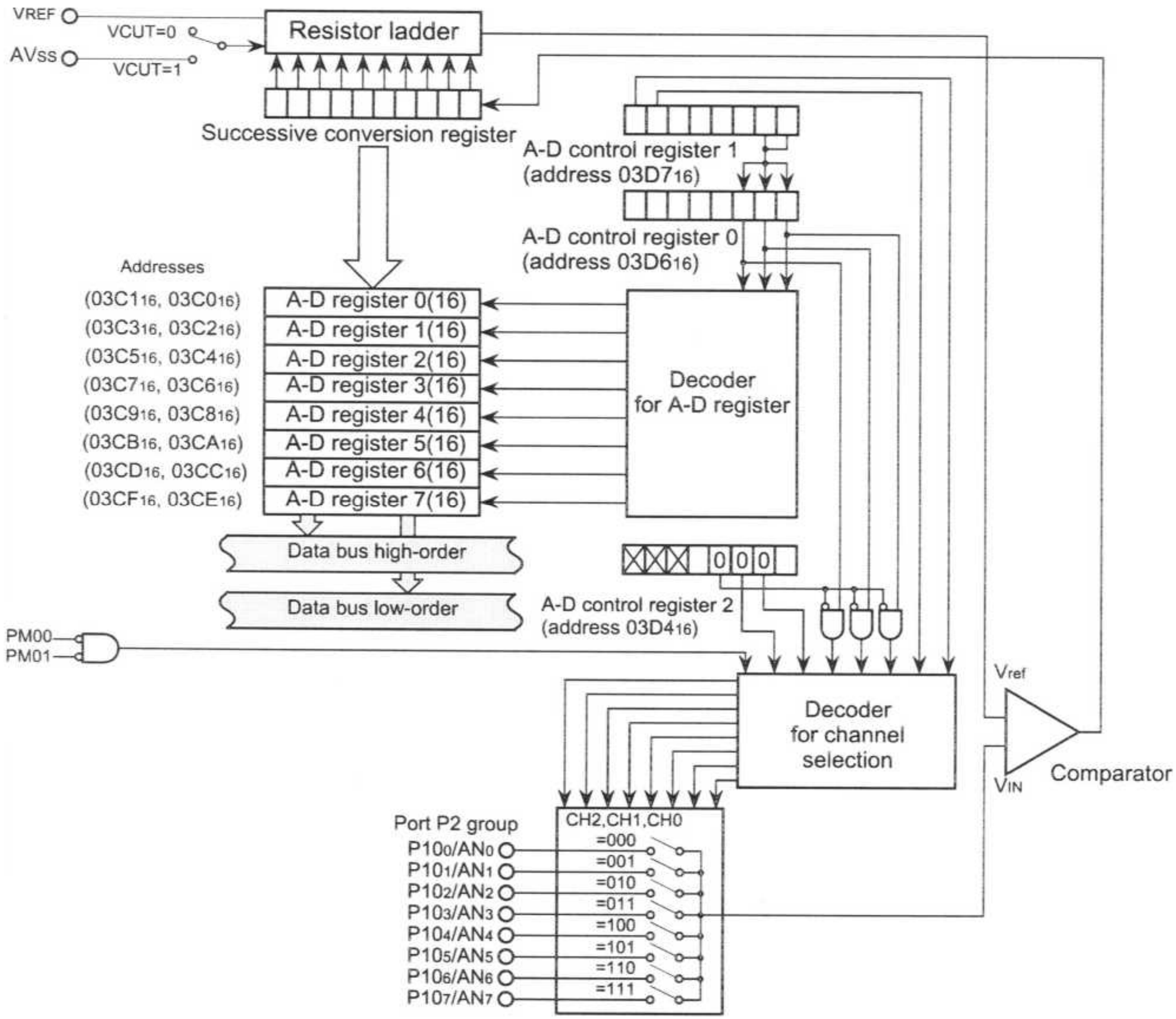


ADC Conversion Speed



- Rates
  - With S/H: 28 ϕ_{AD} cycles for 8 bits, 33 for 10 bits
  - Without S/H: 49 ϕ_{AD} cycles for 8 bits, 59 for 10 bits
- ADC clock generation
  - Can select $\phi_{AD} = f_{AD}$, $f_{AD}/2$, $f_{AD}/3$, $f_{AD}/4$, $f_{AD}/6$, $f_{AD}/12$
  - f_{AD}= f(Xin) = clock/crystal input XIN for MCU
  - See note 2 on p. 152 for frequency restrictions

M30262 Converter Overview (626P similar)



Conversion Modes

Common operation details

- Code starts conversion(s) by setting ADST = 1
- Conversion stops…
  - When complete (ADC sets ADST=0 as indicator) – in one-shot or single sweep mode
  - Code can also stop (set ADST = 0) – primarily for repeat modes
- Result is in result register (16 bits) for that channel (AD0-AD7, 0x03c0-0x03cf)

Modes

- One-shot conversion of a channel
  - Generates interrupt if ADIC register's interrupt level is > 0
- Repeated conversion of a channel
  - No interrupt generated, can read result register instead
- Single sweep mode
  - Converts a set of channels once: Channels 0-1, 0-3, 0-5 or 0-7
- Repeat sweep mode 0
  - Converts a set of channels repeatedly: Channels 0-1, 0-3, 0-5 or 0-7
- Repeat sweep mode 1
  - Converts a set of channels repeatedly: Channels 0, 0-1, 0-2 or 0-3

Control Registers

- ADCON0 (0x03d6), ADCON2 (0x03d4), ADCON1 (0x03d7)

One Shot - Setting Control Registers

```
   adcon0 = 0x80;
        /* 10000000;     /* AN0 input, 1 shot mode, soft trigger
           ||||||||______analog input select bit 0
           |||||||_______analog input select bit 1
           ||||||________analog input select bit 2
           |||||_________A/D operation mode select bit 0
           ||||__________A/D operation mode select bit 1
           |||___________trigger select bit
           ||____________A/D conversion start flag
           |_____________frequency select bit 0 */
   adcon1 = 0x38;
/*   00111000;         ** 10-bit mode, fAD/2, Vref connected
     ||||||||______A/D sweep pin select bit 0
     |||||||_______A/D sweep pin select bit 1
     ||||||________A/D operation mode select bit 1
     |||||_________8/10 bit mode select bit
     ||||__________Frequency select bit 1
     |||___________Vref connect bit
     ||____________External op-amp connection mode bit 0
     |_____________External op-amp connection mode bit 1 */
```

One Shot - Setting Control Registers

```
adcon2 = 0x01;
   /*
00000001;    ** Sample and hold enabled, fAD/2
||||||||______AD conversion method select bit
|||||||_______AD input group select bit 0
||||||________AD input group select bit 1
|||||_________Reserved
||||__________Frequency select bit 2
|||___________Reserved
||____________Reserved
|_____________Reserved */
```

One Shot-Setting Control Interrupts

```
   adic = 0x01;
/* 00000001;              ** Enable the ADC interrupt
   ||||||||______Interrupt priority select bit 0
   |||||||_______Interrupt priority select bit 1
   ||||||________Interrupt priority select bit 2
   |||||_________Interrupt request bit
   ||||__________Reserved
   |||___________Reserved
   ||____________Reserved
   |_____________Reserved */
    _asm ("   fset i") ;  // globally enable interrupts
    adst = 1;                        // Start a conversion here
    while (1){}                      // Program waits here forever
}

#pragma INTERRUPT ADCInt             // compiler directive telling where
                                     // the ADC interrupt is located
void ADCInt(void){
   TempStore = ad0 & 0x03ff;         // Mask off the upper 6 bits of the
                                     // variable leaving only the result
}                                    // in the variable itself
```

Setting Control Registers & Interrupt

In order for this program to run properly, the ADC interrupt vector needs to point to the function. The interrupt vector table is near the end of the startup file "sect30.inc". Insert the function label "_ADCInt" into the interrupt vector table at vector 14 as shown below.

```
.
.
   .lword       dummy_int        ; DMA1(for user)(vector 12)
   .lword       dummy_int        ; Key input interrupt(for user)(vect 13)
   .glb         _ADCInt
   .lword       _ADCInt          ; A-D(for user)(vector 14)
   .lword       dummy_int        ; uart2 transmit(for user)(vector 15)
   .lword       dummy_int        ; uart2 receive(for user)(vector 16)
.
.

#pragma INTERRUPT ADCInt         // compiler directive telling where
                                 // the ADC interrupt is located
void ADCInt(void){
   TempStore = ad0 & 0x03ff;     // Mask off the upper 6 bits of the
                                 // variable leaving only the result
}                                // in the variable itself
```

Repeated ADC

The microcontroller performs repeated A/D conversions, and can read data whenever needed

```
adcon0 = 0x88;
adcon1 = 0x28;
adcon2 = 0X01;
adst = 1;   // Start a conversion here
```

Then in your procedure

```
TempStore = ad0 & 0x03ff;
```

ADC as a Temperature Sensor

A "Thermistor" device is used to convert temperature into a voltage.

There is an equation that needs to be run in software that converts the voltage read to a temperature value. This depends on measure-ments taken on the device.

The code will take the raw ADC value and convert to binary value

deg C	deg F	ThV
-5	23	4.2580
0	32	3.2770
5	41	2.5460
10	50	1.9930
15	59	1.5730
20	68	1.2500
25	77	1.0000
30	86	0.8055
35	95	0.6528
40	104	0.5323
45	113	0.4365
50	122	0.3599

Converting ADC Values

To convert, you will need to use a floating point library (math.h).

Most often, you will want to output ASCII characters. You will need to convert the floating point number to ASCII via successive division.

See the lab web page for examples.

D-to-A Conversion

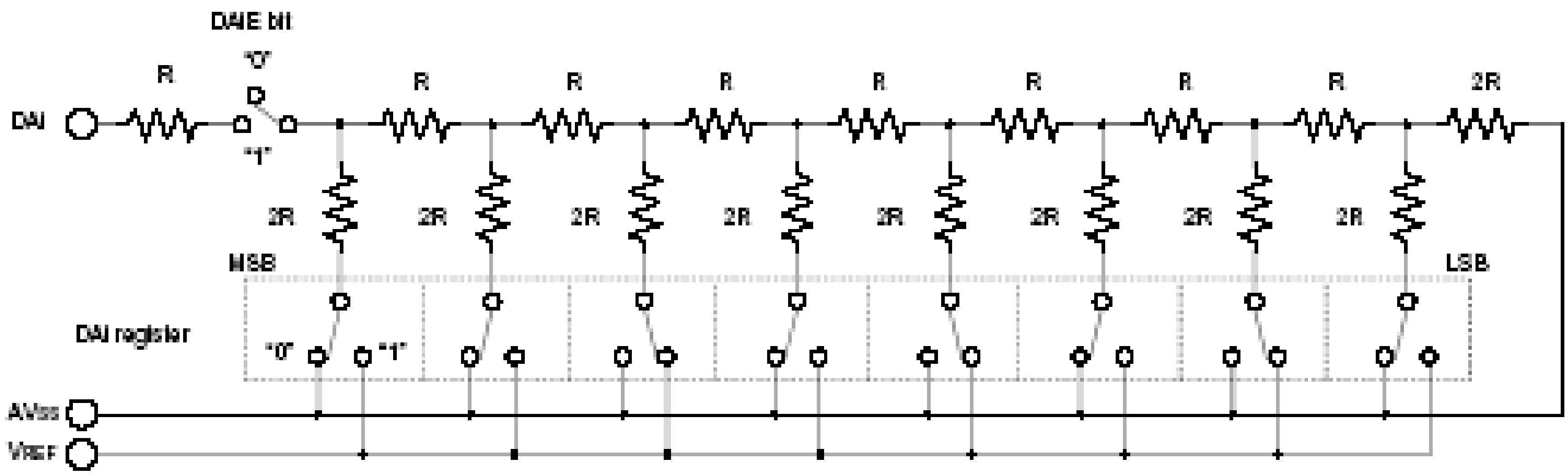
This is an 8-bit, R-2R type D-A converter. There are two independent D-A converters.

D-A conversion is performed by writing to the DAi register (i = 0 to 1). To output the result of conversion, set the DACON register's DAiE bit to "1" (output enabled). Before D-A conversion can be used, the corresponding port direction bit must be cleared to "0" (input mode). Setting the DAiE bit to "1" removes a pull-up from the corresponding port.

Output analog voltage (V) is determined by a set value (n : decimal) in the DAi register.

V = VREF X n/ 256 (n = 0 to 255), VREF : reference voltage

DA1=varname; // write to the DAC (varname is a char)


DAIE bit
"0"
"1"
R
R
R
R
R
R
R
R
2R
DAI
2R
2R
2R
2R
2R
2R
2R
2R
MSB
LSB
DAI register
"0"
"1"
AVss
VREF

D-A control register (Note)

b7 b6 b5 b4 b3 b2 b1 b0

Symbol	Address	After reset
DACON	$03DC_{16}$	00_{16}

Bit symbol	Bit name	Function	RW
DA0E	D-A0 output enable bit	0 : Output disabled 1 : Output enabled	RW
DA1E	D-A1 output enable bit	0 : Output disabled 1 : Output enabled	RW
— (b7-b2)	Nothing is assigned. In an attempt to write to these bits, write "0". The value, if read, turns out to be "0"		—

Note: When not using the D-A converter, clear the DAiE bit (i = 0 to 1) to "0" (output disabled) to reduce the unnecessary current consumption in the chip and set the DAi register to '00_{16}' to prevent current from flowing into the R-2R resistor ladder.

D-Ai register (Note) (i= 0 to 1)

b7 b0

Symbol	Address	After reset
DA0	$03D8_{16}$	Indeterminate
DA1	$03DA_{16}$	Indeterminate

Function	RW
Output value of D-A conversion	RW