

# NET3001 F07

Analog to Digital Conversion

# Digital vs Analog

- most of the activity on microcontrollers is digital
  - programmer sees 0's and 1's
    - to refer to groups of these, we use byte/word
  - engineers see 0V and 3V (voltage)
- many real world activities take place with *variable* quantities, not purely *on* or *off* values

# Transducers

- *input transducer*: converts a real-world measurement to either
  - 0V/3V (digital input)
  - 0...3V (analog input)
- *output transducer*: converts a microcontroller output to a real world event
  - 0V/3V (digital output)
  - pulses between 0V/3V (pulse width modulation)

# Transducers (cont'd)

- *input transducers:*
  - push button                      digital
  - microphone                      analog
  - light sensor                      digital
  - motion detector              digital
  - temperature sensor analog
- *output transducers:*
  - LED                              digital
  - motor                              digital (needs ampl)
  - speaker                          analog/PWM

# Transducers

- in order to allow the microcontroller to measure an analog value
  - the analog value must be converted from its base form into a voltage between 0 & 3V
    - for example, a microphone will convert sound pressure into a small voltage
    - a temperature sensor will convert heat into a small analog voltage
  - the voltage must not change faster than the microcontroller can measure it
    - if we only measure the value 1000 times per second, it should not change faster than that

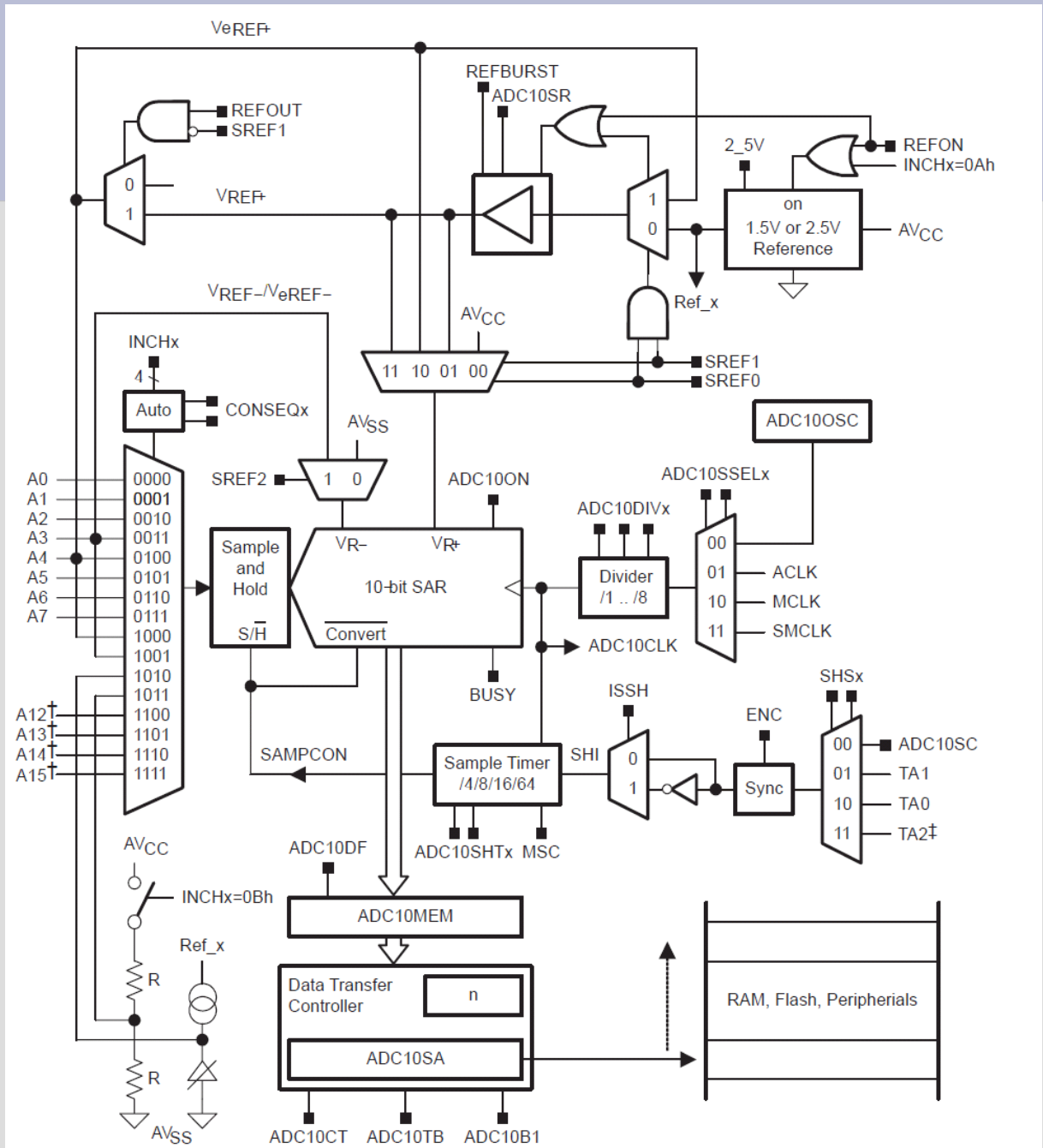
# Analog to Digital Conversion

- there is a section of the microcontroller which is capable of measuring the small voltage on a pin
- returns a value between 0 and 0x3FF
  - a 10 bit conversion
  - 0V returns a measurement of 0
  - 3V returns a measurement of 0x3FF (1023)
  - so each unit of measurement is

$$unit = 3V / 1024 = .00293 V = 3mV$$

# ADC10

- any Ax pin can be input
- INCHx chooses
- ADC10AE disconnects the port
- ENC starts
- BUSY shows when ready
- answer is in ADC10MEM
- ADC10ON powers up



# ADC10 (features not used)

- can be configured to measure
  - between 0 & 3V
  - between 0 & 1.5V
  - between any two (small) voltages ( $V_{ref}$ )
- can be configured to measure several pins in a row (CONSEQ)
- can be configured to interrupt us when a conversion (or group of conversions) is done
- can be configured to drop the results directly into ram

# ADC10 Registers

- ADC10AE
  - one bit per Ax pin
  - disconnects the pin from the port, and connects it to the ADC10
- ADC10CTL0

range select	sample hold	sample rate	ref out	ref burst	mult conv	high ref	ref on	ADC10 ON	IE	IFG	enable ENC	start SC
-----------------	----------------	----------------	------------	--------------	--------------	-------------	-----------	-------------	----	-----	---------------	-------------

- ADC10ON enable chip module
- ENC enable conversion
- SC trigger a conversion
- IE interrupt enable
- IFG interrupt flag

# ADC10 Registers

- ADC10CTL1

inch	s/h select	data format	inv s/h	ADC10 clock rate	ADC10 clock src	CONSEQ	BUSY
------	---------------	----------------	------------	---------------------	--------------------	--------	------

- INCH select Ax pin to convert
- CONSEQ single, sequence or repeat
- BUSY will read a "1" until the conversion is done

- ADC10MEM

0 0 0 0 0 0	conversion result (0...0x3FF, unsigned)
-------------	---