Sensors
and
Analog to Digital Conversion

TODO: Motors and Relays
Introduction

• Microprocessor systems must interact with their environment. They use sensors and actuators in order to do this.

• Sensors and actuators are actually transducers

  A transducer is a device that converts one physical quantity into another

  – examples include:
    • a mercury-in-glass thermometer (converts temperature into displacement of a column of mercury)
    • a microphone (converts sound into an electrical signal).
    • a motor converts electrical energy to mechanical energy.

• We will look at sensors and actuators in this lecture.
Terminology

- Transducers convert one form of energy into another
- Sensors/Actuators are input/output transducers
- Sensors can be *passive* (e.g. change in resistance) or *active* (output is a voltage or current level)
- Sensors can be *analog* (e.g. thermocouples) or *digital* (e.g. digital tachometer)
Sensors

• Almost any physical property of a material that changes in response to some excitation can be used to produce a sensor

• Widely used sensors include those that are:
  – resistive
  – inductive
  – capacitive
  – piezoelectric
  – photoresistive
  – elastic
  – thermal
## Transducer types

<table>
<thead>
<tr>
<th>Quantity being Measured</th>
<th>Input Device (Sensor)</th>
<th>Output Device (Actuator)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Light Level</td>
<td>Light Dependant Resistor (LDR), Photodiode, Phototransistor, Solar Cell</td>
<td>Lights &amp; Lamps, LED's &amp; Displays, Fiber Optics</td>
</tr>
<tr>
<td>Temperature</td>
<td>Thermocouple, Thermistor, Thermostat, Resistive temperature detectors (RTD)</td>
<td>Heater, Fan, Peltier Elements</td>
</tr>
<tr>
<td>Force/Pressure</td>
<td>Strain Gauge, Pressure Switch, Load Cells</td>
<td>Lifts &amp; Jacks, Electromagnetic, Vibration</td>
</tr>
<tr>
<td>Position</td>
<td>Potentiometer, Encoders, Reflective/Slotted Opto-switch, LVDT</td>
<td>Motor, Solenoid, Panel Meters</td>
</tr>
<tr>
<td>Speed</td>
<td>Tacho-generator, Reflective/Slotted Opto-coupler, Doppler Effect Sensors</td>
<td>AC and DC Motors, Stepper Motor, Brake</td>
</tr>
<tr>
<td>Sound</td>
<td>Carbon Microphone, Piezo-electric Crystal</td>
<td>Bell, Buzzer, Loudspeaker</td>
</tr>
</tbody>
</table>
Temperature Sensors

- Resistive thermometers
  - typical devices use platinum wire (such a device is called a platinum resistance thermometers or PRT)
  - linear but has poor sensitivity

A typical PRT element

A sheathed PRT
Temperature Sensors

• Thermistors
  – use materials with a high thermal coefficient of resistance
  – *sensitive* but highly *non-linear*

  – Thermistors (thermally sensitive resistors); Platinum Resistance Thermometer (PRT), very high accuracy.
Temperature Sensors

• pn junctions
  – a semiconductor device with the properties of a diode (we will consider semiconductors and diodes later)
  – inexpensive, linear and easy to use
  – limited temperature range (perhaps -50°C to 150 °C) due to nature of semiconductor material

\[ \text{pn-junction sensor} \]
Temperature Sensors

- Bimetallic switch (electro-mechanical) – used in thermostats. Can be “creep” or “snap” action.

Creep-action: coil or spiral that unwinds or coils with changing temperature
Thermocouples

- Two dissimilar metals induce voltage difference (few mV per 10K) – electro-thermal or Seebeck effect

- Use op-amp to process/amplify the voltage
- Absolute accuracy of 1K is difficult
# Thermocouple Sensor Colour Codes

*Extension and Compensating Leads*

<table>
<thead>
<tr>
<th>Code Type</th>
<th>Conductors (+/-)</th>
<th>Sensitivity</th>
<th>British BS 1843:1952</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>E</strong></td>
<td>Nickel Chromium / Constantan</td>
<td>-200 to 900°C</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>J</strong></td>
<td>Iron / Constantan</td>
<td>0 to 750°C</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>K</strong></td>
<td>Nickel Chromium / Nickel Aluminium</td>
<td>-200 to 1250°C</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>N</strong></td>
<td>Nicrosil / Nisil</td>
<td>0 to 1250°C</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>T</strong></td>
<td>Copper / Constantan</td>
<td>-200 to 350°C</td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
<tr>
<td><strong>U</strong></td>
<td>Copper / Copper Nickel</td>
<td>0 to 1450°C</td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
</tbody>
</table>
Light Sensors

- Photovoltaic
  - light falling on a $pn$-junction can be used to generate electricity from light energy (as in a solar cell)
  - small devices used as sensors are called photodiodes
  - fast acting, but the voltage produced is not linearly related to light intensity
Light Sensors

• Photoconductive
  – such devices do not produce electricity, but simply change their resistance
  – photodiode (as described earlier) can be used in this way to produce a linear device
  – phototransistors act like photodiodes but with greater sensitivity
  – light-dependent resistors (LDRs) are slow, but respond like the human eye
Light sensors: photoconductive cells

- Light dependent resistor (LDR) cell
Light level sensitive switch
Photojunction devices

photodiode

phototransistor
Photovoltaic Solar Cells

- Can convert about 20% of light power into electricity
- Voltage is low (diode drop, ~0.6V)

Solar power is 1.4kW/m^2
Photomultiplier tubes (PMT)

• Most sensitive of light sensors (can detect individual photons)
• Acts as a current source

![Diagram of a photomultiplier tube](image)

**Figure 1**

- **Incoming Photon**
- **Photocathode**
- **Window**
- **Dynodes**
- **Anode**
- **Electrons**
- **Focusing Electrode**
- **Voltage Dropping Resistors**
- **Power Supply**
- **Output Meter**
Force Sensors

- Strain gauge
  - stretching in one direction increases the resistance of the device, while stretching in the other direction has little effect
  - can be bonded to a surface to measure strain
  - used within load cells and pressure sensors
Displacement Sensors

• **Potentiometers**
  – resistive potentiometers are one of the most widely used forms of position sensor
  – can be angular or linear
  – consists of a length of resistive material with a sliding contact onto the resistive track
  – when used as a position transducer a potential is placed across the two end terminals, the voltage on the sliding contact is then proportional to its position
  – an inexpensive and easy to use sensor
Displacement Sensors

• Inductive proximity sensors

I. coil inductance is greatly affected by the presence of ferromagnetic materials
II. here the proximity of a ferromagnetic plate is determined by measuring the inductance of a coil
III. we will look at inductance in later lectures
Displacement Sensors

• Switches
  – simplest form of *digital* displacement sensor
  • many forms: lever or push-rod operated microswitches; float switches; pressure switches; etc.

![A limit switch](image1)

A limit switch

![A float switch](image2)

A float switch
• **Absolute position encoders**
  – a pattern of light and dark strips is printed on to a strip and is detected by a sensor that moves along it
  • the pattern takes the form of a series of lines as shown below
  • it is arranged so that the combination is unique at each point
  • sensor is an array of photodiodes
Displacement Sensors

• Incremental position encoder
  – uses a single line that alternates black/white
    • two slightly offset sensors produce outputs as shown below
  • detects motion in either direction, pulses are counted to determine absolute position (which must be initially reset)
Displacement Sensors

• Other counting techniques
  – several methods use counting to determine position
  • two examples are given below
Positional Sensors: potentiometer

Can be Linear or Rotational

Processing circuit
Positional Sensors: LVDT

Linear Variable Differential Transformer
Positional Sensors: Inductive Proximity Switch

- Detects the presence of metallic objects (non-contact) via changing inductance

- Sensor has 4 main parts: field producing Oscillator via a Coil; Detection Circuit which detects change in the field; and Output Circuit generating a signal (NO or NC)

Used in traffic lights (inductive loop buried under the road). Sense objects in dirty environment. Does not work for non-metallic objects. Omni-directional.
Positional Sensors: Rotary Encoders

- Incremental and absolute types
- Incremental encoder needs a counter, loses absolute position between power glitches, must be re-homed
- Absolute encoders common in CD/DVD drives
Motion Sensors

• Motion sensors measure quantities such as velocity and acceleration
  – can be obtained by differentiating displacement
  – differentiation tends to amplify high-frequency noise

• Alternatively can be measured directly
  – some sensors give velocity directly
    • e.g. measuring frequency of pulses in the counting techniques described earlier gives speed rather than position
  – some sensors give acceleration directly
    • e.g. accelerometers usually measure the force on a mass
Motion sensors/transducers

- Switches, solenoids, relays, motors, etc.
- Motors
  - DC
    - Brushed/brushless
    - Servo
    - Stepper motors
  - AC

**Stepper motor**

- Brushed motor – permanent magnets on armature, rotor acts as electromagnet
- Brushless motor – permanent magnet on the rotor, electromagnets on armature are switched
Piezo transducers

- Detect motion (high and low frequency)
- Sound (lab this week), pressure, fast motion
- Cheap, reliable but has a very limited range of motion
Sound Sensors

• Microphones
  – a number of forms are available
    • e.g. carbon (resistive), capacitive, piezoelectric and moving-coil microphones
    • moving-coil devices use a magnet and a coil attached to a diaphragm – we will discuss electromagnetism later

![Diagram of a microphone with sound waves and output voltage graph]
Sound transducers

- microphone

- speaker

- Note: voice coil can also be used to generate fast motion
Acceloremeter and Gyroscope Basics MEMS
Acceleration Fundamentals

• What is Acceleration?
  – Definition: the time rate of change of velocity. A.K.A.: the time rate of change of the time rate of change of distance

• What are the units?
  – Acceleration is measured in \((\text{m/s})/\text{s}\)

• What is a “g”?
  – A “g” is a unit of acceleration equal to Earth’s gravity at sea level \((1 \text{ g} = 9.81 \text{ m/s}^2)\)

• What is the time rate of change of velocity?
  – When plotted on a graph, velocity is the slope of distance versus time
  – Acceleration is the slope of velocity versus time

\[ a = \frac{\partial v}{\partial t} = \frac{\partial^2 x}{\partial t^2} \]
How to find velocity from distance traveled

\[ V(t=1.160) = 0 \text{ m/s} \]

\[ V(t=0.640) = 1 \text{ m/s} \]
How to find acceleration from velocity

- $a_{(t=0.960)} = 0 \text{ m/s}^2$
- $a_{(t=1.040)} = -10 \text{ m/s}^2$
- $V_{(t=0.640)} = 1 \text{ m/s}$
- $V_{(t=1.160)} = 0 \text{ m/s}$
Acceleration vs. Time

\[ a(t=0.960) = 0 \text{ m/s}^2 \]

\[ a(t=1.040) = -10 \text{ m/s}^2 \]
# Acceleration in Human terms

<table>
<thead>
<tr>
<th>Description</th>
<th>“g” level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Earth’s gravity</td>
<td>1g</td>
</tr>
<tr>
<td>Passenger car in corner</td>
<td>2g</td>
</tr>
<tr>
<td>Bumps in road</td>
<td>2g</td>
</tr>
<tr>
<td>Racing car driver in corner</td>
<td>3g</td>
</tr>
<tr>
<td>Bobsled rider in corner</td>
<td>5g</td>
</tr>
<tr>
<td>Human unconsciousness</td>
<td>7g</td>
</tr>
<tr>
<td>Space shuttle</td>
<td>10g</td>
</tr>
</tbody>
</table>
Why to measure Acceleration?

• Why measure acceleration?
  – Acceleration is a physical characteristic of a system.
  – The measurement of acceleration is used as an input into some types of control systems.
  – The control systems use the measured acceleration to correct for changing dynamic conditions
Sensor Types

• Capacitive
  – Metal beam or micromachined feature produces capacitance; change in capacitance related to acceleration

• Piezoelectric
  – Piezoelectric crystal mounted to mass – voltage output converted to acceleration

• Piezoresistive
  – Beam or micromachined feature whose resistance changes with acceleration
Sensor Types

• Hall Effect
  – Motion converted to electrical signal by sensing of changing magnetic fields

• Magnetoresistive
  – Material resistivity changes in presence of magnetic field

• Heat Transfer
  – Location of heated mass tracked during acceleration by sensing temperature
MEMS sensors

- Microelectromechanical systems (MEMS, also written as micro-electro-mechanical, MicroElectroMechanical or microelectronic and microelectromechanical systems and the related micromechatronics) is the technology of microscopic devices, particularly those with moving parts.
MEMS accelerometer

- A MEMS accelerometer differs from integrated circuits in that a “proof mass” is machined into the silicon. Any displacement of the component causes this mass to move slightly according to Newton’s second law, and that change is detected by sensors. Usually the proof mass disturbs the capacitance of a nearby node; that change is measured and filtered.

https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/
MEMS Gyroscope

• The gyroscope measures angular rate using the Coriolis Effect. When a mass is moving in a particular direction with a particular velocity and when an external angular rate will be applied as show with the green arrow a force will occur, as show with the blue red arrow, which will cause perpendicular displacement of the mass. So similar to the accelerometer, this displacement will cause change in capacitance which will be measured, processed and it will correspond to a particular angular rate.

https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/
Typical Applications

• Tilt / Roll

• Vibration / “Rough-road” detection
  – Can be used to isolate vibration of mechanical system from outside sources

• Vehicle skid detection
  – Often used with systems that deploy “smart” braking to regain control of vehicle

• Impact detection
  – To determine the severity of impact, or to log when an impact has occurred

• Input / feedback for active suspension control systems
  – Keeps vehicle level
Important Setup Rules

• Rigid Mounting
  – Bees Wax
  – Double Sided tape
  – Bolt(s)

• No Loose Wires
  – Loose wires can create false signals
  – Secure wires firmly to mounting body

• Weight of Sensor
  – Should be approximately an order of magnitude less than object being measured
    • Example: CAS = 47g; accelerating object should be more than 470g

• Don’t drop the sensor!
  – Extreme jarring accelerations can cause permanent errors in device output
MPU6050

• MPU6050: Gyroscope + Accelerometer
  – ±2g, ±4g, ±8g and ±16g
  – ±250, ±500, ±1000, and ±2000°/sec

• VCC and GND for power supply: 3-5v

• SCL and SDA for I2C communication with MCU

• XCL and XDA to connect magnetic sensor
SENSOR INTERFACING
Sensor Interfacing

• Switches
  – switch interfacing is also simple
    • can use a single resistor as below to produce a voltage output
    • all mechanical switches suffer from switch bounce
Sensor Interfacing

• Resistive devices
  – can be very simple

  • e.g. in a potentiometer, with a fixed voltage across the outer terminals, the voltage on the third is directly related to position

  • where the resistance of the device changes with the quantity being measured, this change can be converted into a voltage signal using a potential divider – as shown

  • the output of this arrangement is not linearly related to the change in resistance
Sensor Interfacing

- **Whetstone bridge**
  - The Wheatstone bridge illustrates the concept of a difference measurement, which can be extremely accurate.
  - Although today digital multimeters provide the simplest way to measure a resistance. The Wheatstone Bridge can still be used to measure very low values of resistances down in the milli-Ohms range.
  - we can use the Wheatstone Bridge Circuit to interface various transducers and sensors to these amplifier circuits.

![Wheatstone Bridge Diagram]
Sensor Interfacing

- In the figure Rx is the resistor to be measured if all four resistor values and the supply voltage (Vs) are known.

\[ V_g = \left( \frac{R_2}{R_1 + R_2} - \frac{R_x}{R_x + R_3} \right) V_s \]
Capacitance and Inductance Measurements

Schering

Hay

Wien

Owen

Week 6: Sensors
Describing Sensor Performance

• Range
  – maximum and minimum values that can be measured

• Resolution or discrimination
  – smallest discernible change in the measured value

• Error
  – difference between the measured and actual values
    • random errors
    • systematic errors

• Accuracy, inaccuracy, uncertainty
  – accuracy is a measure of the maximum expected error
Describing Sensor Performance

• **Precision**
  – a measure of the lack of random errors (scatter)

(a) Low precision, low accuracy
(b) High precision, low accuracy
(c) High precision, high accuracy
Describing Sensor Performance

• Linearity
  – maximum deviation from a ‘straight-line’ response
  – normally expressed as a percentage of the full-scale value

• Sensitivity
  – a measure of the change produced at the output for a given change in the quantity being measured
Key Points

• A wide range of sensors is available
• Some sensors produce an output voltage related to the measured quantity and therefore supply power
• Other devices simply change their physical properties
• Some sensors produce an output that is linearly related to the quantity being measured, others do not
• Interfacing may be required to produce signals in the correct form
Instrumentation Resources

• https://www.youtube.com/watch?v=hijgA8A-O64
• https://www.youtube.com/watch?v=9opuvLXAetI
Actuators

Motors, Relays, etc
Motors

• When moving or rotating objects with microcontrollers, three kinds of motors are generally used:
  – DC motors,
  – RC servomotors
  – Stepper motors.

• Motors are used to convert electrical energy to mechanical energy. Induction is basic principle behind the operation of motors.
Motors

- If magnets are mounted on a rotating shaft surrounded by the wire with a current flowing on it, you obtain a motor. When the magnets are alternately attracted to one magnet and repulsed by the other, the shaft rotates and a circular motion is obtained.

https://itp.nyu.edu/physcomp/lessons/dc-motors/dc-motors-the-basics/
Analog To Digital Convertors
Analog signals

- A signal representing continuous signal that contains time-varying quantities, for example
  - Pressure of sound waves induce voltage on microphones
  - Temperature of the air
  - Voltage across an analog circuit element, resistor, capacitor, transistor, etc

Sound generated by vibrating membrane (wikipedia)
What is Analog to Digital Conversion?

• Analog to Digital conversion is to convert an analog quantity into a digital number.
• Analog input is typically sampled regularly at intervals of T or at irregular time intervals, so the result is a sequence digital numbers.

\[ V(t) \rightarrow \text{ADC} \rightarrow V(n) \]
Analog to Digital Conversion

• Have an input you wish to measure and input in digital form
  – Measure voltage/current
  – Indirectly measure other properties such as pressure, temperature
  – Value bounded between minimum and maximum
  – The output is a digital number that represents the input value

• MSP430 has a built in A/D sub-system.
Quantization

- Let $0 \leq V(t) < 5V$
- Assume that ADC does 3 bits of conversion, in other words, $2^3 = 8$ intervals.
- Each voltage interval is assigned a quantization code

Any voltage between 0V and 0.625V maps to bit sequence 000
Some Observations

- $V_{\text{ref}}$ is broken up into $V_{\text{ref}}/2^n$ voltage ranges when $n$ is the number of bits of conversion.
- The midpoint for each range is the nominal “quantized” value resulting from the conversion.
- You do not get the exact measurement, but only an approximation limited by the precision of the result.
- Analog to digital conversion resembles to the analog signal if the number of bits of conversion is high.
Popular A/D Techniques

- Integrating ADC
- Flash (parallel) ADC
- Pipelined ADC
- Ramp/Counter ADC
- Successive Approximation Register (SAR) ADC
- Sigma Delta ADC
- Two step ADC
Integrating ADC

- Integrating ADCs provide:
  - high resolution
  - good line frequency
  - noise rejection.

- These converters have been around for quite some time.

- The integrating architecture provides an approach to converting a low bandwidth analog signal into its digital representation.

- These type of converters often include built-in drivers for LCD or LED displays and are found in many portable instrument applications, including digital panel meters and digital multi-meters.
Single Slope Integrating ADC

- An unknown input voltage is integrated and the value compared against a known reference value.
- The time it takes for the integrator to trip the comparator is proportional to the unknown voltage ($T_{INT}/V_{IN}$).
- The accuracy is also dependent on the tolerances of the integrator's R and C values. To overcome this problem, the dual-slope integrating architecture is used.
Dual Slope Integrating ADC

- The advantage over the single-slope is that the final conversion result is insensitive to errors in the component values.
- In other words, any error introduced by a component value during the integrate cycle will be cancelled out during the de-integrate phase.

$$\frac{V_{IN}}{T_{INT}} = \frac{V_{REF}}{T_{DE-INT}}$$

$$T_{INT} = \text{fixed}$$

$$T_{DE-INT} \propto \frac{V_{IN}}{V_{REF}}$$

https://www.youtube.com/watch?v=pzXZnvEKMxs
Multi Slope ADCs

- The normal limit for resolution of the dual-slope architecture is based on the speed of the error comparator.
- For a 20-bit converter and a 1MHz clock, the conversion time would be about 2 seconds.
- The ramp rate seen by the error comparator is about $2V/10^6$ divided by 1 microsecond. This is about 2 microvolts/microsecond. With such a small slew rate, the error comparator would allow the integrator to exceed its trip point. This overshoot (measured at the integrator output) is called the "residue".
- Instead, we could convert the first 10 most significant bits (one integrate/de-integrate cycle), then amplify the residue by 25, then deintegrate again, then amplify the residue by 25, and then deintegrate for the last time.
Flash (parallel) ADC

- Flash analog-to-digital converters are the fastest way to convert an analog signal to a digital signal.
- They are suitable for applications requiring very large bandwidths.
- However, flash converters consume a lot of power, have relatively low resolution, and can be quite expensive.
- This limits them to high frequency applications that typically cannot be addressed any other way.
- Examples include data acquisition, satellite communication, radar processing, sampling oscilloscopes, and high-density disk drives.
- They are based on resistor network and biased comparator inputs.
Flash ADC

Example: 2 bit (4 levels) Flash ADC
Notes

- Digital thermometer code resembles to the mercury thermometer. If there is a 1 in the code, all of the inputs below should be 1.
- Decoder determines the uppermost 1 and outputs its code as the result of the AD convertor
Notes

- The comparators are typically a cascade of wideband low gain stages. They are low gain because at high frequencies it's difficult to obtain both wide bandwidth and high gain.

- They are designed for low voltage offset, such that the input offset of each comparator is smaller than a LSB of the ADC. Otherwise, the comparator's offset could falsely trip the comparator, resulting in a digital output code not representative of a thermometer code.

- A regenerative latch at each comparator output stores the result. The latch has positive feedback, so that the end state is forced to either a "1" or a "0".
Notes

• When the input signal changes before all the comparators have completed their decision, the ADC performance is adversely impacted.
• Measuring spurious free dynamic range (SFDR) is another good way to observe converter performance. SFDR is the ratio of the power of the maximum signal component and the power of the noise component.
• The most serious impact is a drop in signal-to-noise ratio plus distortion (SINAD) as the frequency of the analog input frequency increases. It is the ratio of total received power of the signal to the noise-plus-distortion power.
• The sampling frequency can be improved by adding a track-and-hold (T/H) circuit in front of the ADC.
Notes

• The sampling frequency can be improved by adding a track-and-hold (T/H) circuit in front of the ADC.
Pipelined ADC

- They can provide sampling rates from a few megasamples per second (MS/s) up to 100MS/s+, with resolutions from 8 to 16 bits.
- They offer the resolution and sampling rate to cover a wide range of applications.
- Popular areas of application are CCD imaging, ultrasonic medical imaging, digital receiver, base station, digital video (for example, HDTV), xDSL, cable modem, and fast Ethernet.
The analog input is first sampled and held steady by a sample-and-hold (S&H), while the flash ADC in stage one quantizes it to three bits. The 3-bit output is then fed to a 3-bit DAC and the analog output is subtracted from the input. This "residue" is amplified by four and sent to the next stage. This residue moves through the pipeline, providing three bits per stage. The last stage resolves the last 4LSB bits.
Ramp Counter A/D Conversion

- Ramp/Counter with analog comparator and D/A

If the input voltage is a maximum, counter has to go to the maximum value to get the answer. Can be time consuming.
Successive Approximation Register ADC

- Successive-approximation-register (SAR) analog-to-digital converters (ADCs) are used for medium-to-high-resolution applications.

- SAR ADCs
  - resolutions from 8 to 16 bits
  - Lower than 5 million samples per second
  - low power consumption
  - small form factors

- They are ideal for portable/battery-powered instruments, pen digitizers, industrial controls, and data/signal acquisition.

- For more information
  
  http://www.maxim-ic.com/appnotes.cfm/an_pk/1080/
SAR ADC

- Track/Hold system gets the input voltage and holds it.
- Comparator compares $V_{DAC}$ and $V_{IN}$, and SAR logic decides on the next voltage to try.
- Next to try is loaded into N-bit register and this value is converted to analog equivalence. Then this voltage is compared against $V_{IN}$.
- This continues until $V_{IN}$ and $V_{DAC}$ are very close.
t_{AD} : Aperture delay is the time defined between the falling edge of the sampling clock and the instant when an actual sample is taken

(t_{AJ} : Aperture jitter is the sample to sample variation in the aperture delay
4 bit Conversion Timing

As the time passes, SAR ADC pin points the input voltage and determines a closer match to the input
Sigma Delta ADC

• Sigma Delta analog-to-digital converters (ADCs) are used predominately in lower speed applications requiring a trade off of speed for resolution by oversampling, followed by filtering to reduce noise.

• 24 bit Sigma Delta converters are common in Audio designs, instrumentation and Sonar.

• Bandwidths are typically less than 1MHz with a range of 12 to 18 true bits.
Two Step ADC

- Two Step analog-to-digital converters (ADCs) are also known as subranging converters and sometimes referred to as multi-step or half flash (slower than Flash architecture).
- This is a cross between a Flash ADC and pipeline ADC and can achieve higher resolution or smaller die size and power for a given resolution are needed vs. a Flash ADC. Example MAX153.
Comparison of ADC Technologies

Integrating, Flash, SAR, Sigma-Delta, Pipelined ADCs

http://www.maxim-ic.com/appnotes.cfm/an_pk/2094/
Preference Guide

• Flash: Ultra-High Speed, high power consumption
• SAR: Medium-to-High Resolution (8-16 bits), lower than 5Msp, low power, small form factor
• Integrating: Monitoring DC signals, high resolution, low power consumption, good noise performance.
• Pipeline: High speed, 100+ Msp, 8-16 bits, lower power consumption than flash ADCs.
• Sigma-Delta: High resolution, low to medium speed, no precision external components, simultaneous 50/60Hz rejection, digital filter for reducing anti-aliasing.
Disadvantages

• Flash: Sparkle codes / metastability, high power consumption, large size, expensive.
• SAR: Speed limited to ~5Msps. May require anti-aliasing filter.
• Integrating: Slow Conversion rate. High precision external components required to achieve accuracy.
• Pipeline: Parallelism increases throughput at the expense of power and latency.
• Sigma-Delta: Higher order (4th order or higher) - multibit ADC and multibit feedback DAC.
Conversion Times

• Flash: Conversion Time does not change with increased resolution.
• SAR: Increases linearly with increased resolution.
• Integrating: Conversion time doubles with every bit increase in resolution.
• Pipeline: Increases linearly with increased resolution.
• Sigma-Delta: Tradeoff between data output rate and noise free resolution.
Resolution

- Flash: Component matching typically limits resolution to 8 bits.
- SAR: Component matching requirements double with every bit increase in resolution.
- Integrating: Component matching does not increase with increase in resolution.
- Pipeline: Component matching requirements double with every bit increase in resolution.
- Sigma-Delta: Component matching requirements double with every bit increase in resolution.
Size

• Flash: $2^N - 1$ comparators, die size and power increases exponentially with resolution.

• SAR: Die increases linearly with increase in resolution.

• Integrating: Core die size will not materially change with increase in resolution.

• Pipeline: Die increases linearly with increase in resolution.

• Sigma-Delta: Core die size will not materially change with increase in resolution.
MSP430 ADC12_B
Analog to Digital Converter
ADC12_B | Overview

- Up to 200ksps
- Ultra Low Power Operation
  - 63µA in single ended mode
  - 95µA in differential mode
- Works down to 1.8V
- 8 differential/16 single-ended inputs
- Internal Battery and Temperature Sensor
- Specialized Window Comparator for saving CPU overhead
- DMA-enabled for intelligent interrupt processing

The reference voltage is the maximum value that the ADC can convert.
MSP430 ADC12_B

• The ADC12_B module supports fast 12-bit analog-to-digital conversions. The module implements a 12-bit SAR core, sample select control, and up to 32 independent conversion-and-control buffers.

• The conversion-and-control buffer allows up to 32 independent analog-to-digital converter (ADC) samples to be converted and stored without any CPU intervention.
MSP430 ADC12_B

- **ADC12_B** features include:
  - 200-kSPS maximum conversion rate at max resolution of 12 bits
  - Monotonic 12-bit converter with no missing codes
  - Sample-and-hold with programmable sampling periods controlled by software or timers
  - Conversion initiation by software or timers
  - Software-selectable on-chip reference voltage generation (1.2 V, 2.0 V, or 2.5 V) with option to make available externally
  - Software-selectable internal or external reference
  - Up to 32 individually configurable external input channels with single-ended or differential input selection available
ADC12_B

- Internal conversion channels for internal temperature sensor and $1/2 \times \text{AVCC}$ and four more internal channels available on select devices (see the device-specific data sheet for availability and function)
- Independent channel-selectable reference sources for both positive and negative references
- Selectable conversion clock source Single-channel, repeat-single-channel, sequence (autoscan), and repeat-sequence (repeated autoscan) conversion modes
- Interrupt vector register for fast decoding of 38 ADC interrupts
- 32 conversion-result storage registers
- Window comparator for low-power monitoring of input signals of conversion-result registers
ADC_B BLOCK DIAGRAM
ADC Core

• Core converts an analog input to its 12-bit digital representation and stores the result in conversion memory; the conversion formula is

\[ N_{ADC} = 4095 \cdot \frac{V_{in} - V_{R-}}{V_{R+} - V_{R-}} \]

• \( V_{R+} \) and \( V_{R-} \) are programmable voltage levels: the upper (\( V_{R+} \)) and lower limits (\( V_{R-} \)) of the conversion

• The digital output (\( N_{ADC} \)) is full scale
  – \( \text{0xFFF} \) when the input signal is equal to or higher than \( V_{R+} \)
  – \( \text{0x000} \) when the input signal is equal to or lower than \( V_{R-} \)
  – The input channel and the reference voltage levels (\( V_{R+} \) and \( V_{R-} \)) are defined in the conversion-control memory.
# ADC12CTL0  ADC Ctrl Register 0

<table>
<thead>
<tr>
<th></th>
<th>ADC12SHT1x</th>
<th>ADC12SHT0x</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12CTL0_H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESET</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>ADC12SHT1x: ADC12_B sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM8 to ADC12MEM23. Can be modified only when ADC12ENC = 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000b: 4 ADC12CLK cycles</td>
<td>0001b: 8 cycles, 0010b: 16, 0011b: 32</td>
<td></td>
</tr>
<tr>
<td>0100b: 64 cycles, 0101b: 96, 0110b: 128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111b: 192 cycles, 1000b: 256, 1001b: 384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010b: 512 ADC12CLK cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011b, 1101b, 1110b, 1111b: reserved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC12SHT0x: ADC12_B sample-and-hold time. These bits define the number of ADC12CLK cycles in the sampling period for registers ADC12MEM0 to ADC12MEM7 and ADC12MEM24 to ADC12MEM31. Can be modified only when ADC12ENC = 0.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0000b: 4 ADC12CLK cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001b: 8 cycles, 0010b: 16, 0011b: 32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100b: 64 cycles, 0101b: 96, 0110b: 128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111b: 192 cycles, 1000b: 256, 1001b: 384</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010b: 512 ADC12CLK cycles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011b, 1101b, 1110b, 1111b: reserved</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### ADC12CTL0  ADC Ctrl Register 0

<table>
<thead>
<tr>
<th>Address</th>
<th>Bit 15</th>
<th>Bit 14</th>
<th>Bit 13</th>
<th>Bit 12</th>
<th>Bit 11</th>
<th>Bit 10</th>
<th>Bit 9</th>
<th>Bit 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12SHT1x</td>
<td>ADC12SHT0x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC12MSC</td>
<td>reserved</td>
<td>ADC12ON</td>
<td>reserved</td>
<td>ADC12ENC</td>
<td>ADC12SC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ADC12MSC:** ADC12_B multiple sample and conversion. Valid only for sequence or repeated modes. Can be modified only when ADC12ENC = 0.

**1b** = The incidence of the first rising edge of the SHI signal triggers the sampling timer, but further sample-and-conversions are performed automatically as soon as the prior conversion is completed.

**0b** = The sampling timer requires a rising edge of the SHI signal to trigger each sample-and-convert.

**ADC12ON:** ADC12_B on. Can be modified only when ADC12ENC = 0.

0b = ADC12_B off
1b = ADC12_B on

**ADC12ENC:** ADC12_B enable conversion.

0b = ADC12_B disabled
1b = ADC12_B enabled

**ADC12SC:** ADC12_B start conversion. Software-controlled sample-and-conversion start. ADC12SC and ADC12ENC may be set together with one instruction. ADC12SC is reset automatically.

0b = No sample-and-conversion-start
1b = Start sample-and-conversion
## ADC12CTL0 control bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12SC</td>
<td>0x0001</td>
<td>ADC12 Start Conversion</td>
</tr>
<tr>
<td>ADC12ENC</td>
<td>0x0002</td>
<td>ADC12 Enable Conversion</td>
</tr>
<tr>
<td>ADC12ON</td>
<td>0x0010</td>
<td>ADC12 On/enable</td>
</tr>
<tr>
<td>ADC12MSC</td>
<td>0x0080</td>
<td>ADC12 Multiple SampleConversion</td>
</tr>
<tr>
<td>ADC12SHT00</td>
<td>0x0100</td>
<td>ADC12 Sample Hold 0 Select Bit: 0</td>
</tr>
<tr>
<td>ADC12SHT01</td>
<td>0x0200</td>
<td>ADC12 Sample Hold 0 Select Bit: 1</td>
</tr>
<tr>
<td>ADC12SHT02</td>
<td>0x0400</td>
<td>ADC12 Sample Hold 0 Select Bit: 2</td>
</tr>
<tr>
<td>ADC12SHT03</td>
<td>0x0800</td>
<td>ADC12 Sample Hold 0 Select Bit: 3</td>
</tr>
<tr>
<td>ADC12SHT10</td>
<td>0x1000</td>
<td>ADC12 Sample Hold 1 Select Bit: 0</td>
</tr>
<tr>
<td>ADC12SHT11</td>
<td>0x2000</td>
<td>ADC12 Sample Hold 1 Select Bit: 1</td>
</tr>
<tr>
<td>ADC12SHT12</td>
<td>0x4000</td>
<td>ADC12 Sample Hold 1 Select Bit: 2</td>
</tr>
<tr>
<td>ADC12SHT13</td>
<td>0x8000</td>
<td>ADC12 Sample Hold 1 Select Bit: 3</td>
</tr>
</tbody>
</table>
## ADC12CTL0 control bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12SHT0_0</td>
<td>0x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 0</td>
</tr>
<tr>
<td>ADC12SHT0_1</td>
<td>1x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 1</td>
</tr>
<tr>
<td>ADC12SHT0_2</td>
<td>2x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 2</td>
</tr>
<tr>
<td>ADC12SHT0_3</td>
<td>3x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 3</td>
</tr>
<tr>
<td>ADC12SHT0_4</td>
<td>4x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 4</td>
</tr>
<tr>
<td>ADC12SHT0_5</td>
<td>5x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 5</td>
</tr>
<tr>
<td>ADC12SHT0_6</td>
<td>6x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 6</td>
</tr>
<tr>
<td>ADC12SHT0_7</td>
<td>7x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 7</td>
</tr>
<tr>
<td>ADC12SHT0_8</td>
<td>8x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 8</td>
</tr>
<tr>
<td>ADC12SHT0_9</td>
<td>9x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 9</td>
</tr>
<tr>
<td>ADC12SHT0_10</td>
<td>10x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 10</td>
</tr>
<tr>
<td>ADC12SHT0_11</td>
<td>11x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 11</td>
</tr>
<tr>
<td>ADC12SHT0_12</td>
<td>12x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 12</td>
</tr>
<tr>
<td>ADC12SHT0_13</td>
<td>13x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 13</td>
</tr>
<tr>
<td>ADC12SHT0_14</td>
<td>14x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 14</td>
</tr>
<tr>
<td>ADC12SHT0_15</td>
<td>15x100u</td>
<td>ADC12 Sample Hold 0 Select Bit: 15</td>
</tr>
</tbody>
</table>

msp430fr5969.h

Week 6: Sensors
## ADC12CTL0 control bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12SHT1_0</td>
<td>0*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 0</td>
</tr>
<tr>
<td>ADC12SHT1_1</td>
<td>1*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 1</td>
</tr>
<tr>
<td>ADC12SHT1_2</td>
<td>2*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 2</td>
</tr>
<tr>
<td>ADC12SHT1_3</td>
<td>3*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 3</td>
</tr>
<tr>
<td>ADC12SHT1_4</td>
<td>4*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 4</td>
</tr>
<tr>
<td>ADC12SHT1_5</td>
<td>5*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 5</td>
</tr>
<tr>
<td>ADC12SHT1_6</td>
<td>6*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 6</td>
</tr>
<tr>
<td>ADC12SHT1_7</td>
<td>7*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 7</td>
</tr>
<tr>
<td>ADC12SHT1_8</td>
<td>8*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 8</td>
</tr>
<tr>
<td>ADC12SHT1_9</td>
<td>9*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 9</td>
</tr>
<tr>
<td>ADC12SHT1_10</td>
<td>10*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 10</td>
</tr>
<tr>
<td>ADC12SHT1_11</td>
<td>11*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 11</td>
</tr>
<tr>
<td>ADC12SHT1_12</td>
<td>12*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 12</td>
</tr>
<tr>
<td>ADC12SHT1_13</td>
<td>13*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 13</td>
</tr>
<tr>
<td>ADC12SHT1_14</td>
<td>14*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 14</td>
</tr>
<tr>
<td>ADC12SHT1_15</td>
<td>15*0x100u</td>
<td>ADC12 Sample Hold 1 Select Bit: 15</td>
</tr>
</tbody>
</table>

*msp430fr5969.h*
### ADC12CTL1  ADC Ctrl Register 1

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsvd</td>
<td>ADC12PDIV</td>
<td>ADC12SHSx</td>
<td>RTC AIFG</td>
<td>RTC RDYIFG</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### ADC12PDIV: ADC12_B predivider
- 00b = Predive by 1
- 01b = Predive by 4
- 10b = Predive by 32
- 11b = Predive by 64

#### ADC12SHSx: ADC12_B sample-and-hold source select (000b = ADC12SC bit)

#### ADC12SHP: ADC12_B sample-and-hold pulse-mode select
This bit selects the source of the sampling signal (SAMPCON) to be either the output of the sampling timer or the sample - input signal directly. 0b = SAMPCON signal is sourced from the sample-input signal. 1b = SAMPCON signal is sourced from the sampling timer.

#### ADC12ISSH: ADC12_B invert signal sample-and-hold
- 0b = The sample-input signal is not inverted.
- 1b = The sample-input signal is inverted.

#### ADC12DIVx: ADC12_B clock divider
- 000b = /1, 001b = /2, 010b = /3
- 011b = /4, 100b = /5, 101b = /6
- 110b = /7, 111b = /8

#### ADC12SSELx: ADC12_B clock source select
- 00b = ADC12OSC (MODOSC), 01b = ACLK, 10b = MCLK, 11b = SMCLK
ADC12CTL1  ADC Ctrl Register 1

<table>
<thead>
<tr>
<th>ADC12CTL0_H</th>
<th>ADC12PDIV</th>
<th>ADC12SHSx</th>
<th>RTC AIFG</th>
<th>RTC RDYIFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>rsvd</td>
<td>12</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADC12CTL0_L</th>
<th>ADC12DIVx</th>
<th>ADC12SSELx</th>
<th>ADC12CONSEQx</th>
<th>ADC12BUSY</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td>6</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**ADC12CONSEQx**: ADC12_B conversion sequence mode select. This bit should only be modified when ADC12ENC = 0 except to stop a conversion immediately by setting ADC12CONSEQx = 00 when ADC12ENC = 1.

- 00b = Single-channel, single-conversion
- 01b = Sequence-of-channels
- 10b = Repeat-single-channel
- 11b = Repeat-sequence-of-channels

**ADC12BUSY**: ADC12_B busy. This bit indicates an active sample or conversion operation.

- 0b = No operation is active.
- 1b = A sequence, sample, or conversion is active.
# ADC12CTL1 control bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12BUSY</td>
<td>0x0001</td>
<td>ADC12 Busy</td>
</tr>
<tr>
<td>ADC12CONSEQ0</td>
<td>0x0002</td>
<td>ADC12 Conversion Sequence Select Bit: 0</td>
</tr>
<tr>
<td>ADC12CONSEQ1</td>
<td>0x0004</td>
<td>ADC12 Conversion Sequence Select Bit: 1</td>
</tr>
<tr>
<td>ADC12SSEL0</td>
<td>0x0008</td>
<td>ADC12 Clock Source Select Bit: 0</td>
</tr>
<tr>
<td>ADC12SSEL1</td>
<td>0x0010</td>
<td>ADC12 Clock Source Select Bit: 1</td>
</tr>
<tr>
<td>ADC12DIV0</td>
<td>0x0020</td>
<td>ADC12 Clock Divider Select Bit: 0</td>
</tr>
<tr>
<td>ADC12DIV1</td>
<td>0x0040</td>
<td>ADC12 Clock Divider Select Bit: 1</td>
</tr>
<tr>
<td>ADC12DIV2</td>
<td>0x0080</td>
<td>ADC12 Clock Divider Select Bit: 2</td>
</tr>
<tr>
<td>ADC12ISSH</td>
<td>0x0100</td>
<td>ADC12 Invert Sample Hold Signal</td>
</tr>
<tr>
<td>ADC12SHP</td>
<td>0x0200</td>
<td>ADC12 Sample/Hold Pulse Mode</td>
</tr>
<tr>
<td>ADC12SHS0</td>
<td>0x0400</td>
<td>ADC12 Sample/Hold Source Bit: 0</td>
</tr>
<tr>
<td>ADC12SHS1</td>
<td>0x0800</td>
<td>ADC12 Sample/Hold Source Bit: 1</td>
</tr>
<tr>
<td>ADC12SHS2</td>
<td>0x1000</td>
<td>ADC12 Sample/Hold Source Bit: 2</td>
</tr>
<tr>
<td>ADC12PDIV0</td>
<td>0x2000</td>
<td>ADC12 Predivider Bit: 0</td>
</tr>
<tr>
<td>ADC12PDIV1</td>
<td>0x4000</td>
<td>ADC12 Predivider Bit: 1</td>
</tr>
</tbody>
</table>
**ADC12CTL2**  
**ADC Ctrl Register 2**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
</table>
| ADC12RES | ADC12_B resolution. This bit defines the conversion result resolution. This bit should only be modified when ADC12ENC=0.  
00b = 8 bit (10 clock cycle conv. time)  
01b = 10 bit (12 clock cycle conv. time)  
10b = 12 bit (14 clock cycle conv. time)  
11b = Reserved |
| ADC12DF  | ADC12_B data read-back format. Data is always stored in the binary unsigned format.  
0b: binary unsigned  
1b: signed binary (2s complement) |
| ADC12PWRMD | Enables ADC low-power mode for ADC12CLK  
0b = Regular power mode (sample rate not restricted)  
1b = Low power mode enable |
### ADC12CTL2 control bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12PWRMD</td>
<td>0x0001</td>
<td>ADC12 Power Mode</td>
</tr>
<tr>
<td>ADC12DF</td>
<td>0x0008</td>
<td>ADC12 Data format</td>
</tr>
<tr>
<td>ADC12RES0</td>
<td>0x0010</td>
<td>ADC12 Resolution Bit: 0</td>
</tr>
<tr>
<td>ADC12RES1</td>
<td>0x0020</td>
<td>ADC12 Resolution Bit: 1</td>
</tr>
<tr>
<td>ADC12RES__8BIT</td>
<td>0x0000</td>
<td>ADC12+ Resolution : 8 Bit</td>
</tr>
<tr>
<td>ADC12RES__10BIT</td>
<td>0x0010</td>
<td>ADC12+ Resolution : 10 Bit</td>
</tr>
<tr>
<td>ADC12RES__12BIT</td>
<td>0x0020</td>
<td>ADC12+ Resolution : 12 Bit</td>
</tr>
<tr>
<td>ADC12RES_0</td>
<td>0x0000</td>
<td>ADC12+ Resolution : 8 Bit</td>
</tr>
<tr>
<td>ADC12RES_1</td>
<td>0x0010</td>
<td>ADC12+ Resolution : 10 Bit</td>
</tr>
<tr>
<td>ADC12RES_2</td>
<td>0x0020</td>
<td>ADC12+ Resolution : 12 Bit</td>
</tr>
<tr>
<td>ADC12RES_3</td>
<td>0x0400</td>
<td>ADC12+ Resolution : reserved</td>
</tr>
</tbody>
</table>
### ADC12CTL3 ADC Ctrl Register 3

<table>
<thead>
<tr>
<th>ADCCTL3_H</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12ICH3MAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC12ICH2MAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC12ICH1MAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADC12ICH0MAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

ADC12ICH3MAP: Controls internal channel 3 selection to ADC input channel A26. Can be modified only when ADC12ENC = 0.
0b = external pin selected for ADC ch26
1b = ADC input ch. internal 3 is selected for ADC input channel A26

ADC12ICH2MAP: Controls internal channel 2 selection to ADC input channel A27. Can be modified only when ADC12ENC = 0.
0b = external pin selected for ADC ch27
1b = ADC input ch. Internal 2 is selected for ADC input channel A27

<table>
<thead>
<tr>
<th>ADCCTL3_L</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12TCMAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ADC12CSTARTADDx</td>
</tr>
<tr>
<td>ADC12BATMAP</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

ADC12ICH1MAP: Controls internal channel 1 selection to ADC input channel A28. Can be modified only when ADC12ENC = 0.
0b = external pin selected for ADC ch28
1b = ADC input ch. Internal 1 is selected for ADC input channel A28

ADC12ICH0MAP: Controls internal channel 0 selection to ADC input channel A29. Can be modified only when ADC12ENC = 0.
0b = external pin selected for ADC ch29
1b = ADC input ch. Internal 0 is selected for ADC input channel A29
### ADC12CTL3  ADC Ctrl Register 3

<table>
<thead>
<tr>
<th>ADCCTL3_H</th>
<th></th>
<th>ADC12</th>
<th>ADC12</th>
<th>ADC12</th>
<th>ADC12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>RESET</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>R-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADCCTL3_L</th>
<th>ADC12</th>
<th>ADC12</th>
<th>ADC12</th>
<th>ADC12</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TCMAP</td>
<td>BATMAP</td>
<td>rsvd</td>
<td>CSTARTADDx</td>
</tr>
<tr>
<td>RESET</td>
<td>RW-0</td>
<td>RW-0</td>
<td>R-0</td>
<td>RW-0</td>
</tr>
<tr>
<td></td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td></td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

**ADC12TCMAP**: Controls temperature sensor ADC input channel selection. Can be modified only when ADC12ENC = 0.
- 0b = external pin selected for ADC input ch. A30
- 1b = ADC int. tempera. sensor ch. is selected for ADC input channel A30

**ADC12BATMAP**: Controls 1/2 AVCC ADC input channel selection. Can be modified only when ADC12ENC = 0.
- 0b = external pin is selected for ADC input channel A31
- 1b = ADC internal 1/2 x AVCC channel is selected for ADC input channel A31

**ADC12CSTARTADDx**: ADC12_B conversion start address. These bits select which ADC12_B conversion memory register is used for a single conversion or for the first conversion in a sequence. The value of CSTARTADDx is 0h to 1Fh, corresponding to ADC12MEM0 to ADC12MEM31. Can be modified only when ADC12ENC = 0.

MSP430FR5969 has an internal temperature sensor ch. A30
If ADC12DF = 0: The 12-bit conversion results are right justified. Bit 11 is the MSB. Bits 15-12 are 0 in 12-bit mode, bits 15-10 are 0 in 10-bit mode, and bits 15-8 are 0 in 8-bit mode. If the user writes to the conversion memory registers, the results are corrupted.

If ADC12DF = 1: The 12-bit conversion results are left-justified 2s-complement format. Bit 15 is the MSB. Bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode. The data is stored in the right-justified format and is converted to the left-justified 2s-complement format during read back. If the user writes to the conversion memory registers, the results are corrupted.
ADC12MCTLx  Conv. Mem. Ctl

<table>
<thead>
<tr>
<th>RSVD</th>
<th>ADC12 WINC</th>
<th>ADC12 DIF</th>
<th>RSVD</th>
<th>ADC12 EOS</th>
<th>ADC12RSEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0b</td>
<td>comp. wind. disabled.</td>
<td>comparetor wind. enabled.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>single-ended mode enabled.</td>
<td>differential mode enabled.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0b</td>
<td>not end of sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1b</td>
<td>end of sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ADC12WINC: Comparator window enable. Can be modified only when ADC12ENC=0.

ADC12DIF: Differential mode. Can be modified only when ADC12ENC = 0.

ADC12EOS: End of sequence. Indicates the last conversion in a sequence. Can be modified only when ADC12ENC = 0.

ADC12VRSEL : Selects combinations of VR+ and VR-sources as well as the buffer selection. Note: there is only one buffer so it can be used for either VR+ or VR-, but not both. Can be modified only when ADC12ENC = 0.

0000b: VR+ = AVCC, VR-=AVSS  
0001b: VR+=VREF buffered, VR-=AVSS  
0010b: VR+=VeRef-, VR-=AVSS  
0011b: VR+=VeREF+buffered,VR-=AVSS  
0100b: VR+= VeREF+, VR- = AVSS  
0101b: VR+= AVCC, VR-=VeREF+ buffered  
0110b: VR+= AVCC, VR- = VeREF+  
0111b: VR+=VREF buffered,VR- = VeREF+  

............
ADC12INCHx: Input channel select. If even channels are set as differential, then odd channel configuration is ignored. Can be modified only when ADC12ENC = 0

<table>
<thead>
<tr>
<th>ADC12EOS</th>
<th>rsvd</th>
<th>ADC12RSEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>reset</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADC12RSEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000b: If ADC12DIF = 0: A0; If ADC12DIF = 1: Ain+ = A0, Ain- = A1</td>
</tr>
<tr>
<td>00001b: If ADC12DIF = 0: A1; If ADC12DIF = 1: Ain+ = A0, Ain- = A1</td>
</tr>
<tr>
<td>00010b: If ADC12DIF = 0: A2; If ADC12DIF = 1: Ain+ = A2, Ain- = A3</td>
</tr>
<tr>
<td>00011b: If ADC12DIF = 0: A3; If ADC12DIF = 1: Ain+ = A2, Ain- = A3</td>
</tr>
<tr>
<td>00100b: If ADC12DIF = 0: A4; If ADC12DIF = 1: Ain+ = A4, Ain- = A5</td>
</tr>
<tr>
<td>00101b: If ADC12DIF = 0: A5; If ADC12DIF = 1: Ain+ = A4, Ain- = A5</td>
</tr>
<tr>
<td>00110b: If ADC12DIF = 0: A6; If ADC12DIF = 1: Ain+ = A6, Ain- = A7</td>
</tr>
<tr>
<td>00111b: If ADC12DIF = 0: A7; If ADC12DIF = 1: Ain+ = A6, Ain- = A7</td>
</tr>
<tr>
<td>01000b: If ADC12DIF = 0: A8; If ADC12DIF = 1: Ain+ = A8, Ain- = A9</td>
</tr>
<tr>
<td>01001b: If ADC12DIF = 0: A9; If ADC12DIF = 1: Ain+ = A8, Ain- = A9</td>
</tr>
<tr>
<td>01010b: If ADC12DIF = 0: A10; If ADC12DIF = 1: Ain+ = A10, Ain- = A11</td>
</tr>
</tbody>
</table>

... ... .................................. . ... ...
# ADC12MCTLx control bits

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12INCH0</td>
<td>0x0001</td>
<td>ADC12 Input Channel Select Bit 0</td>
</tr>
<tr>
<td>ADC12INCH1</td>
<td>0x0002</td>
<td>ADC12 Input Channel Select Bit 1</td>
</tr>
<tr>
<td>ADC12INCH2</td>
<td>0x0004</td>
<td>ADC12 Input Channel Select Bit 2</td>
</tr>
<tr>
<td>ADC12INCH3</td>
<td>0x0008</td>
<td>ADC12 Input Channel Select Bit 3</td>
</tr>
<tr>
<td>ADC12INCH4</td>
<td>0x0010</td>
<td>ADC12 Input Channel Select Bit 4</td>
</tr>
<tr>
<td>ADC12EOS</td>
<td>0x0080</td>
<td>ADC12 End of Sequence</td>
</tr>
<tr>
<td>ADC12VRSEL0</td>
<td>0x0100</td>
<td>ADC12 VR Select Bit 0</td>
</tr>
<tr>
<td>ADC12VRSEL1</td>
<td>0x0200</td>
<td>ADC12 VR Select Bit 1</td>
</tr>
<tr>
<td>ADC12VRSEL2</td>
<td>0x0400</td>
<td>ADC12 VR Select Bit 2</td>
</tr>
<tr>
<td>ADC12VRSEL3</td>
<td>0x0800</td>
<td>ADC12 VR Select Bit 3</td>
</tr>
<tr>
<td>ADC12DIF</td>
<td>0x2000</td>
<td>ADC12 Differential mode (only for even regs)</td>
</tr>
<tr>
<td>ADC12WINC</td>
<td>0x4000</td>
<td>ADC12 Comparator window enable</td>
</tr>
<tr>
<td>ADC12INCH_n</td>
<td>n</td>
<td>ADC12 Input Channel n</td>
</tr>
</tbody>
</table>
ADC12HI  Window Comparator High Threshold Register

High threshold: Window comparator high threshold should only be modified when ADC12ENC=0.

If ADC12DF = 0: The 12-bit threshold value is right justified when ADC12DF = 0. Bits 15-12 are 0. Bit 11 is the MSB. Bits 11-10 are 0 in 10-bit mode, and bits 11-8 are 0 in 8-bit mode.

If ADC12DF = 1: The 12-bit threshold value is left justified when ADC12DF = 1, 2s-complement format. Bit 15 is the MSB. Bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.
Low threshold: Window comparator low threshold should only be modified when ADC12ENC=0.

If ADC12DF = 0: The 12-bit threshold value is right justified when ADC12DF = 0. Bits 15-12 are 0. Bit 11 is the MSB. Bits 11-10 are 0 in 10-bit mode, and bits 11-8 are 0 in 8-bit mode.

If ADC12DF = 1: The 12-bit threshold value is left justified when ADC12DF = 1, 2s-complement format. Bit 15 is the MSB. Bits 3-0 are 0 in 12-bit mode, bits 5-0 are 0 in 10-bit mode, and bits 7-0 are 0 in 8-bit mode.
**ADC12IER0  Interrupt Enable Reg. 0**

<table>
<thead>
<tr>
<th>Bit 15 (IE15)</th>
<th>Bit 14 (IE14)</th>
<th>Bit 13 (IE13)</th>
<th>Bit 12 (IE12)</th>
<th>Bit 11 (IE11)</th>
<th>Bit 10 (IE10)</th>
<th>Bit 9 (IE9)</th>
<th>Bit 8 (IE8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12</td>
<td>ADC12</td>
<td>ADC12</td>
<td>ADC12</td>
<td>ADC12</td>
<td>ADC12</td>
<td>ADC12</td>
<td>ADC12</td>
</tr>
</tbody>
</table>

**RESET**

<table>
<thead>
<tr>
<th>Bit 15 (IE15)</th>
<th>Bit 14 (IE14)</th>
<th>Bit 13 (IE13)</th>
<th>Bit 12 (IE12)</th>
<th>Bit 11 (IE11)</th>
<th>Bit 10 (IE10)</th>
<th>Bit 9 (IE9)</th>
<th>Bit 8 (IE8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

**ADC12IEx: Interrupt enable**

- **0b**: interrupt disabled
- **1b**: interrupt enabled.

**Msp430fr5969.h**

**ADC12IEn**  interrupt enable for channel n
## ADC12IER1  Interrupt Enable Reg. 1

<table>
<thead>
<tr>
<th></th>
<th>ADC12 IE31</th>
<th>ADC12 IE30</th>
<th>ADC12 IE29</th>
<th>ADC12 IE28</th>
<th>ADC12 IE27</th>
<th>ADC12 IE26</th>
<th>ADC12 IE25</th>
<th>ADC12 IE24</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>6</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>5</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>4</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>3</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>2</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>1</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

**RESET**

<table>
<thead>
<tr>
<th></th>
<th>ADC12 IE23</th>
<th>ADC12 IE22</th>
<th>ADC12 IE21</th>
<th>ADC12 IE20</th>
<th>ADC12 IE19</th>
<th>ADC12 IE18</th>
<th>ADC12 IE17</th>
<th>ADC12 IE16</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>6</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>5</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>4</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>3</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>2</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>1</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

**ADC12IEx: Interrupt enable**
- 0b: interrupt disabled
- 1b: interrupt enabled.
## ADC12IER0-1  Interrupt Enable

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12IE0</td>
<td>0x0001</td>
<td>ADC12 Memory 0 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE1</td>
<td>0x0002</td>
<td>ADC12 Memory 1 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE2</td>
<td>0x0004</td>
<td>ADC12 Memory 2 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE3</td>
<td>0x0008</td>
<td>ADC12 Memory 3 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE4</td>
<td>0x0010</td>
<td>ADC12 Memory 4 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE5</td>
<td>0x0020</td>
<td>ADC12 Memory 5 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE6</td>
<td>0x0040</td>
<td>ADC12 Memory 6 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12IE7</td>
<td>0x0080</td>
<td>ADC12 Memory 7 Interrupt Enable</td>
</tr>
<tr>
<td>........</td>
<td>.......</td>
<td>........</td>
</tr>
<tr>
<td>ADC12I28</td>
<td>0x1000</td>
<td>ADC12 Memory 28 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12I29</td>
<td>0x2000</td>
<td>ADC12 Memory 29 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12I30</td>
<td>0x4000</td>
<td>ADC12 Memory 30 Interrupt Enable</td>
</tr>
<tr>
<td>ADC12I31</td>
<td>0x8000</td>
<td>ADC12 Memory 31 Interrupt Enable</td>
</tr>
</tbody>
</table>

Week 6: Sensors
### ADC12IER2  Interrupt Enable Reg. 2

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
<td>R0</td>
</tr>
</tbody>
</table>

**RESET**

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>R0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>R0</td>
</tr>
</tbody>
</table>

**Value:**

- **ADC12RDYIE:** ADC12_B local reference buffer ready interrupt enable. (0b: interrupt disabled, 1b: interrupt enabled.)
- **ADC12TOVIE:** ADC12_B conversion-time-overflow interrupt enable. (0b: interrupt disabled, 1b: interrupt enabled.)
- **ADC12OVIE:** ADC12MEMx overflow-interrupt enable. (0b: interrupt disabled, 1b: interrupt enabled.)
- **ADC12HIIE:** Interrupt enable for the exceeding the upper limit interrupt of the window comparator for ADC12MEMx result register. The (0b: interrupt disabled, 1b: interrupt enabled.)
- **ADC12LOIE:** Interrupt enable for the falling short of the lower limit interrupt of the window comparator for the ADC12MEMx result register. (0b: interrupt disabled, 1b: interrupt enabled.)
- **ADC12INIE:** Interrupt enable for the ADC12MEMx result register being greater than the ADC12LO threshold and below the ADC12HI threshold. (0b: interrupt disabled, 1b: interrupt enabled.)
### ADC12IFG0 - Interrupt Flag Reg. 0

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12 IFG15</td>
<td>ADC12 IFG14</td>
<td>ADC12 IFG13</td>
<td>ADC12 IFG12</td>
<td>ADC12 IFG11</td>
<td>ADC12 IFG10</td>
<td>ADC12 IFG9</td>
<td>ADC12 IFG8</td>
</tr>
</tbody>
</table>

**RESET**
- Bit 7: RW-0
- Bit 6: RW-0
- Bit 5: RW-0
- Bit 4: RW-0
- Bit 3: RW-0
- Bit 2: RW-0
- Bit 1: RW-0
- Bit 0: RW-0

ADC12IFGn: (n=0..15) ADC12MEMn interrupt flag. This bit is set when ADC12MEMn is loaded with a conversion result. The ADC12IFGn bit is reset if ADC12MEMn is accessed, or it can be reset with software.

- **0b** = No interrupt pending
- **1b** = Interrupt pending
### ADC12IFG1: Interrupt Flag Reg. 1

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12 IFG31</td>
<td>ADC12 IFG30</td>
<td>ADC12 IFG29</td>
<td>ADC12 IFG28</td>
<td>ADC12 IFG27</td>
<td>ADC12 IFG26</td>
<td>ADC12 IFG25</td>
<td>ADC12 IFG24</td>
</tr>
</tbody>
</table>

**RESET**

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
</tbody>
</table>

The ADC12IFGn: (n=16..31) ADC12MEMn interrupt flag. This bit is set when ADC12MEMn is loaded with a conversion result. The ADC12IFGn bit is reset if ADC12MEMn is accessed, or it can be reset with software.

- **0b** = No interrupt pending
- **1b** = Interrupt pending
ADC12IFGR2  Interrupt Flag Reg. 2

<table>
<thead>
<tr>
<th></th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>RES</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
<td>RW-0</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
| reserved | ADC12 RDYIFG | ADC12 TOVIFG | ADC12 OVIFG | ADC12 HIIFG | ADC12 LOIFG | ADC12 INIFG | ADC12 IFG16

ADC12RDYIFG: ADC12_B local reference buffer ready interrupt flag. (0b = No interrupt pending, 1b = Interrupt pending)
ADC12TOVIFG: ADC12_B conversion-time-overflow interrupt flag. (0b = No interrupt pending, 1b = Interrupt pending)
ADC12OVIFG: ADC12MEMx overflow-interrupt flag. (0b = No interrupt pending, 1b = Interrupt pending)
ADC12HIIFG: Interrupt flag for exceeding the upper limit interrupt of the window comparator for ADC12MEMx result register. (0b = No interrupt pending, 1b = Interrupt pending)
ADC12LOIFG: Interrupt flag for falling short of the lower limit interrupt of the window comparator for the ADC12MEMx result register. (0b = No interrupt pending, 1b = Interrupt pending)
ADC12INIFG: Interrupt flag for the ADC12MEMx result register being greater than the ADC12LO threshold and below the ADC12HI threshold interrupt. (0b = No interrupt pending, 1b = Interrupt pending)
ADC12_B | Window Comparator

- Window Comparator
  - Allows you to configure input threshold levels
  - The ADC conversion results are automatically compared against the thresholds
  - Hi, Lo, and In interrupts indicate which range the result falls in
  - Same thresholds shared among all channels
  - Useful for low power because device can stay in LPM until result falls in window

<table>
<thead>
<tr>
<th>ADC12HI threshold</th>
<th>Set ADC12HIIFG</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12LO threshold</td>
<td>Set ADC12LOIFG</td>
</tr>
<tr>
<td>ADC12IN threshold</td>
<td>Set ADC12INIFG</td>
</tr>
</tbody>
</table>
ADC12_B | Window Comparator Example

ADC12HI threshold

ADC12LO threshold

Set
ADC12INIFG
ADC12_B | Window Comparator Example

Set ADC12HIIFG

ADC12HI threshold

ADC12LO threshold
ADC12_B | Window Comparator Example

Set
ADC12INIFG

ADC12HI threshold

ADC12LO threshold
ADC12_B | Window Comparator Example

ADC12HI threshold

ADC12LO threshold

Set ADC12LOIFG
ADC12_B | Window Comparator Example

ADC12HI threshold

ADC12LO threshold

Set ADC12LOIFG
ADC12_B | Window Comparator Example

- Set ADC12HIIFG
- Set ADC12INIFG
- Set ADC12LOIFG

ADC12HI threshold

ADC12LO threshold
ADC12_B | Differential Mode

- Combine 2 input channels to create a differential input channel
- In this mode the ADC will measure the difference between two channels and store this value in the ADC12MEMx register

![Diagram showing differential mode with inputs A1 and A2, and output Difference to ADC12MEMx Register.](image)
ADC12_B | Int. Channel Mapping

- A26-31 can map to either an External Input or Internal ADC input
- Internal inputs include the temperature sensor and battery monitor
**ADC12IV**  **Interrupt Vector**

In the interrupt service routine, **ADC12IV** register is used to find out the source of the pending interrupt.

<table>
<thead>
<tr>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12IVx</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**RESET**

<table>
<thead>
<tr>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
<th>RW-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**ADC12IVx**

**ADC12IVx: Interrupt vector value.** Writing to this register clears all pending interrupt flags.

- **000h:** no interrupt pending
- **002h:** ADC12MEMx overflow, ADC12OVIFG
- **004h:** Conversion time overflow, ADC12TOVIFG
- **006h:** wind. high interr. flag, ADC12HIIFG
- **008h:** window low interrupt flag, ADC12LOIFG
- **00Ah:** n-window interrupt flag, ADC12INIFG
- **00Ch:** ADC12MEM0 intr. Flag. ADC12IFG0
- **00Eh:** ADC12MEM1 intr. Flag. ADC12IFG1
- **010h:** ADC12MEM2 intr. Flag. ADC12IFG2
- **020h:** ADC12MEM3 intr. Flag. ADC12IFG3
- **040h:** ADC12MEM4 intr. Flag. ADC12IFG4
- **042h:** ADC12MEM27 intr. Flag. ADC12IFG27
- **044h:** ADC12MEM28 intr. Flag. ADC12IFG28
- **046h:** ADC12MEM29 intr. Flag. ADC12IFG29
- **048h:** ADC12MEM30 intr. Flag. ADC12IFG30
- **04Ah:** ADC12MEM31 intr. Flag. ADC12IFG31
- **04Ch:** ADC12RDYIFG intr. Flag.
# ADC12IV Interrupt Vector

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit Value</th>
<th>Interrupt Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC12IV_NONE</td>
<td>0x0000</td>
<td>No interrupt pending</td>
</tr>
<tr>
<td>ADC12IV_ADC12OVIFG</td>
<td>0x0002</td>
<td>ADC12OVIFG</td>
</tr>
<tr>
<td>ADC12IV_ADC12TOVIFG</td>
<td>0x0004</td>
<td>ADC12TOVIFG</td>
</tr>
<tr>
<td>ADC12IV_ADC12HIIFG</td>
<td>0x0006</td>
<td>ADC12HIIFG</td>
</tr>
<tr>
<td>ADC12IV_ADC12LOIFG</td>
<td>0x0008</td>
<td>ADC12LOIFG</td>
</tr>
<tr>
<td>ADC12IV_ADC12INIFG</td>
<td>0x000A</td>
<td>ADC12INIFG</td>
</tr>
<tr>
<td>ADC12IV_ADC12IFG0</td>
<td>0x000C</td>
<td>ADC12IFG0</td>
</tr>
<tr>
<td>ADC12IV_ADC12IFG1</td>
<td>0x000E</td>
<td>ADC12IFG1</td>
</tr>
<tr>
<td>........</td>
<td>..</td>
<td>........</td>
</tr>
<tr>
<td>ADC12IV_ADC12IFG29</td>
<td>0x0046</td>
<td>ADC12IFG29</td>
</tr>
<tr>
<td>ADC12IV_ADC12IFG30</td>
<td>0x0048</td>
<td>ADC12IFG30</td>
</tr>
<tr>
<td>ADC12IV_ADC12IFG31</td>
<td>0x004A</td>
<td>ADC12IFG31</td>
</tr>
<tr>
<td>ADC12IV_ADC12RDYIFG</td>
<td>0x004C</td>
<td>ADC12RDYIFG</td>
</tr>
</tbody>
</table>
Example

- A single sample is made on A1 with reference to Avcc (Analog supply voltage).
- Software sets \texttt{ADC12SC} to start sample and conversion - \texttt{ADC12SC} automatically cleared at EOC. ADC12 internal oscillator times sample (16x) and conversion.
- In Mainloop MSP430 waits in LPM0 to save power until ADC12 conversion complete, \texttt{ADC12_ISR} will force exit from LPM0 in Mainloop on \texttt{reti}.
- If input ADC voltage is greater than \texttt{0x7FF} (i.e. \texttt{A1 > 0.5*AV_{cc}}), P1.0 (LED) set, else reset. Show the correct handling of and ADC12 interrupt as well.
#include <msp430.h>
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD; // Stop WDT
    P1OUT &= ~BIT0;          // Clear LED to start
    P1DIR |= BIT0;           // Set P1.0/LED to output
    P1SEL1 |= BIT1;          // Configure P1.1 for ADC
    P1SEL0 |= BIT1;
    PM5CTL0 &= ~LOCKLPM5;   // enable GPIO

    // Configure ADC12
    ADC12CTL0 = ADC12SHT01 | ADC12ON; // Sampling time, S&H=16, ADC12 on
    ADC12CTL1 = ADC12SHP;          // Use sampling timer
    ADC12CTL2 |= ADC12RES__12BIT; // 12-bit conversion results
    ADC12MCTL0 |= ADC12INCH_1;   // A1 ADC input select; Vref=AVCC
    ADC12IER0 |= ADC12IE0;        // Enable ADC conv complete intr

    while (1) {
        __delay_cycles(5000); // delay 5000 clock cycles
        ADC12CTL0 |= ADC12ENC | ADC12SC; // Start sampling/conversion

        __bis_SR_register(LPM0_bits | GIE); // LPM0, ADC12_ISR will force exit
        __no_operation(); // For debugger
    }
}
Solution

```c
#pragma vector = ADC12_VECTOR
__interrupt void ADC12_ISR(void)
{
  switch(ADC12IV)
  {
    case ADC12IV_NONE:        break;        // Vector 0: No interrupt
    case ADC12IV_ADC12IFG0:                 // Vector 12: ADC12MEM0 intr
      if (ADC12MEM0 >= 0x7ff)               // ADC12MEM0=A1 > 0.5AVcc?
        P1OUT |= BIT0;                      // P1.0 = 1
      else
        P1OUT &= ~BIT0;                     // P1.0 = 0
      __bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
      break;                                // Clear CPUOFF bit from 0(SR)
    case ADC12IV_ADC12RDYIFG: break;        // Vector 76: ADC12RDY
    case ADC12IV_ADC12OVIFG:  break;        // Vector 2: ADC12MEMx ovf
    case ADC12IV_ADC12TOVIFG: break;        // Vector 4: Conv time ovf
    case ADC12IV_ADC12HIIFG:  break;        // Vector 6: ADC12BHI
    case ADC12IV_ADC12LOIFG:  break;        // Vector 8: ADC12BLO
    case ADC12IV_ADC12INIFG:  break;        // Vector 10: ADC12BIN
    case ADC12IV_ADC12IFG1:   break;        // Vector 14: ADC12MEM1

    // ... Put cases for all ADC12IV_ADC12IFG8..... ADC12IV_ADC12IFG31
    default: break;
  }
}
```
MSP430 REF_A

• The reference module (REF) is responsible for generation of all critical reference voltages that can be used by various analog peripherals in a given device.
• The heart of the reference system is the bandgap from which all other references are derived by unity or noninverting gain stages.
• The REFGEN subsystem consists of the bandgap, the bandgap bias, and the noninverting buffer stage, which generates the three primary voltage reference available in the system (1.2 V, 2.0 V, and 2.5 V).
• In addition, when enabled, a buffered bandgap voltage is available.
### REFCTRL0  REF_A Control Reg

<table>
<thead>
<tr>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>reserved</td>
<td>REF BGRDY</td>
<td>REF GENRDY</td>
<td>BGMODE</td>
<td>REFGEN BUSY</td>
<td>REF BGACT</td>
<td>REF GENACT</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RESET</th>
<th>R0</th>
<th>R0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
<th>R-0</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REF BGOT</th>
<th>REF GENOT</th>
<th>REFVSEL</th>
<th>REF TCOFF</th>
<th>rsvd</th>
<th>REFOUT</th>
<th>REFON</th>
</tr>
</thead>
</table>

**REFVSEL**: Reference voltage level select. Can be modified only when REFGENBUSY = 0.
- 00b = 1.2 V available when REFON = 1
- 01b = 2.0 V available when REFON = 1
- 10b = 2.5 V available when REFON = 1
- 11b = 2.5 V available when REFON = 1

**REFON**: Reference enable. Can be modified only when REFGENBUSY = 0.
- 0b = Disables reference if no other reference requests are pending
- 1b = Enables reference

This register is used to generate reference voltage for the ADC system. Once we start generation of the reference voltage, we need to check whether the reference voltage is stabilized or not by REFGENBUSY flag.
EXAMPLE

Using Internal Temperature Sensor

msp430fr59xx_adc12_10.c
ADC Example

- MSP430FR5969 has an internal temperature sensor connected to channel A30.
- We are going to use ADC12 for converting internal temperature sensor value to digital.
- We need to use the calibration data to correct the internal temperature sensor.
#include <msp430.h>
#define CALADC12_12V_30C *((unsigned int*)0x1A1A) // Temperature Sensor Calibration-30 C
#define CALADC12_12V_85C *((unsigned int*)0x1A1C) // Temperature Sensor Calibration-85 C

unsigned int temp;
volatile float temperatureDegC;

int main(void)
{
    WDTCTL = WDTPW + WDTHOLD; // Stop WDT

    // Initialize the shared reference module
    // By default, REFMSTR=1 => REFCTL is used to configure the internal reference
    while(REFCTL0 & REFGENBUSY); // If ref generator busy, WAIT
    REFCTL0 |= REFVSEL_0 + REFON; // Enable internal 1.2V reference

    /* Initialize ADC12_A */
    ADC12CTL0 &= ~ADC12ENC; // Disable ADC12
    ADC12CTL0 = ADC12SHT0_8 + ADC12ON; // Set sample time (8 clock cycles)
    ADC12CTL1 = ADC12SHP; // Enable sample timer
    ADC12CTL3 = ADC12TCMAP; // Enable internal temperature sensor
    ADC12MCTL0 = ADC12VRSEL_1 + ADC12INCH_30; // ADC input ch A30 => internal temperature sensor
    ADC12IER0 = 0x001; // ADC_IFG upon conv result-ADCMEMO
    while(!(REFCTL0 & REFGENRDY)); // Wait for reference generator to settle
    ADC12CTL0 |= ADC12ENC;

    while(1) {
        ADC12CTL0 |= ADC12SC; // Sampling and conversion start
        __bis_SR_register(LPM0_bits + GIE); // LPM0 with interrupts enabled
        __no_operation();

        // Temperature in Celsius.
        temperatureDegC = (float)(((long)temp - CALADC12_12V_30C) * (85 - 30)) / 
        (CALADC12_12V_85C - CALADC12_12V_30C) + 30.0f;
        __no_operation(); // SET BREAKPOINT HERE
    }
}
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void)
{
    switch(ADC12IV)
    {
        case ADC12IV_NONE:        break;        // Vector 0: No interrupt
        case ADC12IV_ADC12OVIFG:  break;        // Vector 2: ADC12MEMx Overflow
        case ADC12IV_ADC12TOVIFG: break;        // Vector 4: Conversion time ovf
        case ADC12IV_ADC12HIIFG:  break;        // Vector 6: ADC12BHI
        case ADC12IV_ADC12LOIFG:  break;        // Vector 8: ADC12BLO
        case ADC12IV_ADC12INIFG:  break;        // Vector 10: ADC12BIN
        case ADC12IV_ADC12IFG0:   break;        // Vector 12: ADC12MEM0 Interrupt
            temp = ADC12MEM0;                     // Move results, IFG is cleared
            __bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
            break;
        case ADC12IV_ADC12IFG1:   break;        // Vector 14: ADC12MEM1
        case ADC12IV_ADC12IFG2:   break;        // Vector 16: ADC12MEM2
        case ADC12IV_ADC12IFG3:   break;        // Vector 18: ADC12MEM3
        case ADC12IV_ADC12IFG4:   break;        // Vector 20: ADC12MEM4
        case ADC12IV_ADC12IFG5:   break;        // Vector 22: ADC12MEM5
        // ..............................................
        case ADC12IV_ADC12IFG27:  break;        // Vector 66: ADC12MEM27
        case ADC12IV_ADC12IFG28:  break;        // Vector 68: ADC12MEM28
        case ADC12IV_ADC12IFG29:  break;        // Vector 70: ADC12MEM29
        case ADC12IV_ADC12IFG30:  break;        // Vector 72: ADC12MEM30
        case ADC12IV_ADC12IFG31:  break;        // Vector 74: ADC12MEM31
        case ADC12IV_ADC12RDYIFG: break;        // Vector 76: ADC12RDY
        default: break;
    }
}
EXAMPLE

Differential ADC

msp430fr59xx_adc12_03.c
ADC Example

- Use a voltage source that can produce between 1V to 3V.
- Use three 10KΩ equal valued resistors to divide the voltage 2xVin/3 and Vin/3.
- Vin is varied from 1V to 3V.
- When Vin = 3V, A2 = 2V and A3 = 1V providing a differential voltage of 1V across the ADC input.
- If A2-A3 >= 1V, P1.0 set, else reset.
- Use ADC12 interrupt.
- Here A1 = P1.2 and A2=P1.3
Output Pin P1.2

- If you would like to change the behaviour of the port pin, you need to select the pin functionality using `PxSELy` registers.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>x</th>
<th>Function</th>
<th>Control Signals and Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>P1DIR.x</td>
</tr>
<tr>
<td>P1.2</td>
<td>0</td>
<td>P1.0 (I/O)</td>
<td>Inp:0, Out:1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TA1 CCI1A</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TA1.1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TA0CLK</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COUT</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A2,C2</td>
<td>X</td>
</tr>
</tbody>
</table>

- For selecting the ADC mode (A2) of P1.2, you have to set

\[
P1SEL1 = \text{BIT}2; \\
P1SEL2 = \text{BIT}2;
\]
Output Pin P1.3

- If you would like to change the behaviour of the port pin, you need to select the pin functionality using $P_xSEL_y$ registers.

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>x</th>
<th>Function</th>
<th>P1DIR.x</th>
<th>P1SEL1.x</th>
<th>P1SEL2.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1.3 (I/O)</td>
<td>1</td>
<td>I:0, O:1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>TA1 CCI2A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>TA1.2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCB0STE</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UCB0STE</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3,C3</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- For selecting the ADC mode (A2) of P1.3, you have to set

$$P1SEL1 = BIT3;$$
$$P1SEL2 = BIT3;$$
#include <msp430.h>

volatile unsigned int result = 0;

int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;            // Stop WDT

    // Configure GPIO
    P1OUT &= ~BIT0;                      // Clear LED to start
    P1DIR |= BIT0;                       // Set P1.0/LED to output
    P1SEL1 |= BIT2 | BIT3;               // Configure ADC inputs A2 and A3
    P1SEL0 |= BIT2 | BIT3;

    // Disable the GPIO power-on default high-impedance mode to activate
    // previously configured port settings
    PM5CTL0 &= ~LOCKLPM5;

    // Configure ADC12
    ADC12CTL0 = ADC12SHT0_15 | ADC12ON;  // Sampling time, ADC12 on
    ADC12CTL1 = ADC12SHP;                // Use sampling timer
    ADC12CTL2 |= ADC12RES_2;             // 12-bit conversion results
    ADC12MCTL0 |= ADC12INCH_2 | ADC12DIF; // Channel2 ADC input select; Vref=AVCC
    ADC12IER0 |= ADC12IE0;               // Enable ADC conv complete interrupt

    while (1)
    {
        __delay_cycles(5000);
        ADC12CTL0 |= ADC12ENC | ADC12SC;    // Start sampling/conversion

        __bis_SR_register(LPM0_bits + GIE); // LPM0, ADC12_ISR will force exit
        __no_operation();                  // For debugger
    }
}
#pragma vector=ADC12_VECTOR
__interrupt void ADC12ISR (void) {
    switch(ADC12IV) {
    case ADC12IV_NONE:        break;        // Vector 0: No interrupt
    case ADC12IV_ADC12OVIFG:  break;        // Vector 2: ADC12MEMx Overflow
    case ADC12IV_ADC12TOVIFG: break;        // Vector 4: Conversion time ovf
    case ADC12IV_ADC12HIIFG:  break;        // Vector 6: ADC12BHI
    case ADC12IV_ADC12LOIFG:  break;        // Vector 8: ADC12BLO
    case ADC12IV_ADC12INIFG:  break;        // Vector 10: ADC12BIN
    case ADC12IV_ADC12IFG0:                 // Vector 12:  ADC12MEM0 Interrupt
        result = ADC12MEM0;                 // read out the result register
        if (result >= 0x0AAB)                 // ADC12MEM0 = A2-A3 >= 1V?
            P1OUT |= BIT0;                      // P1.0 = 1
        else
            P1OUT &= ~BIT0;                     // P1.0 = 0
        __bic_SR_register_on_exit(LPM0_bits); // Exit active CPU
        break;                                // Clear CPUOFF bit from 0(SR)
    case ADC12IV_ADC12IFG1: break;        // Vector 14:  ADC12MEM1
    case ADC12IV_ADC12IFG2: break;        // Vector 16:  ADC12MEM2
    case ADC12IV_ADC12IFG3: break;        // Vector 18:  ADC12MEM3
    case ADC12IV_ADC12IFG4: break;        // Vector 20:  ADC12MEM4
    case ADC12IV_ADC12IFG5: break;        // Vector 22:  ADC12MEM5
    // ..............................................
    default: break;
    }
}