Programming ADC

**Understanding Single-Ended, Pseudo-differential and Fully-Differential ADC Inputs**

* For voltage intput ADC, three different input structures types exists, Single-Ended, Pseudo-differential and Fully-Differential.
* The simples solution is to select and ADC input structure that matches the sensor output. However, there are trades-offs with each structure that should be considered. In addition, if signal conditioning circuitry is used between the sensor and the ADC, this circuitry is used between the sensor and the ADC.

Fully-Differential inputs

* For Maximus noise rejection, use fully-differential inputs. Figure 1, shows an example of fully-differential ADC

Pseudo- Differential inputs

* Are similar to fully-differential inputs in that they separate signal ground form the ADC ground, allowing DC common mode voltages to be cancelled.
* They have little effect on dynamic common mode noise

Common-Mode Voltage and Common-Mode Rejection Ratios

* Common-mode voltage referes to a common voltage (i.e. with the same magnitude and phase that appears on both differential inputs). ‘

Introduction ADC

* In our daily life, anything we deal like temperature, pressure, distance or light intensity are usually in analogue form
* But microcontroller are capable of precessing only digital signals
* We want to interface any analogue sensor with our digital controllers
* There must be something that translate the analogue inputs to digital outputs
* Analogue to digital converters translate analogue signals into a sigital representations of that signal

8 bit ADC

* Sensor -> ADC -> microC -> storage/display
* V\_in -> ADC -> Binary output (D0 – D7)

Characteristics

* Step size
  + It is the smallest change that can be detected by ADC
  + We can control step size using V\_ref signal
  + For an 8-bit ADC the step size is V\_ref/(2^8)
* Resolution
  + ADC has an “n”-bit resolution , where n” can be 8, 10, 12, 16 or 24
  + High reolsution provides smalles step size
* Conversion time
  + It is the time taken by the ADC to convert the analogue input to a digital output
  + Conversion time is decided by the clock source connected to ADC
* V\_ref
  + V\_ref is an input used for reference voltage to ADC
  + If the analogue input volt ranges from 0 to 5V, the V\_ref = 5V

AVR AtMega32 IN-Build ADC

* Because the ADC is widely used in data acquisition, in recent years an increasing number of microcontroller have ann on-chip ADC peripherals like timers and USART
* An on-chip ADC eliminates the need for external ADC connection which leaves more pins for i/O
* Majority of the AVR chips come with Successive Approximation type ADC
* In AtMega32, ADC is multiplexed with PORTA

AVR Atmega32 IN-Build ADC

* The ADC in AtMega has 8 channels, that means up to 8 different sensors can be connected and get their values separately
* The ADC can be operated in
  + Single conversion
  + Free running
* In single conversion mode the ADC does the conversion and then stop
* While in free running mode it is continuously converting

AVR AtMega32 ADC Features

* 10-bit Resolution
* 0.5 LSB Integral Non- Linearity
* +-2 LSB Absolute Accuracy
* 8 Multiplexed single ended input channels
* 7 differential input Channels
* Interrupt on ADC Conversion Complete

ADC

* To get a better accuracy, inductor and capacitor used to provide a stable voltage source to AVCC from VCC
* To provide a stable voltage to Vref and increase the precision of the ADC. A capacitor is connected between AVREF and GND

AVR AtMega IN-BUILD ADC- Registers

* ADC multiplexer selection register -ADMUX
  + For selecting the reference voltage and the input channel
* ADC Control and Status Register A- ADCSRA.
  + It is used to read the status of ADC and also used for controlling ADC
* The ADC Data register – ADCL and ADCH
  + The final result of conversion will be stored on these registers

ADC ADMUX – register

* Bit 7:6 – REFS1:0
  + Reference selection bits – These bits are used to choose the reference voltage
    - **We have alreadu done that by setting V\_ref (Pin 8) to Vcc**
* Bit 5 – ADCLAR
  + ADC left adjust result – Make it 1 to left adjust the ADC results
    - **We don’t have that, maybe there is a register I should look closer at in our ADC internal settings**
* Bits 4:0 – MUX4:0 – analogue channel and gain selection bits
  + There are 8 ADC channels (PA0:7). Which one doo we choose? You can choose it by setting these bits (2^5 = 32 different channel selection combination)
    - **We have 4 ADC channels and 4 MUX inputs (MA0:3 = DB0:3)**
    - **We have t**
  + However, we are concerned with only with the first 8 conditions, initially all the bits are set to zero

ADC ADCSRA – register

* Bit 7 – ADC Enable – to enable the ADC feature
  + **We don’t have to that**
* Bit 6 – ADC Start Conversion – to start the conversion
  + **This can be a bit complicated in our case**
* Bit 5 – ADC Auto Trigger Enable – To enable auto triggering of ADC
  + **Not something we should be concerned about**
* Bit 4- ADC Interrupt Flag – To check whether the conversion is complete or not
  + **Not concerned with in our case**
* Bit 3 – ADC Interrrupt Enable – To enable the ADC interrupt
  + **Not something we need to configure**
* Bit 2:0 – ADC Prescaler Select Bits – The prescaler is determined by selecting the proper combination from the following
  + This has something to do with division factor, I don’t know what that is
  + **Not something I should be concerned with**

ADC ADCL and ADCH

* The result of the ADC conversion is stored here
  + **In our case it is stored DB0:7**
* Since the ADC has a resolution of 10 bits, it requires 10 bits to store the result. Hence one single 8-bit register is not sufficient
* We need two register – ADCL and ADCH (ADC low byte and ADC high byte)
  + **Not something to be concerned about**

**Our ADC Functional description**

* ADC0844 contain 4-channel analogue input multiplexer
* Each MUX input can be configured into one of three modes of operation
  + Operation differential
  + Pseudo differential
  + Single ended
* The specific mode is selected by loading the MUX with proper address
  + Input to the MUX address latch (MA0:3 = DB0:3)
  + The RD must be held high
* Conversion is initiated via the CS and WR lines.
  + If the data from the previous conversion is not read the INTR will be low
* The falling edge of WR signal will reset the INTR line high and ready for the ADC conversion cycle
* The rising edge of WR with RD high, strobes the data in the MA0:3 = DB0:3 inputs into the MUC address latch to select the new input configuration and start a conversion.
* After the conversion cycle, which is set by the internal clock frequency, the digital data is transferred to the output latch and the INTR is asserted low.

**Multiplexer Configuration**

* The deign of these converters utilizes a sampled data comparator structure which allows a differential analogue input to be converted by a successive approximation routine.
* The actual voltage converted is always the difference between the assign “+” input terminal and “–“ input terminal.