Design for Internet of Things

Prof. T. V Prabhakar
Department of Electronic Systems Engineering
Indian Institute of Science, Bengaluru

Lecture - 05
Challenges Part-03

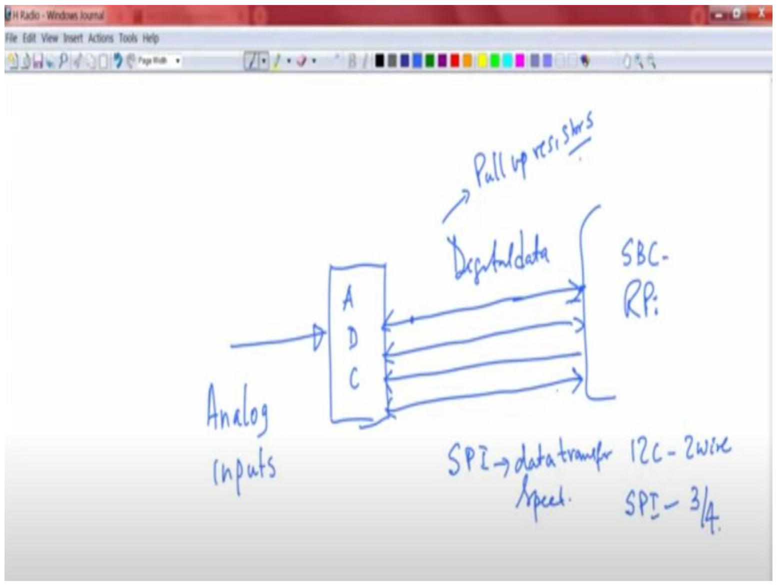
This module is supposed to demonstrate, how to interface a analog sensor to an ADC and acquire the data. For that purpose, we have taken an air quality sensor and that sensor interface we will try to make it and show you with respect to a physical demonstration. Which means that the sensor has to be connected to the analog pin of either an SoC or to that of a standalone ADC that is available.

Now, if you identify a board, let us say an aggregation board like Raspberry Pi or any one of those multi core single board computer-based systems. Usually, such boards will not have analog inputs, that is a problem because it has a different purpose. So, you cannot think about that as a unit for sensing. It is mostly for aggregating data from several embedded systems and then giving the data over a digital bus to this aggregation unit.

So, the purpose is different, and it is a high compute. Because you can run an OS, then it is multicore, it can connect to the internet, it can transfer data as it is receiving data from the digital ports, it can also upload data and so on. So, it is a very powerful single board computer particularly if you look at Raspberry Pi and those kinds of class of single board computers. If you now decide that I will interface a ADC to this Raspberry Pi.

You are left no choice except to look for an isolated one single ADC chip and which gives you digital output so that you can interface it to the Raspberry Pi on the other side, it should be able to give you analog input.

(Refer Slide Time: 02:33)



So, just to sketch this picture, we will have, you want to connect analog inputs on this side, you want to generate ADC conversion here. And then you want to transfer this digital data to a single board computer such as Raspberry Pi. So, these are digital ports. Typically, this digital data and this digital port can either be over a two-wire interface or it can be over a three-wire interface, sometimes even a four wire interface.

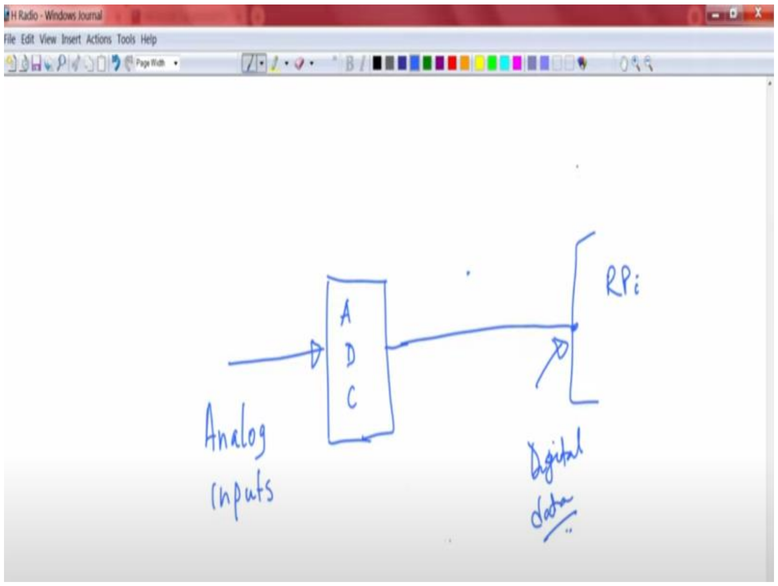
It can be 2 wire, 3 wire or 4 wire. If it is 2 wire interface, actually this arrow is in both direction. So, we should not restrict ourselves to one direction only in fact we can show it here and so on. If it is 2 wire these kinds of systems typically have what are known as pull up resistors. These a digital bus and this digital bus has a name called I2C inter IC communication. That is why it is called I2C.

Sometimes you may want to do it over 3 or 4 wires in which case you will still have these 2 lines, plus you will have an additional third line, which could be unidirectional and then there will be another line. This kind of is also another way of transferring data. And this is also a digital communication between ICs and such an interface is called SPI. Serial Peripheral Interface and the other one as I mentioned was I2C which is just 2 wire.

And SPI can be sometimes 3 wire and sometimes also 4 wire interfaces. Of course, this will give you much higher data transfer speed, is much higher as compared to I2C and so on. It has its own advantages. Why two different types of digital interfaces? Because each one has its own advantage and one of them is indeed high data rate if you want you have to go in for SPI kind of interfaces.

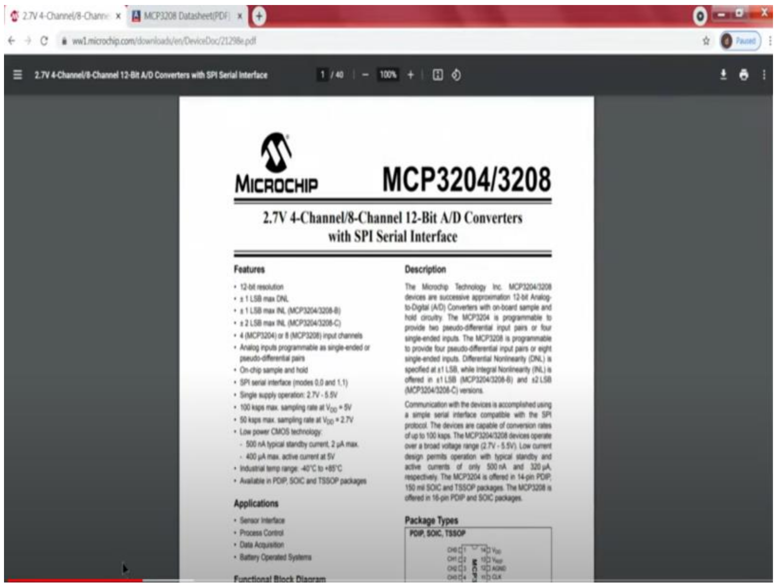
Now, let us take that was not our objective, our objective was to connect a analog ADC chip to a single board computer such as a Raspberry Pi. We drew this picture and let us redraw it again by removing all the clutter that we had and essentially just look at some plain stuff.

(Refer Slide Time: 06:33)



So, here you are getting digital. One such chip that is quite popular in the IoT world is this chip called MCP3208.

(Refer Slide Time: 06:59)


2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface

ww1.microchip.com/downloads/en/DeviceDoc/21298e.pdf

MCP3204/3208

2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface

Features

- 12-bit resolution
- ± 1 LSB max DNL
- ± 1 LSB max INL (MCP3204/3208-B)
- ± 2 LSB max INL (MCP3204/3208-C)
- 4 (MCP3204) or 8 (MCP3208) input channels
- Analog inputs programmable as single-ended or pseudo-differential pairs
- On-chip sample and hold
- SPI serial interface (modes 0,0 and 1,1)
- Single supply operation: 2.7V - 5.5V
- 100 ksps max. sampling rate at $V_{DD} = 5V$
- 50 ksps max. sampling rate at $V_{DD} = 2.7V$
- Low power CMOS technology:
  - 500 nA typical standby current, 2 µA max.
  - 400 µA max. active current at 5V
- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

Applications

- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

Functional Block Diagram

Description

The Microchip Technology Inc. MCP3204/3208 devices are successive approximation 12-bit Analog-to-Digital (A/D) Converters with on-board sample and hold circuitry. The MCP3204 is programmable to provide two pseudo-differential input pairs or four single-ended inputs. The MCP3208 is programmable to provide four pseudo-differential input pairs or eight single-ended inputs. Differential Nonlinearity (DNL) is specified at ±1 LSB, while Integral Nonlinearity (INL) is offered in ±1 LSB (MCP3204/3208-B) and ±2 LSB (MCP3204/3208-C) versions.

Communication with the devices is accomplished using a simple serial interface compatible with the SPI protocol. The devices are capable of conversion rates of up to 100 ksps. The MCP3204/3208 devices operate over a broad voltage range (2.7V - 5.5V). Low current design permits operation with typical standby and active currents of only 500 nA and 320 µA, respectively. The MCP3204 is offered in 14-pin PDIP, 150 mil SOIC and TSSOP packages. The MCP3208 is offered in 16-pin PDIP and SOIC packages.

Package Types

PDIP, SOIC, TSSOP

Now, if you open a data sheet like this, what you will be sort of alarmed is to see these features. And you must know these features in some detail, particularly in this course, and understanding the datasheet becomes a very critical part, let me first give you a high-level view of these parameters. And then we will do a demonstration and then get into further details of these parameters.

First of all, the resolution of this ADC is 12-Bit, that means every time it samples an analog value, it converts it into 12-Bits. That is the idea of being a 12-Bit resolution ADC. Then this ADC, any sampling operation, any analog signal sampling operation cannot be noise free, cannot be without nonlinearities in the sampling process. Any ADC is characterized by the nonlinearities associated with that chip; you have no way but to escape from that.

So, first of all, you should get an idea what are all the nonlinearities and how did the manufacturer or the vendor quantify those nonlinearities. First thing that occurs to you is after 12-Bit resolution, the second parameter he shows is DNL. DNL essentially stands for differential nonlinearity. The DNL parameter is plus minus one LSB maximum. How did you arrive at this plus minus one LSB? We will come to that separately.

Similarly, there is something called INL which is integral nonlinearity that is also given as plus minus 1 LSB INL. And MCP3208 and MCP3204 there is no difference except that 3208 gives you 8 analog inputs. You can interface up to 8 analog inputs and the chip provides an on-chip sample and hold because you have to sample the analog input and hold the output so that the output is exactly for that sampled input.

So, you need a another circuit called sample and hold and that sample and hold is integrated inside it. If you are not clear, hold on for two minutes and I will explain how that thing works and how the overall architecture of an ADC actually looks like. Then we said that the analog inputs can either be with respect to ground and analog output from a sensor with respect to ground or with respect to the difference between two outputs.

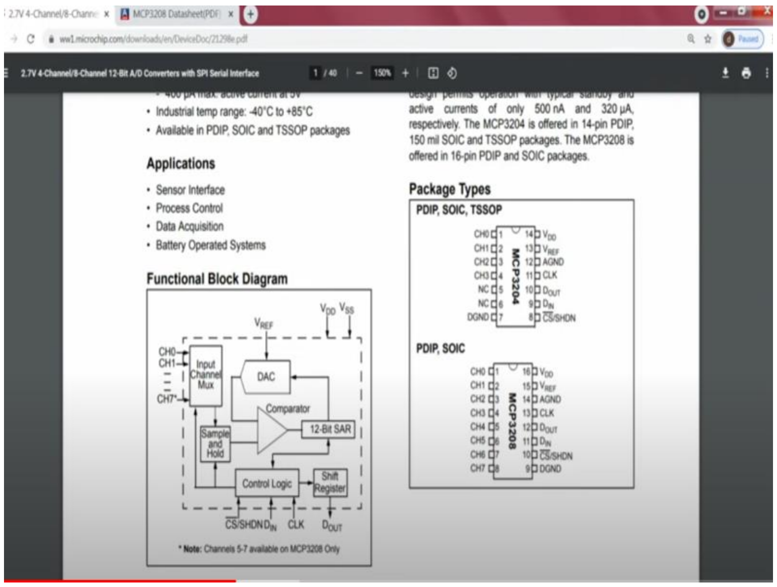
Typically, if you configure it in a bridge, if you put a sensor output you are trying to draw sensor output from a bridge circuit, you will get differential output. There is no ground there. It is the difference between V1 and V2 that you are taking in and then converting that. So, you will get two voltages essentially, but if you are measuring if you are getting a sensor output with respect to output voltage and ground, then it is called single ended, otherwise it is differential.

So, you use differential, why you have to use differential? Sometimes why does someone want to use differential? The simple reason that any noise that appears in one line will also appear on the other line and therefore, when you feed it to any amplifying circuit, the noise from both of them get canceled out and you will only get a clean signal that is why you need differential input. In fact, several digital buses actually providing only differential signals.

And this is a very important requirement because noise if you have a single ended one noise is very simple, easy to get injected into that line. Because it is active with respect to ground, any small lift from the ground is a voltage and that voltage will interfere with the sensor and the sensors output and then that sensor output will be swinging with respect to the actual output that has come.

So that can create a sort of an issue where you are not able to digitize that value properly. So, with single ended, I mean the differential you will avoid that problem anyway, so, that is one thing. And then I mentioned about sample and hold. I also mentioned about the SPI interface. So, I would not go into the detail, so this chip supports that. This chip also says that you provide me a range over a operating voltage range anywhere from 2.7 to 5.5. And people actually use 5 volts sometimes.

(Refer Slide Time: 12:14)



2.7V 4-Channel/8-Channel 12-Bit A/D Converters with SPI Serial Interface

- Industrial temp range: -40°C to +85°C
- Available in PDIP, SOIC and TSSOP packages

Applications

- Sensor Interface
- Process Control
- Data Acquisition
- Battery Operated Systems

active currents of only 500 nA and 320 μA, respectively. The MCP3204 is offered in 14-pin PDIP, 150 mil SOIC and TSSOP packages. The MCP3208 is offered in 16-pin PDIP and SOIC packages.

Functional Block Diagram

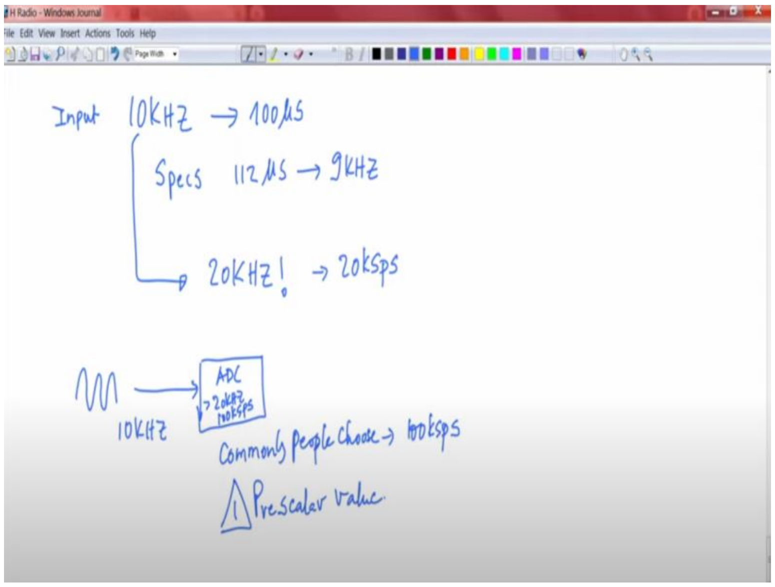
* **Note:** Channels 5-7 available on MCP3208 Only

Package Types

And also, people use 3.3 and so on. next parameter is 100 kilo samples max sampling rate at VDD is equal to five volt and it is down to 50 ksps kilo samples per second max if the VDD is operating at a lower voltage of 2.7 volts. So, directly related to the sampling frequency, the sampling rate, is our sampling frequency, is directly dependent on the operating voltage here. Now, let us eject out of this datasheet.

Otherwise, you will be wondering why this sampling is an very important thing. Let us go back to this basic picture that we drew, and I will explain a little bit about the sampling itself. If you take a 10 kilo hertz signal it boils down to 100 microseconds, you do one over 10 kilo hertz you will get 100 microseconds. Now, let us say an ADC specification provides 112 microseconds. That means, if you convert this back you will get maybe I should just organize this slide a little better, I will take a new sheet and we will go to that.

(Refer Slide Time: 13:51)


H Radio - Windows Journal
File Edit View Insert Actions Tools Help
Input 10KHZ → 100µS
Specs 112µS → 9KHZ
20KHZ! → 20ksps
ADC
>20kHz
100ksps
10KHZ
Commonly People Choose → 100ksps
Prescalar value

So, let us start here you have a signal an input signal which is coming onto the ADC at 10 kilo hertz, which means it is 100 microseconds, period is 100 microseconds. The specifications that you have with you with respect to an ADC in your hand gives you 112 microseconds, let us say you have 112 microseconds. if you convert these 112 microseconds, you will get back how much close to 10 kilo hertz but far away because you get about 9 kilohertz.

Now if you want to sample a 10 kilo hertz signal if you want to sample successfully and apply the Nyquist criteria for a 10 kilo hertz input signal, you need at least 20 kilohertz. Now, by the way, people interchangeably use 20 kilo hertz to 20 kilos samples per second both mean one and the same. So, I will replace this with 20 ksps. Now it is easily connecting right, because you saw the same number. 20 ksps is something that you must minimum support.

If you are getting an input, this is your input to the ADC. So let me draw this here. This is your 10 kilohertz signal. This is your ADC; it should sample the signal. Let us say this is your analog signal. It should sample the signal at least 20 kilohertz, if not higher than that. Typically, people choose it is common to choose 100 ksps that means you are doing 10x, why 10x? Because it is arriving at 10 ksps, you are sampling at 100 ksps.

Now, how to do this? That is the question. So, commonly people choose this is just a thumb, they choose 10 times that, so it should be 100 ksps. Now, some SoCs and some ADCs will let you tune with what is known a prescaler. Please look up, I do not want to diffuse, so I want you to look up the prescaler. This is something that you can configure on your ADCs there is something called a prescalar value, which you can try. And that will essentially change the sampling rate.

Why I gave this example is Arduino typically does this 9 kilo hertz sampling or 9 ksps. And you do not get anything if you do a sampling at that 1x the input you get nothing. And 2x is the minimum. That is what Nyquist sampling says. And yet, it is that is also the very rare minimum that one can use. So, people usually choose up to about 10x which is quite common. So having got this much to understand, let us go back to the datasheet.

So, I will keep ejecting and coming back so that we connect everything about the datasheet. So, now let us see this the sampling ksps maximum is 100 ksps that means at 10 kilo hertz signal can beautifully get sampled no problem at all. If you supply a 10 kilo hertz signal, because you are sampling at 10x times more, that is possible. But that should be possible only if you operate the chip at 5 volts.

Otherwise, you can only do it at 5x the input frequency, I am making an assumption that the input is at 10 kilohertz, please note, that is why I am saying 5x then it comes to 50ksps. And that is possible you are putting it down to 5x because you are also operating at a lower voltage. See, you have to be very careful about all these numbers. Because why are you taking 10 kilo hertz and giving you 5x at 2.7?

If your signal is swinging at 10 kilo hertz, I do not think you should ever compromise on working at a 2.7. You should put it back to 5 volt and you should do 10x sampling only. This option you should choose 5 ksps when your signal is at 1 kilohertz not 10 kilohertz. That will help you to get at least 50x, you will get something like close 50x which will allow you to sample.

So, it depends now on your input frequency. So, analog signal frequency that is coming in and accordingly you should choose that this is one part of the story. Then let us come to the technology under which this chip was manufactured. So, let me just expand this to a little bit so that you see these things much more clearly. So, you see low power CMOS technology 500 nano amps standby current is so much and active current is 400 micro amps and all that.

It appears to be good for good low power system. And it appears to be used in applications such as sensor interface, and so on. That is also a very nice thing about it. Now, we should know, I mentioned to you that I will explain a little bit about, how this ADC works? And I could not find a better picture than what I found in this little diagram here. So let us spend some time, I drew that previously, about mux, analog mux, and so on. But I think the best capture is actually in this picture. So let us go through this picture, I will give you a high-level view.

Basically, any analog signal will get fed here, channel 0 to Channel 7 are your 8 analog inputs, which coming here. The actual conversion is only on one channel at a time. Please note this, it is only one at a time because the core of the electronics, which is all this system here is just one block. For example, you have only one DAC, you have only one comparator, and you have one 12-Bit SAR, and so on. So, these blocks are the heart of everything.

And they essentially can only work on one signal at a time. So, although you have channel 0 to channel 7, as inputs, you can connect 8 sensors at a micro instant in time, you basically can only sense one, and then you have to write your software logic to go to the next, the next and so on. So, giving you a pseudo feeling that all 8 sensors are being sensed in a simultaneous manner, you will only get that pseudo sense.

But you must note that there is a delay. When you do round Robin, for instance, channel 0, channel 1, channel 2, channel 3, go up to channel 7, and then you come back to channel 0. That delay is what you will have to factor in when you have your signals coming in. And that works very well when the signals are varying slowly or at a lower frequency, you will be able to satisfy this condition of moving from channel 0 to channel 7 and then coming back to channel 0 in time to catch up for the next wave that comes in.

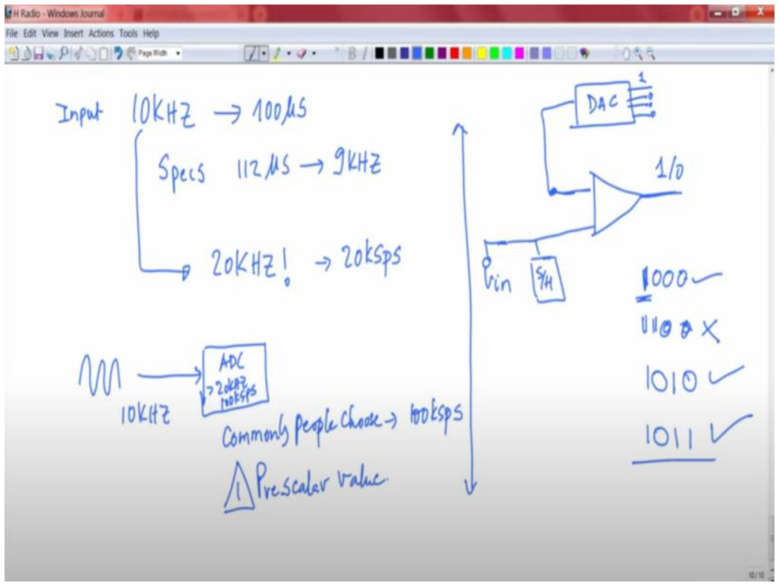
So, all that will have to be factored in. Actually, a successive approximation ADC, there are many types of ADC, there is something called flash ADC, there is a sigma Delta ADC, and so on. So many different types of ADC are there. This is one type of ADC very popular ADC, up to about 16-Bits or so if you take 4-Bit ADC, 8-Bit ADC, 12-Bit ADC, 16-Bit ADC and so on. They more or less use the successive approximation ADCs

And that is what is contained in these three blocks. DAC, Comparator, 12-Bits are the three heart of the whole system. And that is where the whole ADC actually works. It is strange that your ADC actually contains a DAC that means it has a DAC digital to analog converter. Why? Now we will understand why that actually happens. So, when you have an input signal coming in on channel 0, that is being fed as an input to this comparator through this sample.

And hold, I mentioned to you that sample and hold is a very important block, which will hold the output for that input signal that is very important. So, it will hold the output. So that signal is fed here. Now comparator is actually a device, a block which gives you 1 or a 0, it would not do anything more than giving you ones or zeros. In fact, your input signal is actually sensed whether it is 1 or 0 actually, it is actually does not get converted in that sense.

The actual conversion is actually happening based on what the 12-Bits are feeds into the DAC when the 12-Bits are feeds into the DAC, the DAC gives an output and then you get a comparator output which is 1 or 0. Let me give you an example here.

(Refer Slide Time: 24:27)



So, let me go back, how this actually does is? This is your comparator; this is your Vin now you have to be careful. I will not put my Vin directly, I will pass it through S and H here, I will just show it like this. This one is actually coming from the DAC output, that means it is giving you an analog output. Now what is the DAC output? Why is it giving an analog signal? Because that is the business of the DAC.

It is supposed to take a digital input and then gives you an analog output. How it starts is, it will first start with, let us say you have a 4-Bit ADC. It will first put one in the MSB 0, 0, 0 corresponding to this one in the MSB some voltage is given and then it will check whether this will give you a 1 or a 0. Now, a comparison actually happens in this comparator based on the output of the comparator.

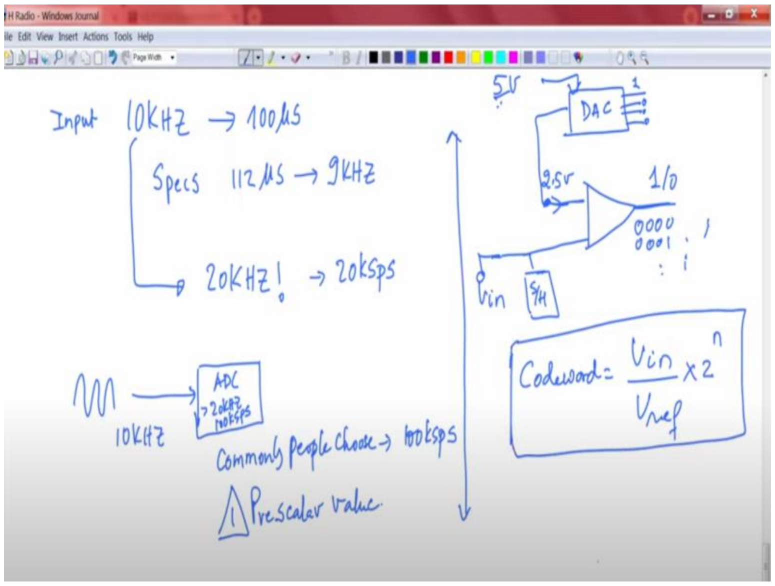
You actually know whether this one should be retained or whether this 1 should be flipped and you should try 0100 or whether you should try if it actually gives you a one output you are happy then you say I will try 1100. Please note, whether you are trying MSB one, whether if this succeeded giving you a corresponding output, then you will try 1100. Now, if this failed, what you will do, this succeeded already.

So, what you will do, you will make 1, this failed, so, I will make it 0, then I will make this bit one, then this I will try. If I try like this and if this succeeded fantastic. So, I will retain this bit and this bit, now I will try this bit, which one I will try, one, this failed, I leave that, this passed earlier. Now, I will try this if this also passes then this is the output finished. So, each time you will do a approximation, you will do a comparison and then that comparison is essentially what is fed out.

So, you do one bit at a time one MSB position at a time and keep rotating it. So, let us go back and I hope you got this picture. If you indeed got that picture, let us go back and look at this one. So, this is your comparator and this comparator this 12-Bits is actually giving you the digital value or the code word that is held, the digital value that is held for which the comparator passes based on the MSB and the other corresponding bit positions.

So, with what will DAC give you an output? It will give you an output based on its reference to which it is fed. Vref here, when you talk about Vref in an ADC actually it is being fed to the DAC block of the Vref. So, your analog input is actually compared with that of the reference that is fed to the DAC and that Vref actually will give you a corresponding analog, with respect to Vref, you will get a corresponding output from the analog output from the DAC.

(Refer Slide Time: 28:15)



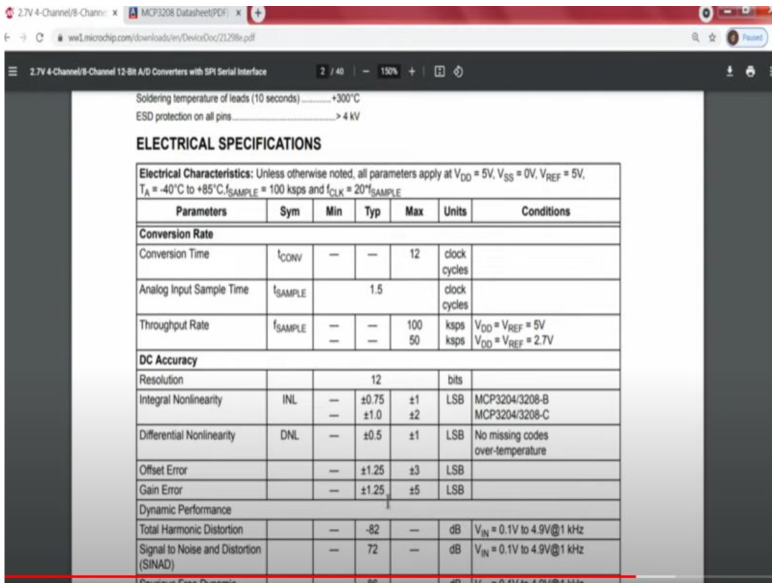
That is how your two analog signals may come. In other words, let me put back this picture, if this is confusing, if this is Vref here, let me shift it a little bit. If this is 5 volts, this will be what 1000 try and tell me where exactly this 1000 comes. This will be exactly at half, it will be exactly at half, correct? If you go above 0000, 0001, 10 and all that, and then one here is essentially exactly half, half of five volts is how much, 2.5 volts.

So, if this pattern is sent, it is actually giving you how much here 2.5 volts, that is all. And now you compare your input with respect to this 2.5 volts which is being fed. And then you come up with simple expression, which is code word is equal to Vin divided by Vref. So, you see it is the ratio of Vin by Vref into the two-power n, where n is the resolution. This is the basic expression which everybody starts off when we talk about ADCs.

So let us come back here. So, you see this functioning we now know very well. Beautifully we have understood how the ADC works. Still, we have not yet figured out how do we interface this

ADC? We just discussed a few things. This you can read; you can download this datasheet and you can read it. But we have not yet crossed over, because there are some very crucial parameters, which we have to understand before we move on and what are those.

(Refer Slide Time: 30:28)



Soldering temperature of leads (10 seconds) ....+300°C
ESD protection on all pins.................................> 4 kV

ELECTRICAL SPECIFICATIONS

Electrical Characteristics: Unless otherwise noted, all parameters apply at V_{DD} = 5V, V_{SS} = 0V, V_{REF} = 5V, T_A = -40°C to +85°C, f_{SAMPLE} = 100 ksps and f_{CLK} = 20*f_{SAMPLE}

Parameters	Sym	Min	Typ	Max	Units	Conditions
Conversion Rate						
Conversion Time	t_{CONV}	—	—	12	clock cycles	
Analog Input Sample Time	t_{SAMPLE}		1.5		clock cycles	
Throughput Rate	f_{SAMPLE}	— —	— —	100 50	ksps ksps	V_{DD} = V_{REF} = 5V V_{DD} = V_{REF} = 2.7V
DC Accuracy						
Resolution			12		bits	
Integral Nonlinearity	INL	— —	±0.75 ±1.0	±1 ±2	LSB	MCP3204/3208-B MCP3204/3208-C
Differential Nonlinearity	DNL	—	±0.5	±1	LSB	No missing codes over-temperature
Offset Error		—	±1.25	±3	LSB	
Gain Error		—	±1.25	±5	LSB	
Dynamic Performance						
Total Harmonic Distortion		—	-82	—	dB	V_{IN} = 0.1V to 4.9V@1 kHz
Signal to Noise and Distortion (SINAD)		—	72	—	dB	V_{IN} = 0.1V to 4.9V@1 kHz

Those come from this datasheet here continuing on the datasheet, one of them is the conversion time. If you feed a clock to the ADC, you need 12 clock cycles to convert from analog to the digital domain that is what the conversion time is. Then you have analog input sample time throughput rate. We mentioned this already, some people call it sampling frequency, some people call it throughput rate and so on.

It is nothing but that 100 and 100 ksps at Vref of 5 volts and VDD of 2.7. We said it will come down to 50 ksps. Then we said all the things about accuracy of the ADC, DC accuracy of the ADC look at what he has said. Resolution is 12-Bits that appears great, we know about the resolution. But what about the other problems with respect to ADC and its linearity? Definitely it has a integral nonlinearity it is called INL.

And that is quantified with the typical number of plus minus 0.75 LSB and maximum it can go up to plus minus one LSB. And differential nonlinearity is also quantified here. Offset error is plus minus 1.25 and the maximum it can go up to is plus minus 3 LSB. Gain error plus minus 1.25, plus minus 5 LSB, then total harmonic distortion is this number minus 82 dB, which has a

Vin of 0.1 volt to 4.9 volt at one kilohertz and signal to noise. And distortion is now down to 72 for the same specification.

So, there are quite a few parameters which you have to understand because your choice of resolution actually has a bearing on this. It is a very important parameter, at least you should know a few parameters. And I will give you some interesting insights into these things. So, that you will be able to read data sheets and understand them even better.

Because this course is about choosing components, how do you do these components unless you know these components becomes difficult for you to choose them for your project to choose them to make you understand this in a structured way. So, our next task is to understand if you pair DC accuracy parameters, particularly offset error, gain error, and then we will just touch upon integral nonlinearity and differential nonlinearity.

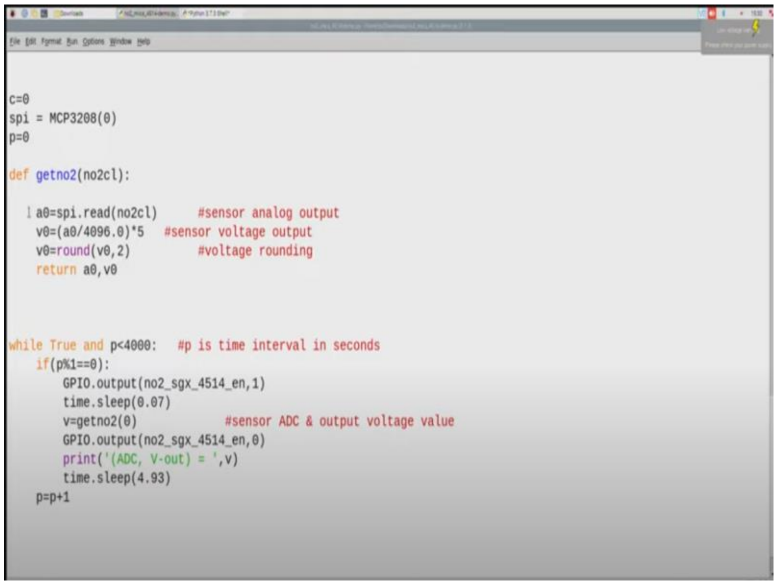
But before we do that, let us do a demo. Let us finish with a small demo. So that this monotony of lecturing is stopped and then we will come back and discuss these four DC accuracy parameters. You see this board, this board has several sensors, this is an air quality sensor board, you will see that there is a sensor here, there is a sensor here and another sensor here and there is one more sensor down below here this is this here sorry this one.

This is an interesting sensor. This is a sensor called SGX4514. It is mounted directly on the board; it looks like a chip and it is a NO2 nitrogen dioxide sensor and carbon monoxide sensor. Both it will give, and it is cross sensitive to other gases. So that is the idea. And it is the semiconductor gas sensor, these some of the sensors that you see here are actually electrochemical sensors.

But the one that looks like a chip indeed is a semiconductor sensor. Now, all these sensors are connected to a Raspberry Pi board. Which is connected on top of this, and it just goes on top of this, there is a covering box and on top of this box, there is a board, and you can see the standard headers, directly interface to the Raspberry Pi. But if you reverse this board, you will see the whole lot of electronics here and what you see here, this one for instance is indeed our famous ADC chip and this ADC chip directly interfaces to the Raspberry Pi.

And that is the nice thing about this ADC chip can actually interface very well. This one, this ADC chip can interface very well directly because it gives digital output, takes analog input and gives you a digital output and our SPI interface, it connects directly to the Raspberry Pi, forget about all what you see here. This is all this using a glue gun to arrest some wires, which we had to patch up. This is just a sample board that we were using.

(Refer Slide Time: 36:31)



```
c=0
spi = MCP3208(0)
p=0

def getno2(no2cl):

    a0=spi.read(no2cl)      #sensor analog output
    v0=(a0/4096.0)*5   #sensor voltage output
    v0=round(v0,2)          #voltage rounding
    return a0,v0



while True and p<4000:   #p is time interval in seconds
    if(p%1==0):
        GPIO.output(no2_sgx_4514_en,1)
        time.sleep(0.07)
        v=getno2(0)            #sensor ADC & output voltage value
        GPIO.output(no2_sgx_4514_en,0)
        print('(ADC, V-out) = ',v)
        time.sleep(4.93)
    p=p+1
```

So, what is the corresponding code, software for this particular board? Well, let us look at that, that one essentially talks about acquisition of data and how do you take that data and how do you acquire the data and then how do you convert that data into corresponding analog input? You want to know what is the analog value that came in through your code. So, you see now, if you look carefully, the MCP3208 SPI is indicated, that is SPI.

Code in the computer says that you read over SPI you can see that a0 is equal to SPI dot read you are saying, that means, the digital value corresponding to that analog output is actually stored in a0. Next what you do, you want to know, what is the sensor voltage output? So, what do you do? Well, it was quite simple, what you do is, this essentially is you are interested in the input voltage which you can simply get from this expression here.

This is code word divided by 2 power n times Vref. That is essentially what we are shown in the code as well. So, if you look carefully into the code part, the code part simply says a0 that is your code word essentially the digital output divided by 2 power n, which is in our case it is a 12-Bit ADC at 4096 times, the reference voltage is 5 volts. So, you got back the analog input as seen at the input of the ADC.

So, this is how you acquire the data. And now once you know your input voltage for that particular voltage, what is the processing you will have to do? The datasheet of SGX would have told you so, you go there and do a few more manipulations, you will be able to read NO2 quite accurately. Will do that also, will try and see what is the end of the chain of how you take this analog value that you painstakingly measured through the ADC and then got the digital output and then converted it back into the analog output form.

And how can you use it effectively, and then we can process from there. We will do all of that. But before that, I promised that I will go into a little more of detailing with respect to the ADCs and its performance, which we will take up in the next class. Thank you very much.