CHARGING INFRASTRUCTURE

Prof. Apurv Kumar Yadav

Department of Electrical Engineering

Indian Institute of Technology Roorkee

Week-12

Lecture-56

Lec 56: DC Charging System (BEVC-DC001)-II

Hello everyone welcome to the lecture number 56 of this NPTEL lecture series on charging infrastructure and today we will carry forward our discussion on the DC charging system using BEVC DC001 now if we talk about you know our discussion in the previous class we have discussed about DC001 charging system and CAN communication Further we have also seen how the BEVC-DC001 works and what are the key parameters of the BEVC-DC001 charging system. Here we have input 3 phase AC and it supports 48V battery system with the maximum current going up to 200A which corresponds to the maximum power level of 10kW. And also it supports up to 72 volt battery systems with the maximum current rating of 200 ampere which leads to the maximum power level of 15 kilowatt.

And in this the communication is derived from IEC 61851 which is again been incorporated in AIS 138 part 2 especially the annexure G. part and it uses the GBT27930 application layer that means it defines how the CAN data frames are being represented and we have discussed that it uses the CAN communication and it uses the GBT20234.3 connector with the 5 meter standard cable and it was introduced by the Department of Heavy Industry Government of India. And then we move there and see how the detail communication takes place in DC001 charging system. And in that we have seen that it's the GBT27930-2015 standard defines the rule for implementing the physical link layer and the data link layer, which is again dependent upon the SAEJ1939 standard. and ISO 11898 standard and also represent how the application layer of that particular communication is being represented that means it indicates how the different parameters how they are grouped together what are their parameter group numbers, what are the messages which are there, how they have been represented and how the data frames are being transmitted in the form of protocol data unit format. All those things will be defined in GBT27930-2015 standard and they have derived the standard by taking this GBT27930 standard further it mentions that the speed with which it can go is 250 kbps and it uses CAN 2.0b identifier which is nothing but 29 bit identifier and that means it can accommodate 2 raise to power 29 distinct or unique messages in its database and it only have two participants it is a dedicated communication between the charger and the vehicle or you can say in the vehicle it's a BMS And whenever the charger says that they are compliant with the DC-001, then they must have the addresses which is defined as 56 in hexadecimal or 86 in the decimal.

And on the other hand, the vehicle or the BMS which is compliant to DC-001 must have the address which is nothing but 244. or F4 which is in the hexadecimal format. Both the 56 hexadecimal and F4 is in hexadecimal and that is the address for the BMS or you can say for the vehicle controller which is actually communicating. And this CAN communication is dedicated CAN communication which does not have any connection with other CAN systems in the vehicle like powertrain CAN or the ECU CAN or you can say the other sensor or the CAN communication system which are available in the vehicle. So there are two different CAN communication systems which are there.

One is dedicated for this charging purpose in this particular charging system and one is for the main vehicle communication. That means both these CAN are independent of they will not interfere within themselves. So, before understanding how the communication over the CAN takes place in DC-001, we have started understanding about CAN communication, why the CAN communication, why it is been widely accepted among different OEMs in the automotive industry. So, we have understood that the CAN communication is nothing but called as the Control Area Network is basically a two-wise serial communication where the messages are sent sequentially. so in can what we have is we have the can message one set of message which has been sent from one node to another node will be called as the can message so whenever one node is communicating they will be sending one can message now it is basically message based protocol which means that in the message itself the priority of the messages are been defined they also allows for you know several features to be added just via software alone you need not to add the additional hardware for future expansion or for future modification you need not to put any additional hardware into it there are certain advantages to it it is low cost because it reduces the copper content in the vehicle by reducing the number of wires which is laid down inside the vehicle It is a primary centralized communication where all the nodes are connected via CAN bus and the information when once shared by one of the sender will be looking by all the other nodes as well. It is a robust communication means it is not affected by the nearby electromagnetic interferences because the message informations or the information related to message priorities things that will be defined in the message itself. So it is easy to implement to define what is the message IDs and all. and error compatibility so each CAN message will have the CRC embedded into it which is Cyclic Agency Code which will be embedded into that message which will help in identifying whether the frames are correct frame or is there any error involved in that frame or not again it is very flexible because it's a message based protocol so just by changing the software you can improvise or you can modify

or you can add the new nodes in the system and since in the message itself all the information are embedded only the messages has to be modified such that your entire communication network can be changed so that's why it is very flexible it can be easily expandable as well plus it is a multi master configuration because any number of nodes connected onto the CAN bus can be the sender node and others will be the listener node so it is like this the node which is sending the message will be occupying the entire CAN bus and they'll be putting the messages and which will be heard by all the other nodes which are connected onto the channel and once the message is heard by different nodes they convert themselves to receiver node and once they receive the message they will cross verify from their CAN database that whether the message which are coming are correct or not and if it is correct then they will take it in and perform required operation if it is not within their database they will just ignore the message and they stop taking the data for that particular CAN message again since in the message itself the message ids are embedded so each message has this message id it is like an address to that particular message and lower the message id higher will be the priority of that particular message and this is one of the important you can say property of the can data frames which are used to easily identify which node will actually own the bus and put in the information and which node will be just there to be ready to receive those data that means it becomes the receiver node and which one will become the sender node so that will be defined by this message id the node which is sending the message with highest priority

will be actually owning the bus while others will be just convert themselves to the receiver nodes and they will be just listening the messages which will be sent by the sender and that sender will keeps on changing depending upon the priority of the messages or you can say whose information is there in the message id in that can a message which is been put in onto the node So let us see the other understanding related to CAN communication because that will help us to understand how the priorities are defined, how the messages are put forward into the CAN bus. Now if we talk about the CAN communication as I mentioned it is a two wire system defined by CAN high and CAN low. So two wire system CAN high and CAN low which will be running across the area over which that is covering the network and through that wire only the different nodes are being connected together. Then the data is sent differentially onto the CAN high and CAN low channel.

Now this sending of this differential signal on this CAN high and CAN low channels will help the CAN communication to be robust against the external electromagnetic interferences. Now this is the way by which you know because of the differential signaling on the CAN high and CAN low wire will actually help the CAN communication to be robust against the external electromagnetic interference. And since they are differentially sent so finally the difference between the voltages at the CAN high and CAN low channel will actually gives the actual data. Now this is again as I mentioned eliminates the corruption of data due to electromagnetic interferences from the nearby circuit. now can communication has two logic one is logic 0 another one is logic 1. the logic 0 is called as the dominant logic and logic 1 is called as the recessive logic as the name suggests the logic 0

is the dominant logic and logic 1 is the recessive logic that means if in a canvas if one of the node let's say node 1 and node 2 is connected if let's say node 1 is pushing logic 0 and node 2 is pushing logic 1 it's the dominant logic which is the logic 0 from the node 1 will actually dominate this recessive logic 1 or you can say the information which is sent by N2 and then this N2 will get converted into receiver node and it will start accepting whatever the data which is pushed in on this CAN bus by the node 1 and node 1 will become the receiver transmitter node that is the reason why the logic 0 is called as a dominant logic and logic 1 is called as the recessive logic and similarly in the message id also if you look very carefully message id lower the message id higher will be the message priority that's why if the message id is smaller the more significant bytes are having more zeros as compared to message id having the larger value so as a result of which That message ID will have more priority because the logic 0 is the dominant logic. We will see that how this dominant logic and successive logic works as we go ahead.

And the differential voltage as we mentioned that there is a differential signaling on the CAN high and CAN low channels. So, the actual voltage is the difference of the voltage on the CAN high and CAN low wires. The dominant logic is detected whenever the difference voltage is greater than 0.9 volt and recessive logic is detected whenever the difference voltage is less than 0.5 volt. This indicates it is a logic 0 which has been pushed onto the CAN bus. This indicates the logic 1 or you can say the recessive logic which has been pushed onto the CAN bus.

now as we have known that logic zero is a dominant logic wherever the logic zero is been applied a pulse of 3.5 volt will come on the can high wire and a pulse of 1.5 will be on the can low channel what i mean by that is here this is the dominant logic which is equal to logic zero the moment the controller is trying to put the logic zero logic 0 after the can trans receiver which is the front end for any controller which has been connected to the can bus this particular can trans receiver by once it receives the logic 0 it will make the voltage at the can high bus to be equal to a 3.5 volt and the voltage on this one to be 1.5 volts so this is 3.5 volt this is 1.5 volt And similarly, whenever the logic one is being applied, which is the recessive logic, both the CAN high and CAN low will be at 2.5 volt. And that's when the V difference here is less than equal to less than equal to 0.5 volt, which was shown over here. Similarly, when the logic 0 is there, the voltage difference, you know, if you take the V difference, it is somewhere around equal to 2 volts, which is greater than equal to 0.9 volts.

That's when it is the logic 0, which is being pushed at this data is nothing but the logic 0, while this data is nothing but the logic 1 or the recessive logic, which is being pushed onto the CAN bus. Now this CAN high and CAN low wires are actually the shielded twisted pair cables which will be put onto the area over which the network is being laid down and this is there in order to avoid any interference to the nearby circuit As well make the communication more reliable from interference due to the nearby circuit. So, putting the twisted pair will avoid any interference to nearby circuit and using the differential signaling will give the more reliable communication and against the interference due to the nearby circuit. So, both it provides because of the differential signaling and shielded twisted pair of CAN high and CAN low wires.

then every CAN bus will have the terminating resistance of 120 ohm as you can see over here this have been made so at the two ends of the bus so the bus if it is like this there will be 120 ohm on this side 120 ohm on this side and all the nodes are actually connected together i mean a connector to that can high and can low something like this this is n1 and two and three this is shown over here and one comprises of the controller and the can trans receiver at the front end so it has the two terminating resistance of 120 ohm which is kept at the end of the bus this terminating registers are used to match the impedance of the entire bus and helps in mitigating the signal integrity issues like rise time and false time because this communication is actually happening at 250 kbps range or slightly higher speed range so that's why these terminating resistance are kept to mitigate this signal integrity issues in the CAN channel or CAN bus and in this particular CAN communication actually the node works like it opens the collector configuration what i mean by that is the nodes can able to pull down the CAN bus to dominant logic, but they cannot able to make it to recessive logic. What I mean by that is, let's say there are three nodes connected together, node 1, node 2, if they are connected to the CAN bus, in that case, if any of the nodes is pushing the logic 0 into the bus, it will make the entire bus goes down to 0.

However, in case if they wanted to make the logic of this bus to be equal to 1, then all the nodes must be sending the recessive logic which is 1. So, it works like an N gate kind of thing where we have let's say 3 nodes which are coming over there. There could be all the nodes will be putting 0, 0, 0. All the nodes could be putting this particular case. They will be putting this

different combinations we are writing now in that case the CAN bus logic will be if any of the node is pushing 0 the output of the CAN bus logic will be 0 that means it is a dominant logic here also it is a dominant logic here also it is a dominant logic here also it is a dominant logic here also it is a dominant logic here also it is a dominant logic here also it is a dominant logic however in this case since all the nodes are putting logic 1 In that case, the overall bus logic will be the logic 1. So, you can also imagine the CAN bus to be looking like something like this. Open collector format something like this. this is node one logic node two logic node three logic if i'm applying one one one to this one then this will this particular thing will be zero on this side and as a result of which all the three transistor will be off and that's when the logic one if you say it's five volt is a logic one that will be coming at the output otherwise if any of the nodes will become zero

The output will be 1. Once the output will be 1, any of the transistor turns on and they will pull down the entire line to be the zero voltage. This is how the CAN communication actually works. Now, once we have understood the electrical properties of CAN communication, let us also understand how the data in the CAN communication is being represented. again there were two kinds of data frames which are there one is called as a can 2.a data frame it has the unique identity which is the message id which has the 11 bit identifier that means we can have 2 power 11 distinct messages

distinct messages since we have 11 bit identifier we can separately define 2 raised to power 11 distinct messages only when we are using CAN 2.0a data frame and in this data frame since there are only 2 power 11 distinct messages sometimes when we have more number of messages you require larger identifier so that more different message ids can be defined and that will actually result into CAN 2.0b data frame So let us see how the data frame in the can get represented. What is the meaning of each, you know, segment of that data frame. Now in the CAN 2.0 a data frame what we have is we have first bit which is the start of SOF bit which is also called as the start of frame bit which is used to actually synchronize different nodes which indicates the start of the CAN message actually and this entire thing this entire thing is the CAN data frame and it is having nothing but called as a CAN message. Every things are combined together will called as the CAN message.

Then you have the second thing which is the identifier which is 11 bit identifier in CAN 2.0a. In this 11 bit identifier it stabilizes the priority of the message. So as I told that there are message ID associated with each CAN message. So in this CAN message, this is nothing but the place where your message ID is being incorporated. And that message ID will be nothing but having 11 bit identifier.

That means we can define 2 raise power 11 distinct message IDs. That means we can accommodate 2 raise power 11 distinct messages. And it stabilizes the priority of the message.

The lower the binary value, the higher will be the priority of that particular message. The lower the binary value of identifier of this particular 11 bit, the higher will be the priority of the messages.

and it is primary because the lower the binary value means will be having the 0 coming in first as compared to other message id having higher binary value and that's when since the 0 is there that will dominate the bus as compared to the other message and that's when the node whose message is dominating the CAN bus will be the node which will be actually pushing in the information and while the other nodes will be just receiving that those information and they will get convert into the receiver node while the node whose message has the lower binary value will be owning the bus and will be pushing in the entire CAN message onto the bus and once the entire CAN messages is been put in the next CAN message will then initiate and then again the nodes will fight among each other and whichever has the lowest message id will win the bus and will then push the message into the bus and will be the sender while the other will be just the listener or you can say the receiver or you can say the one who is sending the message will be the master while the other one will be the slaves so that's why it is also the multi master configuration nodes we have So if we take the example let's say we have two nodes they are trying to communicating with the each other using the CAN bus and they both are trying to push the information into the CAN bus which is shown over here and let's say the node N1 is trying to push the CAN message having the message ID which is nothing but defined by 11 bit identifier. So let's say N1 message identifier

Which is an 11-bit identifier, so let us say it is nothing but 0 0 0 0 1 1 1 1 1 0 1. So, it is a—let's say—this is the Node 1 message identifier, which it is trying to push into the bus. Similarly, Node N2 is also having a message identifier. I mean, they are also trying to push the message with an identifier, nothing but given as 0 0 1 1 1 0 0 0 1 1 1. Assume this as an example I am saying. So, the binary value for this is nothing but 125, and the binary value for this is nothing but your 455. The two nodes are trying to push the message with the identifier having the value 125 and 455. Now, since we know that in CAN communication, the messages are sent serially.

So, that means they start sending the most significant bit. So, the first bit has gone, the second bit has gone, and the third bit has gone. Now, if you see, the first and second bits are the same, so both nodes are trying to push the dominant logic into the CAN bus. So, the CAN bus will automatically have the bus logic equal to the dominant bus logic. Now, the moment the third most significant bit is being sent, what happens is that Node 1 is trying to push the dominant logic into the CAN bus, and Node 2 is trying to push the recessive logic into the CAN bus. As a result, what happens is that the bus will now dominate that recessive logic, and the CAN bus logic is nothing but 0. As a result, N2 understands that it is no longer in a position to push the message because the other node is actually on the bus and is trying to push the information.

Thus, they will now turn into the receiver node and will only receive the information placed on the CAN bus from the other node. Now, in this, if you see, as we have told, the message with the lower binary value has the highest priority. So, if we see, the message with the lower binary value has more zeros in the most significant bits. As compared to the message having the higher binary value, so here, if you see, we have in the four most significant bits, we have the zeros. However, in the case of the message from Node N2, which has the higher message ID or the larger message ID, has only two zeros in the most significant bits. Now, as a result, this message, which is put by N1, will have higher priority compared to this message having the message ID pushed by the N2 node. So, this indicates that the lower the binary value, the higher the priority of the message. That means the higher-priority message will be the one staying on the CAN bus compared to the other messages being pushed by the other nodes. So, this is how the identifier defines the priority of messages. And just by changing this 11-bit identifier—which can be done using software—one can easily set the priority of the messages and modify the CAN communication accordingly. That means they can define which message to be given more priority compared to others.

the other messages so it can be done in software so that's why the modification is very easy in case of CAN communication then comes our RTR this is RTR bit now this RTR bit is nothing but let me write RTR This RTR bit, it is nothing but single remote transmission request bit, which indicates whether the frame is the data frame or the remote request frame. Now, what it indicates, this bit is used to request the data from another node onto the CAN bus. Whenever it is logic 0 or the dominant logic, the frame indicates it's the data on the bus. And whenever the RTR bit is recessive bit, the frame indicates the request for the data from the another node that has the message ID.

what it indicates is since it is the remote transmission required it is particularly for that node which has the same message id if you wanted some response from that node so then the message id will have the recessive logic and then the node which is receiving it will then put in the data and whenever the rtr bit is dominant is the data which is been there for the entire bus or you can say it's a kind of a broadcast message to all the nodes which are being connected to the CAN bus. Then we have the IDE bit. It's a single identifier extension bit. which indicates when we have the dominant logic, that indicates it's the CAN 2.0a data frame.

If it has the recessive logic, then it indicates it's the CAN 2.0b or it's the extended CAN data frame. So this CAN 2.0a or 11-bit identifier is also sometimes called as the standard data frame. Standard data frame. and this standard data frame having no extension which is being transmitted into the canberra the r0 is basically the reserved bit the dlc or you can say the data length it is also called as the data length code is the bits which are there to indicate what is the length of the data which is been sent in this particular CAN message so in the CAN message we have some data which is been pushed into it and since maximum we can go from 0 to 8 bytes of data 0 to 8 bytes of data that means we can go from 0 to 64 bits of data in one CAN message

so this dlc bit indicates how many number of bytes of data you are actually sending it through this particular can message which is being pushed onto the canvas so this indicates data length code and since we have maximum number of eight byte of information we can send that's why we require four bit of information because the binary value for it will be one zero zero zero So we require to have the four bits, one, two, three, four, four bits of this particular DLC bits in this particular DLC segment, we require four number of bits to show how many number of bytes of data we are sending it with this scan message. then we have the data which is being embedded in this can message now this data we can actually send maximum up to the 8 bytes of data so this is this entire thing is the can message in the can message we have the data the data are also called as the signals and sometimes also called as the channels so there are different nomenclature for this data or signal or channels and maximum you can send up to 64 bits of data or channel and minimum you can send 0 bits of data that means you can send from 0 to 8 bytes of data And multiple configurations, you can send the data, you can send 64 different data or you can send 64 different signals of 1 bit each or you can send 8 signals of 8 bit each or you can send 1 signal of 64 bit.

So many combinations which are possible in one can message you can just send 64 different data or you can 64 different information or signals with each having one bit of information. You can also have 8 different signals or data with each of 8 bit information. Similarly, you can send one signal of 64 bit information. So you can send any number of data through this CAN messages which is up to maximum up to the 64 bit in one CAN message. Then you have the cyclic redundancy check.

It is basically for error detection and it is mostly a 16 bit number which is been incorporated in the CAN 2.0 data frame. then you have the acknowledgement bit now this is basically the two bit with one bit as the acknowledgement bit and other one is the delimiter bit so every node which is receiving this message obviously this accurate message overwrites this generally the sender will put this acknowledgement bit to one that means at recessive logic and the receiver once it receives the accurate message by checking the CRC or cyclic redundancy check code They will identify it is the accurate message and they will overwrite this recessive bit with the dominant bit and which indicates that it is a error free message which has been sent. So we can say sender put the recessive logic and receiver overwrites with the dominant logic. So this is why this is important after the CRC, these acknowledgement bits are being incorporated.

So one bit is acknowledgement bit, another one is the delimiter for acknowledgement bit. Delimiter is used to give some time for the receiver frame to understand whether the messages are accurate or not and then convert that logic from the recessive logic to the dominant logic. then comes the end of frame bit which is again it's a 7 bit field or you can say 7 bit of data which are kept at the end of the message frame or you can say can frame which indicates the end of the can message and finally you have ifs which is the inter frame space which is again a 7 bit inter frame space which contains the time required by the receiver to move the received message into its buffer So, this is how the CAN 2.0a data frames are being represented. Similarly, we also have CAN 2.0b data frame which is also sometimes called as the extended data frame because in this case we have the 11 bit identifier as in the case what we have seen

in previous scan 2.08 data frame along with this 11-bit identifier there is extended 18-bit identifier which is being added with the 11-bit identifier which makes the 29-bit identifier what it indicates that we can actually accommodate 2 raised to power 29 distinct message ID that indicates We can actually accommodate 2 raised to the power 29 distinct messages. So each

CAN message must have the unique message ID and we can only have 2 raised to the power 29 unique message ID with this CAN 2.0b data frame. Now, in this case, SOF, this 11-bit identifier is the similar bits which has been discussed in CAN 2.0 data frame. In this, there is another bit called SRR bit, which is a substitute remote request bit.

Again, it is a recessive bit which indicates that the CAN message is in the extended data frame and it allows the nodes to recognize the extended CAN frames without needing to know the entire 29-bit. So, after this 11-bit, SRR is been incorporated in this one and then what we have is we have the next one which is IDE which is again the recessive bit this is recessive bit which indicate that after this bit there is 18 bit identifier which is been followed which has to be added with this 11 bit identifier from the your CAN 2.0b data frame and then finally you have the you know complete 29-bit identifier which actually indicates the total priority of the CAN message the lower the binary value larger will be the priority of this message and in case to CAN 2.0b it is and 29-bit identifier and again instead of there R0 we also have one bit which is been incorporated as a reserve bit for future expansion Similarly, we have the data which is again 0 to 8 bytes of data.

We have CRC, we have acknowledgement bit, we have EOF end of frame, we have IFS to provide the sufficient time for the receiver to put those information in the buffer and we have the RTR which is again the remote transmission request in case you are demanding a response from a one particular node so you can put this RTR to appropriate logic such that the other node will understand okay this message ID has been sent and I have to respond to that message with the required data which is been asked for so this is the CAN 2.0 A and CAN 2.0b identifier this is how the data in the CAN is been pushed in it has the maximum 8 bytes of information in one CAN message and those 8 bytes of information is suffixed and prefixed with the required the bits for first reliable and the error-free communication between the two nodes in the system it has identifier which defines the message id it has Data length code which defines the what is the length of data you have been sending and then it has the CRC and acknowledgement bit for making sure the accurate information is being sent. Then in CAN communication there is one important phenomenon which is taking place which is the arbitration.

Now, in the arbitration, as the name suggests, it's the arbitration. It indicates that whenever more than one node is trying to push in the information or trying to send the information. Let's say we have node 1, node 2 and node 3 and they're trying to push in information in the CAN bus. So during that time all the three nodes should be fighting with each other because all the three will be sending the message at the same time. Then there will be arbitration or it's a method by different nodes will define which will own the CAN bus and will send the information on to the bus while the other nodes will then once they decided that if they are not sender node then they will convert themselves to the receiver node and they will just be in the receiving mode receiving the information which is available on the CAN bus.

So, for example, if node 1, node 2, node 3 is trying to send the information on the CAN bus. So, let us see how the things goes around. So, let us say at this point, all the nodes, node 1, node 2, node 3 are pushing logic 0 or the dominant logic onto the CAN bus. Since they are putting dominant logic onto the CAN bus. So, all the three are pushing 0.

So, obviously the CAN bus logic will be the dominant. Dominant logic which is the logic 0 and as a result of which you know your V difference will be somewhere around 2 volt that means greater than 0.9 volt. You know the V difference or you can say the CAN bus voltage level. Voltage level. So we cannot say which node is pushing the information onto the CAN bus.

All the three nodes are pushing in the same data or same bit. So that's why that's not a problem for all the nodes and the CAN bus data has been defined as a logic 0. Then the next bit has been pushed in by all the three nodes and till that time all are the sender node. They all will be sending the information. now in the second bit all the bits are sending 111 all the three nodes are sending 111 that means the logic of the entire node will then be nothing but logic 1 and that's when the v difference is 0 volt which is less than 0.5 volt now when the third bit is still all the three nodes could not decide which one is owning the bus so then the third

bit in the message will be pushed in now when the third bit is been pushed in you can see that the node one is pushing zero the node two is pushing zero however the node three is pushing one so since the two nodes are pushing zero the entire bus voltage will be greater than zero which is again nothing but equal to two volt and the bus will be at the dominant logic zero That is when the node 3 will understand that the node 1 and node 2 is sending the information 0. That's when the node 3 will understand that I am pushing the value 1 in the CAN bus or recessive logic in the CAN bus. But the CAN bus logic is 0. because other nodes are pushing in the zero logic that's when the node 3 realizes okay now I lost the chance I lost the battle and then they will convert it into the sender node and that is when the node 3 will decide that it will become the receiver node because

it is trying to push in logic one but already the canvas logic is dominant logic so that means there is some other node which is pushing the dominant logic or has the message id you can say because that the starting is the message id which which has been pushed in so it has the message id which has a smaller boundary value that means that node 3 realizes the message which has been putting in is of less priority and that's why it will stop sending the message and it will convert into the receiver node and it will now only listens to other nodes which are which are actually defining the logic of the CAN bus then the fourth bit whenever the fourth bit information is been sent in it is again the zero zero logic already the node three is out of the question so node one and node two is still pushing the same value so still node one and node two cannot decide who will be winning the bus so that's why they will leave the logic will still be zero and they will leave the bus like this again here we have two volt coming in Then in the fifth bit, it is the logic 1. All the two nodes will be sending their logic 1. Node 3 has already converted into the receiver node.

Since both are pushing in logic 1, the CAN bus logic will be 1. That is when this is at the 0 volt. then after that in the sixth bit case what happens is that the node one is pushing the zero logic since at least one node is pushing the zero logic which is the dominant logic the logic of the canvas will be zero and the voltage of the canvas will be nothing but greater than 0.91 which is 2 volt and that is when the node 2 understood that i am pushing in the recessive logic but or the CAN bus logic is a dominant logic that means there is some other node who would have already pushed in the logic 0 that's when the node 2 will now understood that it has lost the battle and it will now become the receiver node while the node 1 will still be the sender node and that's when node 1 wins the arbitration that means win the battle between who will be owning the CAN bus and after this whatever the data which has been pushed by node one the only those messages or the those logic will be only with determining the canvas logic because it is zero here canvas zero this is logic one canvas logic is one here it is zero can zero it is zero that means beyond this point

The node 2 realizes that there is some other node which is pushing in the message which has a message ID having lower binary value as compared to its message ID which belongs to the message which has been pushed onto the CAN bus and that's when it's the node 2 which realizes it lost the version and node 1 will finally win the arbitration. and then node 1 will keep on sending the that particular can message and after that only the node 2 and node 3 will then initiate the messages again and again here when whenever the first bit is been sent it is 0 0 so canvas logic is 0 when second bit is there 1 1 so canvas logic is 1 still they could not decide in the third bit whenever the node 2 is sending logic 0 however the node 3 is sending logic 1 but since the load 2 is sending logic 0 it's the CAN bus whose logic level will be defined as a 0 and that is when the node 3 realizes that it has lost the battle and it will now become the receiver node and which also indicates that message id as compared to other message which is being pushed by the node 2 will have the lower message id as compared to the message id with the node 3 sending and that is when the node 2 will win the bus and they will start pushing all the entire can message onto the can bus and it is a message based protocol message priority will be defined in the message itself and any note could be the master note that means any note could be the sender note while other notes will be the receiver note depending upon the priority of the messages they have been putting onto the canvas now

After we know how the CAN data is being presented, after we know whenever the multi nodes are been pushing in the information, which node will be winning the world. Let us also indicate how the CAN nodes or the nodes which are being connected over the CAN bus, how they understand whether the messages which are pushed onto the CAN bus is of the relevance or not. So in that case, there is something called CAN database file, which has been defined. Now, CAN database file or the text file basically which contains the information like CAN messages, signals and their different parameters. For example, let's say one CAN message, let's say CAN message having the message ID equal to 1, message ID equal to 1 has let's say 10 signals.

Could be, you know, 1 bit, 2 bit, 3 bit, anything, the summation should be less than 64 bit. And each signal, if there are 10 signals, so let's say 5, 6, 7, 8, 9, 10. All the 10 signal information like what will be the offset value, what will be the gain value, what will be the real data, what are the different factors we have to do, what is the scaling factor, what is the offset factor, what is the unit, all those parameters will be incorporated corresponding to all the 10 signals. In DC001 also both the nodes which is the EV as well as the EVSE will have the CAN database and in CAN database different messages are there and each communication message has several signals and each signal will have its own parameter defined in the CAN database file. So each CAN message will have multiple signals.

I mean we could have one signal of 64 bit. We could have two signals of 64 let's say 8 bit each we could have 8 signals of 8 bit each so we could have 64 signals of 1 bit each that means we could have 64 information with each information having 1 bit of size and each device manufacturer which is actually been connected to CAN will have the CAN database file which is been put in along with the project and for each signals which are there the CAN database file will define the rules for conversion to the required engineering units so following data will be stored in the CAN databases we have the message name we have the message id we have the dlc bit of that message we have different signals names We have the location of different signals, that means from which bit it will be starting to which bit it will be ending.

We have different size of signals, we can say channels or signals. In the message, that means how many number of signals, what is the length of the signal, what is the start location of the signal, everything will be incorporated. What is the signal type, whether it is a character, whether it is an integer, different types it has been defined. what is the byte order that means how the signals are been arranged whether it is a little engine or the big engine they will include this scaling factor and different unit string that means what will be the unit of that particular signal which has been there they will have the range what is the minimum range and the maximum value they have the offset value so that means if you wanted to extract the actual data so actual data will be nothing but the raw data which you are receiving multiplied by scaling factor which we are getting which is incorporated in the CAN message so this raw data is the data received from message received or data extracted from message so we have this one plus scaling factor from database

for corresponding to that signal and then this will be plus with offset value corresponding to that signal from data database you will get and this database will be kept in the node itself so whenever one node is winning the bus and they are pushing the message so every node will see the message and they will decide whether the message is of their relevance or not and how they define they will take the message they will extract the message id and they will compare it within the database that whether that message id is there or not if it is there that means it is a relevant message if it is not there they will decide it is a irrelevant message for them and they will just ignore the next sets of data which is there in the can message so this is how the CAN database file plays an important role so in a CAN network we have the CAN bus which is comprises of CAN high and CAN low wire we have the CAN database file which has to be incorporated in each node and we have the another thing is what will be the CAN data frame which you are taking either CAN 2.0a or CAN 2.0b and accordingly they can do the CAN communication between the two nodes in DC001 charging system we also have the CAN database which has different messages at different stages of charging they have their message id using the parameter data unit format which we will discuss in the next class they have different CAN databases which is been there which comprises of CAN message having different CAN signals defining the length of the signal defining the location of the signal start bit or what is the indication of each signal they will represent they will also tell what is the scaling factor what is the offset factor and that is how once you receive the CAN message you extract the data from the data you

take out the different signals which are there and that signal will corresponding to that raw data from the raw data and using the information in the can database file you will actually extract actual data so this is how the can communication works we will discuss and we will solve some examples to make you understand how the messages are received and how they get interpreted in case of dc001 charger in the next class till then thank you