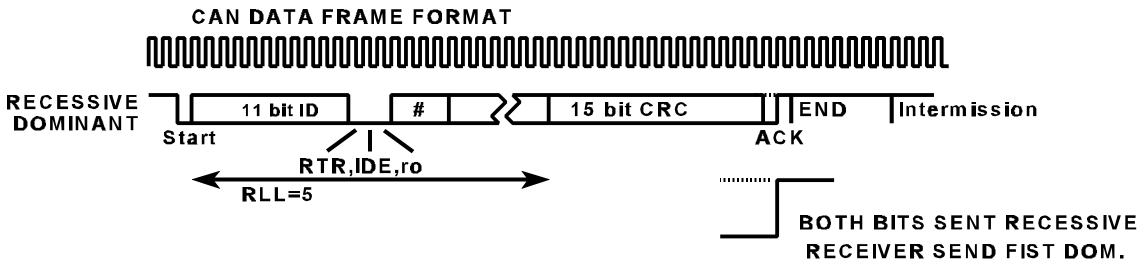
THE CONTROLLER AREA NETWORK (CAN)

We shall discuss some of the basics of this type of data network before looking at the application. Data is sent in FRAMES as is RS232 data and, in fact, they both begin with a START bit. The format (short form) for this data-link protocol is as shown below.

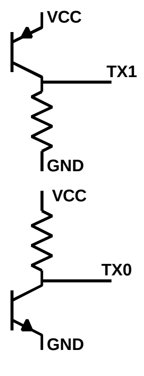


In this protocol, 1's and 0's are replaced with "recessive" and "dominate." Why this change will be seen later. We shall describe this format in terms of the RS232 format.

Start Bit: Both use one start bit.		
Data:	RS232 uses 5 to 8 bits.	CAN uses 0 to 8 bytes
Error:	RS232 may use a parity bit	CAN uses a 15 bit CRC
End:	RS232 may use 1 or 2	CAN uses 7 (plus 3 more).

In addition

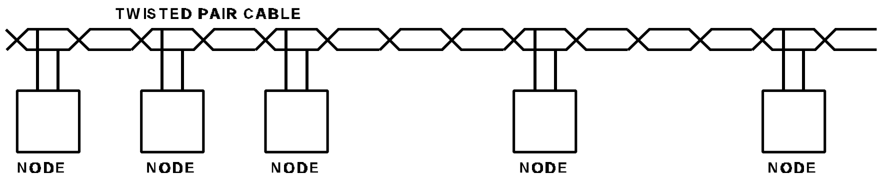
CAN has a variable number of data bits (bytes) as specified in the four bits marked #.
The data bytes follow this.
CAN has an IDENTIFIER to tell where the data came from (11 bits).
RTR, IDE and r0 are always zero in a data frame, ACK will be explained later.
CAN uses RLL coding of length five for bit up to but not including CRC
RS232C with so few bits does not need this.
In this case RLL is defined as "after 5 bits at the same logic level, stuff in a bit at the opposite level (stuff in a 0 after 5 ones and vice-versa).



Let's now consider *dominant* and *recessive*. Unlike RS232 where data is sent via one line and ground, CAN uses a "push-pull" output that runs a twisted pair cable. To send a "Dominate" bit, both transistors are turned on so that TX0 goes low and TX1 goes high. To send a "Recessive bit", both transistors are turned off so that TX1 goes low and TXO goes high.

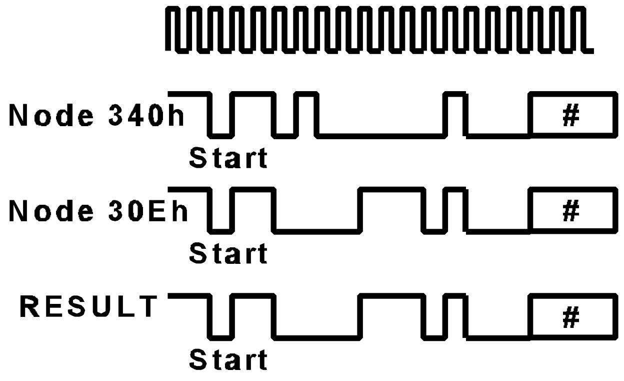
There are two reasons for this configuration. One is that the CAN network was develped in Germany by Bosch for use in motor vehicles and has now been standardized for that use. Motor vehicles are very hostile toward communication signals and the balanced twisted pair line helps to minimize noise problems. In addition, CAN is NOT a point -to-point protocol like RS232C. It is a CSMA-CR network - "Carrier Sense, Multiple Access, with Collision Resolution. Lets take a closer look!

Instead of "nodes" at each end of the cable, there are nodes (many nodes) positioned along the cable.



With multiple nodes,
a MULTIPLE ACCESS network is needed where any node may send (and receive) data over the BUS. Since two nodes can not send data at the same time, each node must be able to sense if data is currently being sent (Carrier Sense). In this case, since a RLL of 5 is used, the seven "stop" bits at the end signal "no carrier" and that transmission is ok.

Still, two nodes might begin at (almost) the same time. What happens then? Suppose that a node with an ID of 340 (hex) and another node with an ID of 30E (hex) both start sending at the same time?

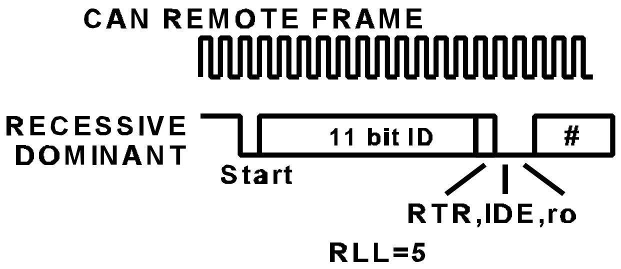


Since dominate bits take precedence over recessive bits, when the third ID bit is sent, node 340 sends a recessive bit and node 30E sends a dominate bit which overrides the recessive one. Node 30E has no way of knowing this has happened but node 340's receiver senses the dominate bit and knows that so other node is also transmitting. It, therefore, is obliged to disconnect itself from the network and allow the other node to continue. Since two nodes may not have the same ID, a collision will be sensed by the time the last ID bit is sent.

Now, we still have a few unanswered (unasked?) questions. We can tell which node is sending the data, but who is supposed to receive it? Did they receive it? Why was the data sent at all?

There are other types of frames besides data frames. A REMOTE frame looks like a DATA frame except that:

(a) no data bits are sent.
(b) the RTR bit is sent recessive.
(c) The ID # is the number of the node that should supply data.



If node 340 wants data from node 304, it sends a REMOTE frame with address 304. Node 304 responds by sending a DATA frame with address 304. Any other node that wants the data from 304 can also receive it.

Did the data get there? Refer to the figure at the beginning of the article. Two bits near the end are for ACKnowledge. The sending node sends both passive. A receiving node, <u>for that one bit time</u> sends a dominate bit for the first ACK bit. The sending node can sense this and know that (at least) one node received the message.

We have now discussed the basic protocol. Let's take a look at the overall network. Since this network was designed for automotive applications, it has some special characteristics (in addition to those already noted). Automotive applications? In a recent issue of TI's "Integration Magazine," a diagram was shown illustrating (at least) 18 different applications of microprocessors in automobiles (ingitition/injection system, transmission, ride control, braking system, etc.). It is necessary that some of these processors obtain information from others. The transmission controller, for example, needs to obtain the current RPM from the instrumentation controller, etc.

Because many of these are relatively small microprocessors, they do not possess the speed necessary to handle networking - a controller chip is needed. The chip must be relatively small and inexpensive. One such chip is Intel's 82527. We will not go into the programming aspects of this chip (you may never see one) but we will go into the operational aspects.

RECEIVING DATA. Receiving data is not like with a UART that is supposed to receive all data sent to it. In this case, a given node should receive only certain messages. The 82527 has sufficient memory capacity to store a total of 15 frames (either frames waiting to be sent or frames received). When the ID field of a message is sent over the bus, the receiver compares it with a mask register and an id register which specify which bits of the message ID must match bits programmed into the id register. Only those messages which match will be "received" (i.e. loaded into memory and, eventually, sent to the CPU.

SENDING DATA. Data may be sent up to be sent but held until the desired time. By setting a bit in a control register, the message will be sent at the first available idle time on the bus.