Voice over CAN

Mahesh Mahajan, Monoj Baruah, Srinivas T, Suresh Sureddi
Wipro Technologies, Bangalore, India

There are numerous applications of CAN carrying real time data. However CAN is still not become popular for carrying voice streams data. This paper delves on the possibility of transmitting voice streams over CAN. A detail analysis of various CODEC algorithms such as G.726, G.729 and G.711 shows how effectively CAN bandwidth can be used for voice transmission. The solution is realized with a specially suited proprietary protocol for Voice data. This paper gives the findings on following: CODEC suitable for voice communication over CAN, MIPS and memory requirements, effective CAN bus lengths, CAN bus loads and bandwidth calculations, buffer management, Quality of Service (QoS), assignment of CAN message identifiers.

Brief Overview of CAN

The Controller Area Network (CAN) was originally developed by BOSCH GmbH as a advanced serial communications protocol to pass information between controllers on an automotive network and thus reduce the growing complexity of the wiring harness on modern car design.

The few important CAN features with respect to voice communication are as follows:

- CAN is a multi-master bus with an open, linear structure of connected nodes. The number of nodes is not limited by the protocol. The nodes don't have a specific address. Instead the address information is contained in the identifiers of the transmitted message, indicating the message content and the priority of the message. The updated CAN specification provides for either 11 ID bits or for a larger identifier range using 29 bits. The 11-bit ID format is referred to as the standard format and is governed by the CAN standard 1.2/2.0A, while the 29-bit ID is referred to as the extended format. CAN standard 2.0B cater to the both 11 bit and 29-bit ID's.
- CAN use a highly sophisticated error detection protocol. Probability of undetected data corruption in CAN is ~1 $*10^{-13}$ per CAN message transmission.
- Data rate can be configured up to 1MBit/s. At lower bit rate, bus length can go up to 5 KMs.
- CAN protocol have a guaranteed transmission times. It has no overhead and bus performance is independent of number of participants.

For voice transmission, which one to opt for? Standard or Extended format?

The standard format has 11-bit identifier while extended CAN identifier uses 29-bit identifier. For just carrying voice over CAN, many identifiers are not needed hence 11-bit identifier is more than sufficient. The same is used for prototyping.

Is the bandwidth sufficient for voice transmission?

In order to determine this, following calculations are done:

Single CAN frame consists of

1 Start bit + 11 Identifier bits + 1 RTR bit + 6 Control bits + 64 Data bits + 15 CRC bits + 14 Stuff bits (varies) + 1 CRC delimiter + 1 ACK slot + 1 ACK delimiter + 7 EOF bits + 3 IFS (Inter Frame Space) bits

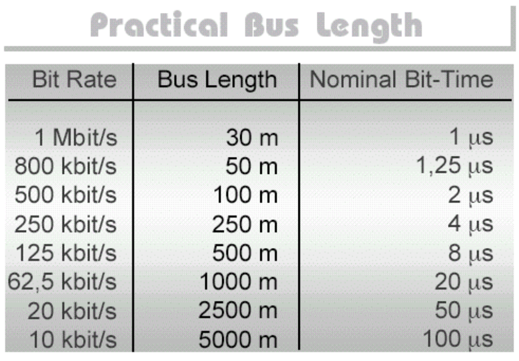
= Total 125 bits.

(Note: The number of stuff bits inserted depends on the data transmitted. Here on an average 14 stuff bits is considered)

For example, for a data transfer rate of 125 Kbit/s, for 500 m bus length, single message frame takes ~ 1 ms to transmit. Or in another way:

Number of messages / second = 125,000/125 = 1000 messages /sec.

Thus a single CAN message frame containing 8 bytes of voice data takes ~ 1 ms. Before actually analyzing whether this transfer rate is sufficient for voice transmission or not, following are the alternatives examined to embed voice payload into CAN frames.


Practical Bus Length

Bit Rate	Bus Length	Nominal Bit-Time
1 Mbit/s	30 m	1 µs
800 kbit/s	50 m	1,25 µs
500 kbit/s	100 m	2 µs
250 kbit/s	250 m	4 µs
125 kbit/s	500 m	8 µs
62,5 kbit/s	1000 m	20 µs
20 kbit/s	2500 m	50 µs
10 kbit/s	5000 m	100 µs

Table 1: CAN bit rate © CiA

Alternative 1:

In a message-oriented communication like CAN, the voice data is recognized by CAN Object Identifiers (COB). In this alternative, first the COB id is assigned to indicate total length of the message. On the receiver side after receiving first packet, the receiver will wait to receive rest of the packets as specified in message length.

Alternative 2:

In the proprietary application layer protocol solution, few bytes in each packet could be reserved as message header and rest of the byte can contain voice data. Advantage of this method is that there is no need of extra COB-ID needed to transmit indicating length of message as explained in previous method. However with this solution, the effective bandwidth gets reduced, as in every packet some bytes will be reserved for message header instead of voice data.

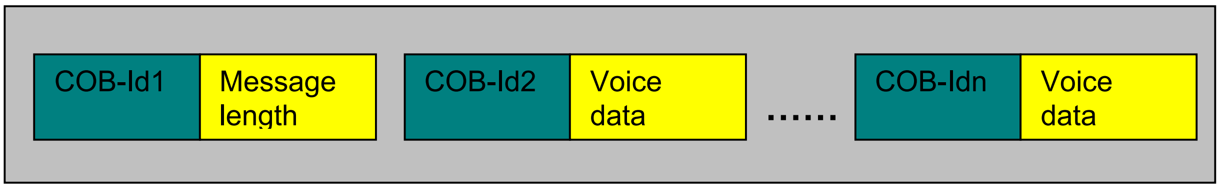


Figure 1: Alternative 1 for voice packet communication

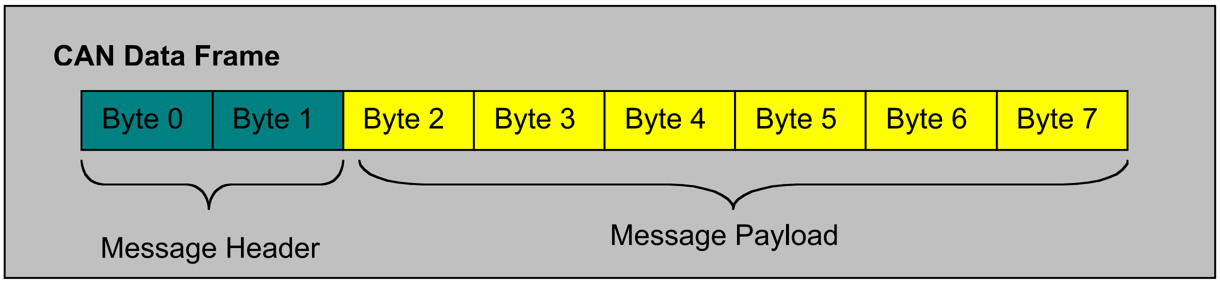


Figure 2: Alternative 2 for voice packet communication

Considering 2 bytes for message header, rest of the six bytes will then carry actual data. This is represented as shown in Figure 2.

This 2-byte message header can contain proprietary header information, node identification, control information, message number or message length, service code etc. Depending upon the fields, the number of bytes for header can be increased or decreased.

Does it need CODEC with higher compression ratio?

Considering a voice payload of 20 ms, 160 bytes are needed to be transmitted. To transmit 160 bytes of data with alternative 2 above (i.e. 6 bytes of voice payload in each packet), approximately 26 packets are required. (160 bytes gets divided into 6 bytes of each packet; ~ (160 / 6) = 26.66)

Each frame consists of 125 bits. Thus for 20 ms voice payload, bits required to be transmitted are: 3250 (26 * 125) bits.

To calculate per second rate: (3250 * 50) = 166625 bits /s or 166.6 Kbps.

Hence with G.711 CODEC, up to 250 meters voice data can be easily transmitted. (Refer to the table 1)

However with other high compression CODECs, CAN bus can easily be extended to longer distance. The calculations done are summarized in table 2 for various CODEC.

CODEC	Peak Rate (Kbps)	Packet Size (Bytes for 20 ms)	CAN Bandwidth required
G.711 (PCM)	64 Kbps	160	166625 bits / s (166.6 kbps)
G.726 (ADPCM)	32 Kbps	80	83333 bits/s (83.33 Kbps)
G.728	16 Kbps	40	41666 bits /s (41.66 Kbps)
G.729	8 Kbps	20	20833 bits /s (20.83 Kbps)

Table 2: Compression ratio vs. CAN bandwidth requirement

Is RTP type of protocol required?

In VOIP technology, RTP protocol is used to transmit & control the real time voice streams. However, standard RTP stack functionality is not needed to transmit voice over CAN. Following are some of the points in this regard:

- RTP itself does not provide any mechanism to ensure timely delivery nor provide guarantee of quality of service but relies on lower-layer services to do so. In our case, CAN layer will be responsible for timely and guaranteed delivery of messages.
- RTP provides functionality suited for carrying real-time contents e.g., a timestamp and control mechanisms, which is possible and can be implemented with CAN protocol.
- The sequence numbers included in RTP allow the receiver to reconstruct the sender's packet sequence. This is required in IP network as packets takes different route in an IP network and they may arrive at different time at the receiver end. In case of CAN, such a situation does not arise.
- Most of the RTP header fields such as Padding, Header extension, Marker, Payload type, Sequence number and time stamp are of little use in CAN network.

Hence RTP type of protocol is not required for CAN voice transmission.

The mechanism proposed and used is as follows:

The control of the bus can be shared by both the nodes. Each node will take control of the bus periodically say after every 10 ms. This way both the nodes will get access to the bus to share the information. Transmit and receive buffers will be available at both side. The voice messages will be pumped into these buffers. When transmit buffer event is created, the data in the buffer will be put onto the bus. If the bus is controlled effectively, bandwidth will not be wasted in arbitration.

Alternative to this is to use the clock of both CPU's to synchronize the voice data transfer.

Architecture and Setup

To realize voice over CAN implementation, CAN evaluation boards having DSP processors are used. The DSP processor has integrated CAN controller and on chip ADC. The DAC is external to the DSP processor. Instead of using speaker and microphone, which might have added complexity in the interface hardware design and issues such as echo suppression etc., prototyping is done with headphones. The headphones are interfaced with ADC and DAC on both the boards. The CAN PCI card is used along with CAN analyzer to debug the CAN network and analyze the packets on the CAN bus. Figure 3 depicts the setup.

Some of main modules in the realized application are as follows (refer to figure 4):

MicroCANopen stack: This module represents a higher protocol layer over CAN. This module interacts directly with CAN driver. Application layer receives messages from MicroCANopen stack and route them to appropriate application modules. Similarly transmit messages received from various application modules are routed to MicroCANopen stack.

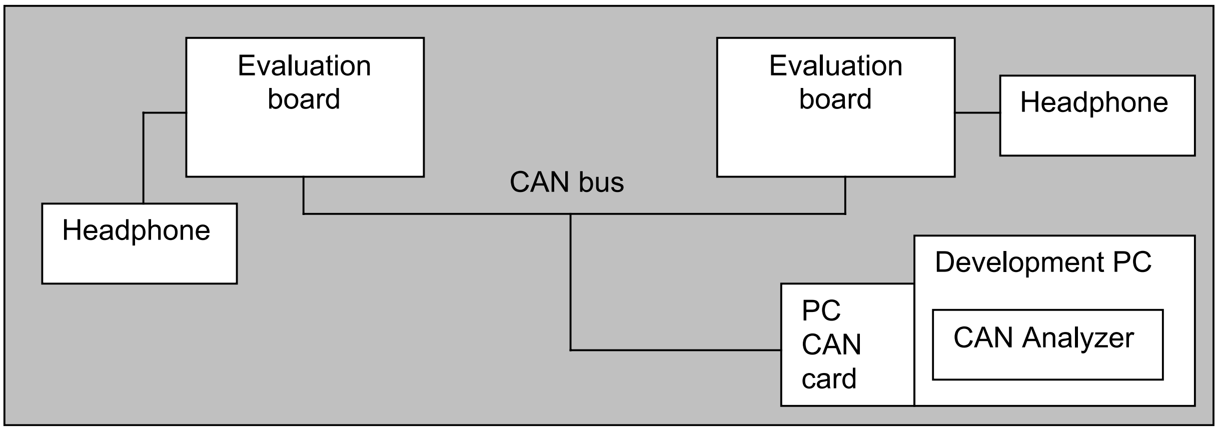


Figure 3: Setup for voice communication over CAN

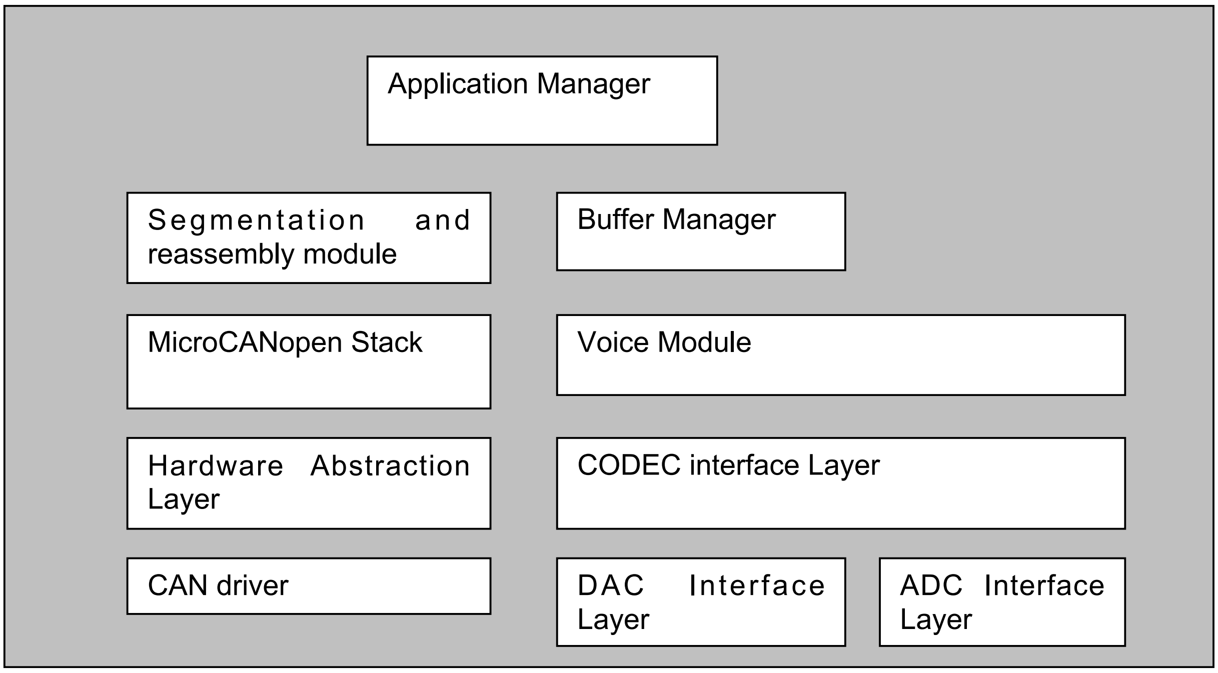


Figure 4: Software blocks

Voice Module: This module collects the raw voice samples from ADC interface layer, encode them and put in transmit buffer. Similarly get decoded data from receive buffer, decode it and send it to DAC interface layer. This module also performs control functions like speaker volume & microphone gain adjustments.

ADC & DAC Interface Layer: This module provides interface to collect samples from ADC driver and similarly provides interface to send samples to DAC driver.

CODEC Interface Layer: This module is a CODEC abstraction layer. This layer is for interfacing different CODECs to ease the performance analysis.

Buffer Manager: This module provides interface between Voice Module and Application Manager. It maintains voice transmit buffers and voice receive buffers.

Segmentation & Reassembly Layer: Application data, which is to be sent to CAN lower layer is formatted and segmented. Similarly receiving data from CAN lower layer is reassembled and formatted before giving it to application layer.

Result of Performance Test:

With the help of G.711 CODEC, voice communication for a distance of 200 meters is achieved successfully.

Quality of voice against different bus lengths is also measured. The table 3 shows the various performance parameters for 10 meters bus length.

BAUD RATE	**BUS LOAD (%)**	**MEMORY**	**QUALITY OF NOISE (in the scale of 0-5)**		
			Noise (0-max. noise, 5-min. noise	**Distortion (0-max. distortion, 5-min. distortion)**	**Volume (0-low volume, 5-high volume)**
1 Mbps	26	Code: 150 KB Data: 11KB	3.5	3.5	4
500 Kbps	33	Same as above	3.5	3	4
250 Kbps	99	Same as above	3.5	1.5	4
125 Kbps	99	Same as above	0.5	1	4

Table 3: Performance results

Conclusion:

By this innovative prototyping, it's successfully proved that voice can easily be transmitted over CAN bus. Even the application programming is much simpler than the IP protocol (VOIP) due to in built features of CAN protocol.

This solution will be immensely useful in situations, where already the CAN bus exists and instead of having different media for voice, the same CAN bus can be extended for voice transmission.

In some automation applications, in addition to process data, a voice communication is desired. This solution is suitable in application areas of factory automation, building automation, vehicles and transportation etc.

Mahesh Mahajan
Wipro Technologies
1300, Crittenden Lane, 2nd Floor
Mountain View, CA, 94043
USA
Phone: 650-316-3555
Fax: 650-316-3467
E-mail: Mahesh.mahajan@wipro.com
Website: www.wipro.com

Monoj K. Baruah
Wipro Technologies
271 Hosur Main Road, Bangalore, India
Phone: 91 80 5539134
Fax: 91 80 5539701
E-mail: monoj.baruah@wipro.com

Srinivas T
Wipro Technologies
271 Hosur Main Road, Bangalore, India
Phone: 91 80 5539134
Fax: 91 80 5539701
E-mail: srinivas.tangudu@wipro.com

Suresh Sureddi
Wipro Technologies
271 Hosur Main Road, Bangalore, India
Phone: 91 80 5539134
Fax: 91 80 5539701
E-mail: suresh.surredi@wipro.com