iCC 1995

2nd international CAN Conference

in London (United Kingdom)

Sponsored by

Motorola Semiconductor
National Semiconductor
Philips Semiconductors

Organized by

CAN in Automation (CiA)
international users and manufacturers group
Am Weichselgarten 26
D-91058 Erlangen
Phone +49-9131-69086-0
Fax +49-9131-69086-79
Email:headquarters@can-cia.de
URL: http://www.can-cia.de

James J. Dattolo
Applications Engineer
Dearborn Group Technology

Society of Automotive Engineers (SAE) Implementation of CAN for Heavy Duty Truck and Bus Market -- Specification J1939

The Society of Automotive Engineers' Heavy Duty Truck and Bus Division created a committee to form a standard for multiplexed control busses in their vehicles. The architecture of these vehicles are complex in nature, comprised of several modules (cab, trailer1, trailer2, etc.). Therefore they chose CAN as the basis of their standard, specifically version 2.0B, for the 29 bit identifiers. Several documents were created that built upon the Bosch specifications but details all seven layers of the ISO/OSI (Open System Interconnect) model.

This paper will show how and why certain implementations were used. Using 29 bit identifiers allowed segmentation of the identifier into content, source, destination address, etc. The physical layer is also specified; it is similar to ISO 11898. The need for very fast transmission rates, typically under 20 milliseconds, forced the application layer to pack similar data inside one CAN message. The process of decoding these messages is difficult because in some cases one data parameter may affect how other parameters are decoded. Due to this, several tools have been created to help facilitate decoding of messages.

Introduction

The SAE formed a committee to create a standard for a control bus in heavy trucks and busses. Standard practice J1939 was the result. The standard has started where older, slower UART (Universal Asynchronous Receiver Transmitter) protocols have left off, namely J1708, J1587, and

J1922. The standard is comprised of a set of documents including one or more for each layer of the ISO/OSI (Open System Interconnect) model. The committee made provisions for a global specification as well as sub specifications. Currently there is a sub specification for agricultural equipment.

Since these systems have the potential for many intelligent sub systems, the designers of J1939 have chosen the 29 bit version of CAN, V2.0B. The CAN identifier is broken down into the following sections: Priority, Data Page, PDU (Protocol Data Unit) Format, PDU Specific, and Source Address. J1939 defines two different PDU Format types; one's specific field represents a destination ID, and the other allows for a extension of the format field. Depending on the identifier, namely the PDU fields and Data Page, the data portion of the CAN frame may be decoded.

The Application Layer document, J1939/7, defines how parameters are converted into raw binary and packed into each CAN frame. In addition to packing data, the value of one parameter may determine if some parameters are present in the message or if those parameters are to be ignored by the receiver. The J1939Tool from the Dearborn Group, as well as other tools on the market, allow engineers to quickly build and decode messages, both offline and in real-time acquisition of messages on a J1939 bus.

J1939 Document Organization

The actual J1939 Document is the top level of a hierarchy of related documents. These documents are designated by a systematic nomenclature given by J1939/NX, where X is the applicable Industry and N the OSI layer number. N=1 is used to designate global Truck and Bus functions. Another example would be for North American Agricultural Equipment (N=2). The top level document, for agricultural equipment, would be connoted J1939/02 and would contain a listing for all related documents. The current complication of documents is listed in Table 1.

Document	Description
J1939	Top level document that describes the network in general, the OSI layering structure, the subordinate document structure and provides control for all preassigned values and names.
J1939/01	Application Document for Truck and Bus Control and Communications
J1939/02	Application Document for Agricultural Equipment Control and Communications
J1939/11	Physical Layer, 250K Bits/sec, Shielded Twisted Pair
J1939/21	CAN 29 bit Identifier Data Link Layer
J1939/31	Truck and Bus Network Layer document
J1939/4X	Transport Layer Document has not been defined yet
J1939/5X	Session Layer Document has not been defined yet
J1939/6X	Presentation Layer Document has not been defined yet
J1939/71	Vehicle Application Layer Document
J1939/72	Virtual Terminal Application Layer Document
J1939/81	J1939 Network Management Document

Table 1 List Of All Related J1939 Documents

Why 29 Bit Identifiers?

In a heavy truck or farm implement, there can be many different modules that govern the operation of the machinery. These modules have data that can be grouped with others of similar functionality. Control strategies are often performed at central computers in the vehicle, leaving the data collection and any preprocessing to sub systems, often intelligent sensors. The source of the data is often important to the control strategies. In some cases, it makes sense for a specific module to receive a particular piece of information. Source and destination addresses allow for these situations. A priority field at the beginning of the identifier would give important messages a better chance of transmission on time. Also, priorities may be changed dynamically during certain

critical situations. The developers of J1939, in an effort to reduce overhead and used bus bandwidth, proposed to pack messages with similar data and update rates into one message. To do this, additional bits of information are needed to tell the receiver what type of message it is receiving.

Another argument for 29 bit identifiers is the ability for a tractor to have may trailers hooked up at once. Source and destination address possibilities can be quite large. Standard 11 bit CAN just does not have enough bits available for decoding the message quickly. Another drawback is the encoding schemes have to change every time another type of message or module is created. 29 bit CAN holds the key to quick decoding and flexibility in adding messages and types of modules without changing the identifier creation strategies. J1939 has left many spaces to allow future designers to fill.

Currently there are two CAN 2.0B compliant devices on the market, the Intel 82527, and Seiman's 80C167. Both have the ability to sort messages into 15 predefined message boxes. They also have one global mask and a separate mask for message 15. The masks allow the module to sort messages ignoring certain fields, such as source and priority. The module only needs to be concerned with the PDU and Data Page fields; These may be looked up in a table to determine relevant parameters.

Data Link Layer Document J1939/21

The J1939/21 adds to the existing CAN 2.0B specifications for the CAN Data Link Layer. The document is primarily concerned with the makeup of the 29 bit identifier field and breaks it into separate fields. Table 2 illustrates the relationship between the fields and the 29 bit identifier.

CAN Identifier Bit	J1939 Bit
ID 29	Priority 3
ID 28	Priority 2
ID 27	Priority 1
ID 26	Reserved
ID 25	Data Page
ID 24	PDU Format 8
ID 23	PDU Format 7
ID 22	PDU Format 6
ID 21	PDU Format 5
ID 20	PDU Format 4
ID 19	PDU Format 3
ID 18	PDU Format 2
ID 17	PDU Format 1
ID 16	PDU Specific 8
ID 15	PDU Specific 7
ID 14	PDU Specific 6
ID 13	PDU Specific 5
ID 12	PDU Specific 4
ID 11	PDU Specific 3
ID 10	PDU Specific 2
ID 9	PDU Specific 1
ID 8	Source Address 8
ID 7	Source Address 7
ID 6	Source Address 6
ID 5	Source Address 5
ID 4	Source Address 4
ID 3	Source Address 3
ID 2	Source Address 2
ID 1	Source Address 1

Table 2 Relationship between CAN 29 Bit Identifier and J1939 Assignments

While according to the CAN data link layer the entire 29 bit field is used to determine priority, in J1939 the 3 bit Priority field is used to modify the priority inherent in the rest of the message. The J1939 committee has set aside a single reserved bit for future expansion of identifiers. The Data Page bit is used to distinguish the messages that are presently being defined and those that will be defined in the future. The Protocol Data Unit (PDU) Format field is an 8 bit field that identifies what data is contained within the message. Depending on the value of the PDU Format field, the 8 bit PDU Specific field takes on either a Destination Address or Group Extension (additional 8 bits of PDU Format); Destination Address is present when PDU Format is between 0 and 239, while the Group Extension is present when PDU Format is between 240 and 255. The last 8 bits define the Source Address of the sending node. For a given network every address must be unique. Therefore, only 256 different nodes may be present on any on J1939 network.

Currently there are five J1939 message types supported; Commands, Requests, Broadcasts/Responses, Acknowledgments, and Group Functions. Commands are destination specific and are used to cause some specific action to happen at the destination. Requests are used to gather information; they may be global or destination specific. Broadcast/Responses can either be an unsolicited broadcast of information or a response to a Command or Request. Acknowledgments are responses to specific commands; they may be either positive or negative. Group functions are used for special groups of functions, such as network functions, proprietary messages, and multipacket transport functions.

Some messages have been defined to contain more than 8 bytes of information. In this case the data must be split between multiple CAN data frames. The rules governing the handshaking and packetization are currently contained in the J1939/21 Data Link document; these rules make up the J1939 Transport Protocol. Only one multipacket, global destination, message may be sent from a given device at a given time. However, receiving nodes must recognize that multiple multipacket messages can be received interspersed with one another from different sources. Currently the J1939 Committee is balloting a revised Transport Protocol.

In addition to the messages defined by J1939's Application Layer document (J1939/7), customer proprietary messages may be constructed as follows. 29 bit destination specific proprietary messages are to be constructed with DP = 0 and PDU Format = 239. Other non-destination specific messages are to use DP = 0, PDU Format = 255, and PDU Specific = 255. The other fields must still follow J1939 specifications. 11 bit CAN 2.0A messages can also be used as proprietary messages. The 11 bit identifiers must comply with the format found in Table 3. This allows for proprietary communications to happen without conflict due to different manufacturers making modules.

CAN Identifier Bit	J1939 Bit
ID 11	Priority 3
ID 10	Priority 2
ID 9	Priority 1
ID 8	Source Address 8
ID 7	Source Address 7
ID 6	Source Address 6
ID 5	Source Address 5
ID 4	Source Address 4
ID 3	Source Address 3
ID 2	Source Address 2
ID 1	Source Address 1

Table 3 11 Bit Identifier Assignments

Application Layer Document J1939/7X

The J1939 Application Layer document specifies how to convert parameters to and from raw binary and how to pack them into the CAN frame. J1939 uses the Least Significant Byte first or Intel format to store the raw data. In J1939, there are three basic types of data: normal or floating point, control/discrete, and ASCII.

Normal data parameters, such as output shaft speed or retarder torque, take on a range of floating point values. These values can be converted into raw binary by using the J1939 defined resolution and offset values. These are defined for each parameter but often take the same form for similar units, i.e., Percent Torque or RPM. In addition to linear scaling, two discrete values that normal data parameters can take on are an error state (ERROR) and not available (NA). These values do not use the resolution or offset values and can be encoded in binary depending on the size of the normal data parameter in bytes. Table 4 shows the valid ranges of normal data parameters and where ERROR and NA are encoded.

Range Name	1 Byte	2 Bytes	4 Bytes	ASCII
Valid Signal	00 to FA	0000 to FAFF	00000000 to FAFFFFFF	01 to FE
Reserved	FB to FD	FB00 to FDFF	FB000000 to FDFFFFFF	none
ERROR	FE	FExx	FExxxxxx	00
NA	FF	FFxx	FFxxxxxx	FF

Table 4 Normal Data Valid Ranges (From J1939/71)

Control/Discrete parameters are bit level parameters. They are always found within byte boundaries and are often grouped together in the first byte of data. Discrete parameters are usually measured in response to a request, while Control parameters tend to turn on and off functions at the destination. J1939 defines the states for each parameter within the J1939/7X document but gives recommendations for the meanings. Tables 5 and 6 give the transmitted ranges of Discrete and Control Parameters.

Range Name	Value
Disabled (Off, Passive, etc.)	00
Enabled (On, Active, etc.)	01
Error Indicator	10
Not Available or not installed.	11

Table 5 Transmitted Values for Discrete Parameters (From J1939/71)

Range Name	Value
Command to Disable (Turn off, etc.)	00
Command to Enable (Turn on, etc.)	01
Reserved	10
Don't care /Take no action (Leave function as is)	11

Table 6 Transmitted Values for Control Commands (From J1939/71)

Discrete parameters can also have an additional purpose. Depending on the value of the parameter, certain other parameters may be decoded differently. Some parameters may be temporally disabled for this message or replaced entirely. This is referred to as a PGN (Parameter Group Number) extender or Modal Dependencies. In this way, multiple messages may be mapped onto one CAN identifier. At this time only one message has been identified, Torque Speed Control One(TSC1). Depending on the state of the Override Control Mode parameter the message may take on a control or limiting functions.

ASCII parameters are often associated with multipacket messages. Examples are Vehicle Identification Number (VIN) and Make/Model/Serial Number information. ASCII fields are usually delimited with a ASCII "*" character and allow for variable length fields.

The current organization of the document places the messages and parameters in different sections and cross-references them with a X.X.X.X type numbering system. This system is cumbersome and time consuming for the engineer decoding messages. There are now several software tools to help engineers build messages as well as decode them.

Example Messages

An example conversation between an engine and the transmission can be found in Table 7. The transmission is attempting to shift to a higher gear. XXXX and YY denote some speed or torque

that can take on any value depending on the current gear and throttle position. The Raw Data column contains example data, all unknown information has been set to the NA state.

PGN	Source	Destination	Action/Function	Raw Data
ETC 1	Trans	Engine/ASR	Decision to Shift (Shift in Progress)	DF FF FF FF FF FF FF FF
TSC 1	Trans	Engine	Override Priority to High and Control Torque to 0	DE FF FF 00 FF FF FF FF
TSC 1	Trans	Retarder	Disable Control Mode Torque = 0	DC FF FF 00 FF FF FF FF
EEC 1	Engine	Trans	Torque = 0	F6 FF 7D FF FF FF FF FF
TSC 1	Trans	Engine	Control Speed to XXXX RPM	DD XX XX FF FF FF FF FF
EEC 1	Engine	Trans	Speed = XXXX	F6 XX XX YY FF FF FF FF
TSC 1	Trans	Engine	Speed/Torque Limit Override Priority to Highest	CF XX XX YY FF FF FF FF
ETC 1	Trans	ASR	Allow ASR	DF XX XX YY FF FF FF FF

Table 7 Example J1939 Messages (From J1939 Document)

Tools

Due to the complex nature of building and decoding messages, a variety of tools are being created to help engineers. Currently, there are no commercially available products that meet all of J1939 requirements. CANalyzer from Vector allows users to build generic application layers for CAN and use them to record and transmit messages. CANalyzer does not support the ability for ERROR and NA conditions on normal data parameters. Also Control and Discrete parameters are shown in bit form only; the user still has to look up the definition from the J1939/7X documents. The Dearborn Group is now beta testing its implementation of a J1939 program to allow users to receive, decode, and transmit messages. It is called the J1939Tool and it runs user Microsoft Windowsä

The J1939Tool is broken up into the following sections: Data Dictionary, Message Monitoring, Message Generation, Message Manipulation, and User Interface. The J1939Tool allows users to create data dictionaries of messages. This Data Dictionary then can be used to decode and encode messages through out the rest of the program. The tool also has the capability to capture messages from the bus and save them on disk. While capturing messages, the user has the ability to monitor the bus in real time through scrolling trace displays, fixed position message displays, and bar graphs. After capturing the messages, the user can search for messages that conform to a complex set of criteria: by message name, source address, destination address, priority, message type and time stamp. These criteria may be logically OR'ed together to create a complex search. The tool can also highlight any messages found by the search, which may be combined with the next search in a more complex AND-OR search. Messages that were captured and saved to disk may be later transmitted onto the bus. In addition to these messages, the user may encode messages and add them to an existing file or a new file. In the message manipulation section, the Window's Clipboard is used to allow the user to quickly move messages from file to file.

Conclusion

J1939 allows for multiple vendors to create controllers, sensors, and actuators for the heavy duty truck and bus market that will communicate with other J1939 compliant devices. This assurance will allow more and more complex systems to be designed with little or no chance of components

not fitting in. J1939 continues where J1708, J1587 and J1922 left off. CAN offers a much higher bandwidth than standard UART based communications. Also CAN offers a much higher reliability and guaranteed latency times.