International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 10 (November 2014) www.ijirae.com

The House Intelligent Switch Control Network based On CAN bus

A.S.Jagadish
Department Electronics and Telecommunication Engineering,
Bharath University

Abstract*— *The Embedded Technology is now in its prime and the wealth of Knowledge available is mind-blowing. Embedded System is a combination of hardware and software. Embedded technology plays a major role in integrating the various functions associated with it. This needs to tie up the various sources of the Department in a closed loop system. This proposal greatly reduces the manpower, saves time and operates efficiently without human interference. Meanwhile, the continuing increase of electronic devices results in geometric growth of lead number and wiring becomes hard in limited space in automobile, restricts the expansion of internal control functions.

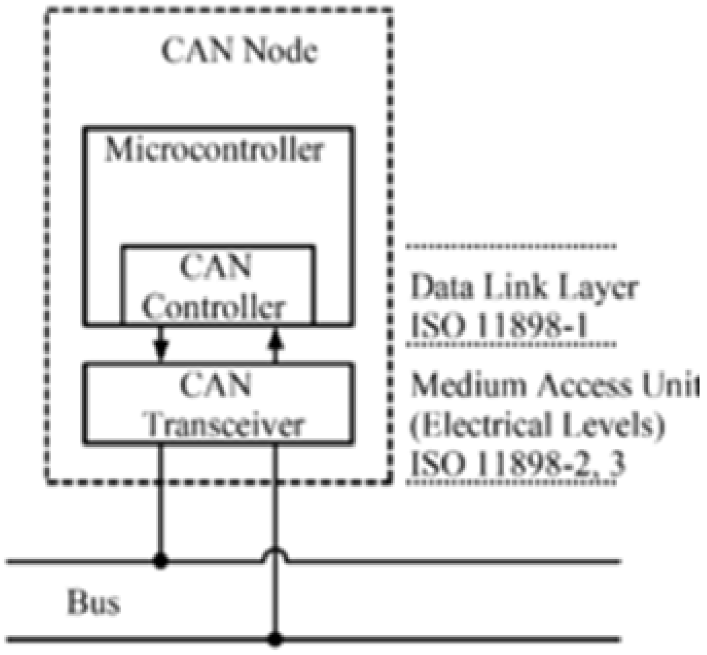
Keywords*—*controller area network(CAN) bus, passive infrared sensor (PIR sensor), switch Control , relay - types

I. Introduction

CAN bus (for controller area network) is a vehicle bus standard designed to allow microcontrollers and devices to communicate with each other within a vehicle without a host computer. CAN bus is a message-based protocol, designed specifically for automotive applications but now also used in other areas such as aerospace, maritime, industrial automation and medical equipment. The modern automobile may have as many as 70 electronic control units (ECU) for various subsystems.[6] Typically the biggest processor is the engine control unit. Others are used for transmission, airbags, antilock braking/ABS, cruise control, electric power steering, audio systems, power windows, doors, mirror adjustment, battery and recharging systems for hybrid/electric cars, etc. Some of these form independent subsystems, but communications among others are essential. A subsystem may need to control actuators or receive feedback from sensors. The CAN standard was devised to fill this need.

II. Architecture

CAN is a multi-master serial bus standard for connecting Electronic Control Units [ECUs] also known as nodes. Two or more nodes are required on the CAN network to communicate. The complexity of the node can range from a simple I/O device up to an embedded computer with a CAN interface and sophisticated software. The node may also be a gateway allowing a standard computer to communicate over a USB or Ethernet port to the devices on a CAN network.


CAN Node
Microcontroller
CAN Controller
CAN Transceiver
Data Link Layer ISO 11898-1
Medium Access Unit (Electrical Levels) ISO 11898-2, 3
Bus

A. CANbus Node

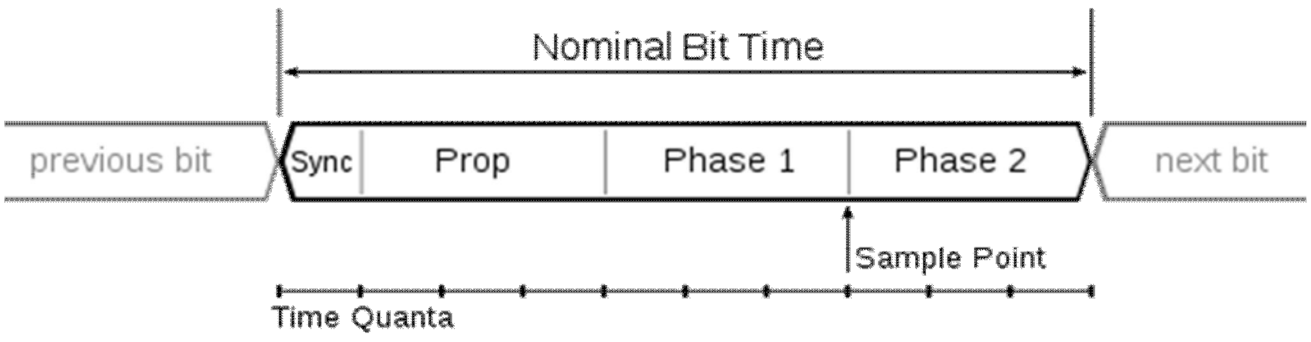
- Central processing unit, microprocessor, or host processor
    - The host processor decides what the received messages mean and what messages it wants to transmit.
    - Sensors, actuators and control devices can be connected to the host processor.
- CAN controller; often an integral part of the microcontroller
    - Receiving: the CAN controller stores the received serial bits from the bus until an entire message is available, which can then be fetched by the host processor (usually by the CAN controller triggering an interrupt).
    - Sending: the host processor sends the transmit message(s) to a CAN controller, which transmits the bits serially onto the bus when the bus is free.
- Transceiver Defined by ISO 11898-2/3 Medium Access Unit [MAU] standards
    - Receiving: it converts the data stream from CANbus levels to levels that the CAN controller uses. It usually has protective circuitry to protect the CAN controller.
    - Transmitting: it converts the data stream from the CAN controller to CANbus levels.

Each node is able to send and receive messages, but not simultaneously. A message or Frame consists primarily of the ID (identifier), which represents the priority of the message, and up to eight data bytes. A CRC, acknowledge slot [ACK] and other overhead are also part of the message. The improved CAN FD extends the length of the data section to up to 64 bytes per frame. The message is transmitted serially onto the bus using a non-return-to-zero (NRZ) format and may be received by all nodes.

The devices that are connected by a CAN network are typically sensors, actuators, and other control devices. These devices are connected to the bus through a host processor, a CAN controller, and a CAN transceiver.

B. Bit Timing

Each node in a CAN network has its own clock, and no clock is sent during data transmission. Synchronization is done by dividing each bit of the frame into a number of segments: synchronization, propagation, phase 1 and phase 2. The length of each phase segment can be adjusted based on network and node conditions. The sample point falls between phase buffer segment 1 and phase buffer segment 2, which helps facilitate continuous synchronization. Continuous synchronization in turn enables the receiver to be able to properly read the messages.



An example CAN bit timing with 10 time quanta per bit.

III. FRAMES

A CAN network can be configured to work with two different message (or "frame") formats: the standard or base frame format (described in CAN 2.0 A and CAN 2.0 B), and the extended frame format (only described by CAN 2.0 B). The only difference between the two formats is that the "CAN base frame" supports a length of 11 bits for the identifier, and the "CAN extended frame" supports a length of 29 bits for the identifier, made up of the 11-bit identifier ("base identifier") and an 18-bit extension ("identifier extension").

The distinction between CAN base frame format and CAN extended frame format is made by using the IDE bit, which is transmitted as dominant in case of an 11-bit frame, and transmitted as recessive in case of a 29-bit frame. CAN controllers that support extended frame format messages are also able to send and receive messages in CAN base frame format. All frames begin with a start-of-frame (SOF) bit that denotes the start of the frame transmission.

CAN has four frame types:

- Data frame: a frame containing node data for transmission
- Remote frame: a frame requesting the transmission of a specific identifier
- Error frame: a frame transmitted by any node detecting an error
- Overload frame: a frame to inject a delay between data and/or remote frame

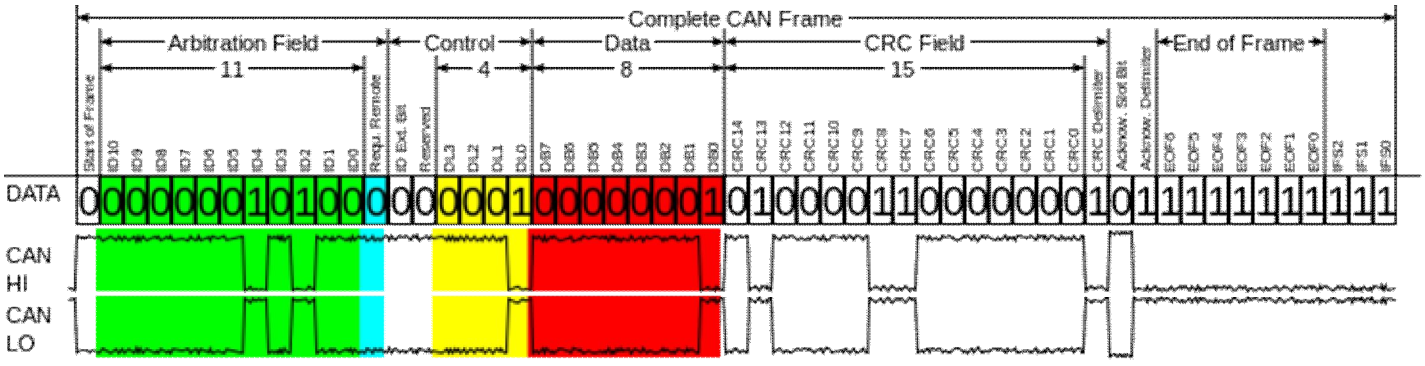
1) Data frame

The data frame is the only frame for actual data transmission. There are two message formats:

- Base frame format: with 11 identifier bits
- Extended frame format: with 29 identifier bits

The CAN standard requires the implementation must accept the base frame format and may accept the extended frame format, but must tolerate the extended frame format.

Base frame format



CAN-Frame in base format with electrical levels without stuffbits

The frame format is as follows:

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier for the data which also represents the message priority
Remote transmission request (RTR)	1	Dominant (0) (see Remote Frame below)
Identifier extension bit (IDE)	1	Declaring if 11 bit message ID or 29 bit message ID is used. Dominant (0) indicate 11 bit message ID while Recessive (1) indicate 29 bit message.
Reserved bit (r0)	1	Reserved bit (it must be set to dominant (0), but accepted as either dominant or recessive)
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check

CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

Extended frame format

The frame format is as follows:

Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier A	11	First part of the (unique) identifier for the data which also represents the message priority
Substitute remote request (SRR)	1	Must be recessive (1). Optional
Identifier extension bit (IDE)	1	Must be recessive (1). Optional
Identifier B	18	Second part of the (unique) identifier for the data which also represents the message priority
Remote transmission request (RTR)	1	Must be dominant (0)
Reserved bits (r0, r1)	2	Reserved bits (it must be set dominant (0), but accepted as either dominant or recessive)
Data length code (DLC)	4	Number of bytes of data (0–8 bytes)[a]
Data field	0–64 (0-8 bytes)	Data to be transmitted (length dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

It is physically possible for a value between 9–15 to be transmitted in the 4-bit DLC, although the data is still limited to eight bytes. Certain controllers allow the transmission and/or reception of a DLC greater than eight, but the actual data length is always limited to eight bytes.

The two identifier fields (A & B) combine to form a 29-bit identifier.

2) Remote frame

- Generally data transmission is performed on an autonomous basis with the data source node (e.g., a sensor) sending out a Data Frame. It is also possible, however, for a destination node to request the data from the source by sending a Remote Frame.
- There are two differences between a Data Frame and a Remote Frame. Firstly the RTR-bit is transmitted as a dominant bit in the Data Frame and secondly in the Remote Frame there is no Data Field.

RTR = 0 ; DOMINANT in data frame
RTR = 1 ; RECESSIVE in remote frame

In the very unlikely event of a Data Frame and a Remote Frame with the same identifier being transmitted at the same time, the Data Frame wins arbitration due to the dominant RTR bit following the identifier. In this way, the node that transmitted the Remote Frame receives the desired data immediately.

3) Error frame

The error frame consists of two different fields:

- The first field is given by the superposition of ERROR FLAGS (6–12 dominant/recessive bits) contributed from different stations.
- The following second field is the ERROR DELIMITER (8 recessive bits).

There are two types of error flags:

Active Error Flag
six dominant bits – Transmitted by a node detecting an error on the network that is in error state "error active".
Passive Error Flag
six recessive bits – Transmitted by a node detecting an active error frame on the network that is in error state "error passive".

4) Overload frame

The overload frame contains the two bit fields Overload Flag and Overload Delimiter. There are two kinds of overload conditions that can lead to the transmission of an overload flag:

1. The internal conditions of a receiver, which requires a delay of the next data frame or remote frame.
2. Detection of a dominant bit during intermission.

The start of an overload frame due to case 1 is only allowed to be started at the first bit time of an expected intermission, whereas overload frames due to case 2 start one bit after detecting the dominant bit. Overload Flag consists of six dominant bits. The overall form corresponds to that of the active error flag. The overload flag's form destroys the fixed form of the intermission field. As a consequence, all other stations also detect an overload condition and on their part start transmission of an overload flag. Overload Delimiter consists of eight recessive bits. The overload delimiter is of the same form as the error delimiter.

IV. Conclusions

In this paper, a network control system based on the CAN bus is presented. The cost, anti-jamming capability and stability are all taken into consideration in the house intelligent switch design. This project has been tested and verified on hardware and software. And the test shows it is reliable on controlling lights in a building with a wide prospect of market .

References

[1] High speed CAN (HSC) for vehicle Application at 500kbps, SAEJ2284,1999.
[2] Controller Area Network (CAN) Basics, AN713: Microchip Technology,Inc.
[3] Rick Stoneking,Anadigics, Inc, A Simple Can Node Using the MCP2519 and PIC 16F876; Microchip Inc. 2002.