Communication Protocols in Industry and Automotive Applications

M.Tech. Seminar

by

Rajesh Deshpande
04307908

under the guidance of

Prof. M. C. Chandorkar

Department of Electrical Engineering
Indian Institute of Technology, Bombay
Nov 2004

Contents

List of Figures

List Of Abbreviations

- PLC - Programmable Logic Controller
- ISO - International Standards Organization
- IPC - An association connecting Electronic Industries
- MAP - Manufacturing Automation Protocol
- OSI - Open Systems Interconnection
- IEC - International Electrotechnical Commission
- PID - Proportional Integral Derivative
- AI - Analog input
- AO - Analog Output
- EIA - Electronics Industries Association
- CRC - Cyclic redundancy check
- IP - Internet Protocol
- TCP - Transmission control Protocol
- CiA - CAN in Automation manufacturer's group
- SAE - Society of Automotive Engineers

Chapter 1

Introduction

The IT revolution in automation technology and embedded systems is opening up new savings potentials in the optimization of system processes. Communication in automation is becoming increasingly direct, horizontally at field level as well as vertically through all hierarchy levels. At field level the distributed peripherals, such as I/O modules, measuring transducers, drive units, valves and operator terminals communicate with the automation systems via an efficient, real-time communication system. At cell level, the programmable controllers such as PLCs and Computers communicate with each other using large information packets. Network infrastructures for industrial communications are complex. This seminar is an attempt to briefly introduce most commonly used protocols in industry applications and study CAN protocol in some depth. The first part gives an overview of the different communication systems used in automotive applications whereas second part explains CAN protocol in detail. Chapter 2 creates a background by explaining the Open Systems Interconnection model. Chapter 3 describes the use of fieldbus in process control. Chapters 4 and 5 give introduction to working of PROFIBUS and MODBUS in industrial networking. Chapter 6 explains breif idea of higher level protocols DeviceNet and CANOpen based on CAN. The last chapter presents working of CANbus in detail.

Chapter 2

Communication Protocols and their architecture

A communication protocol is a specification of a set of rules for accomplishing a particular type of communication. These rules may be like what medium to use, signal levels, analog or digital, serial or parallel, peer- to- peer or broadcast and likewise. These specifications also give rules on how to communicate with different nodes i.e. addressing mechanism, error correction, acknowledgment, bus traffic management, right up to establishing the communication from user's point of view. All this is accomplished by a layered architecture so that protocols in a layer perform specific functions. The Open Systems Interconnect *OSI* has defined a model of the layered architecture commonly known as OSI model. It is a 7 layered model(fig 2.1). Following is the brief description of each of these layers-

The physical layer defines electrical and physical properties of the medium as well as signal. Data link layer groups the raw bits into frames. The data link layer is responsible for error free transmission of the frames. The network layer handles addressing and routing of the packets over the network. Transport layer handles flow control and error handling. The session layer establishes and maintains sessions across the network. The presentation layer is responsible for syntax and semantics of data i.e. it handles protocol conversion, encryption, compression, character representation etc. The application layer is nothing but a user interface which allows access to the network services that support applications. Thus, any general communication system can be modeled into this layered architecture.

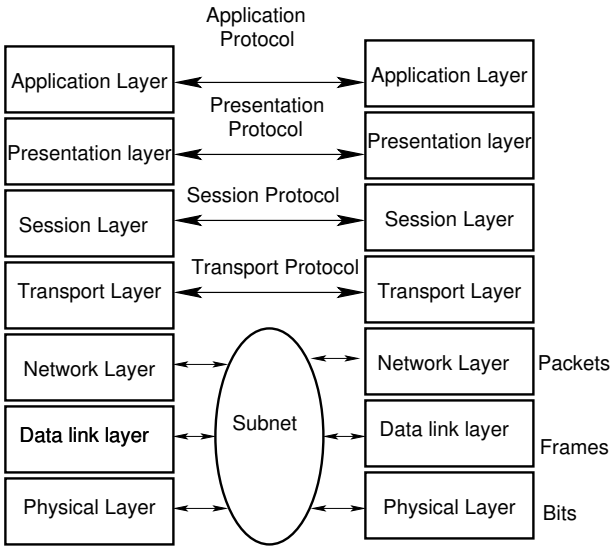

Figure 2.1: The ISO OSI model

Chapter 3

Field bus for Process Control

Originally developed as a digital replacement for 4-20mA analog current loop, the concept of fieldbus was extended by the introduction of smart field devices.

3.1 Fieldbus and OSI

Foundation Fieldbus targets distributed control in process automation. Foundation Fieldbus technology comprises the physical layer, the communication stack, and the user application. These components fit in the OSI communication model.

Foundation Fieldbus does not implement layers three through six of the OSI model, because process control does not require the services of these layers. An important part of Foundation Fieldbus is the defined user application, layer eight.(fig 3.1)

3.2 Physical layer

Foundation Fieldbus uses the ISA S50.02-1992 and IEC 1158-2 physical-layer. These standards specify data-communication rates of 31.25 kbps, 1 Mbps, and 2.5 Mbps. Devices operating at 31.25 kbps can draw their power directly from the network. Low speed devices can use twisted pair to carry both power and signals.

3.3 Communication Stack

Communication stack provides an interface of user application to the physical layer. The data link layer manages access to the fieldbus through a centralized bus scheduler, a link active scheduler LAS. Fieldbus offers cyclic scheduled communications to execute control

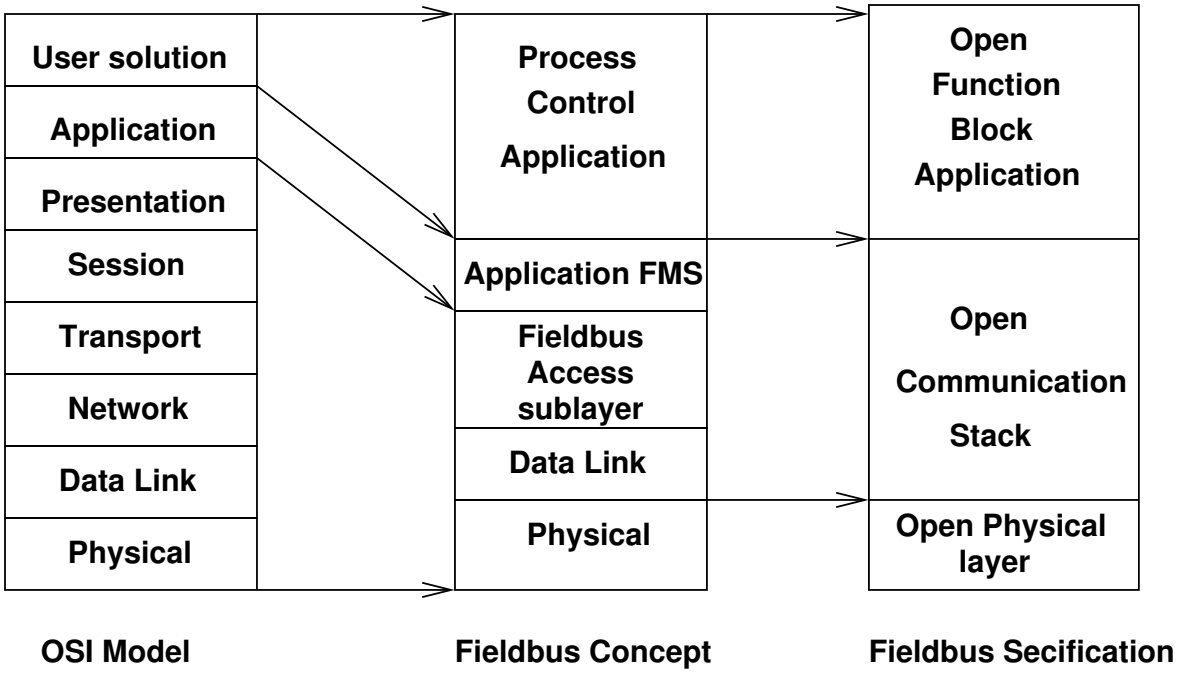

Figure 3.1: Fieldbus and OSI model

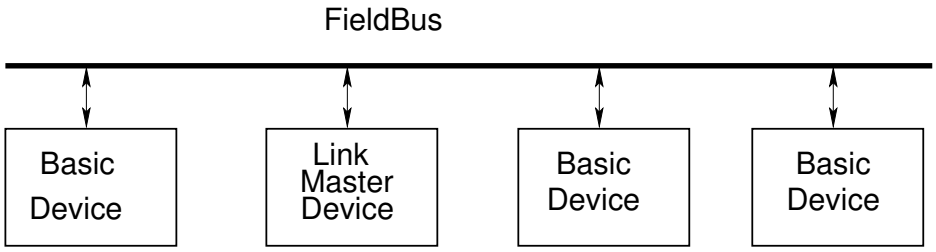

Figure 3.2: Fieldbus Elements

loops and also acyclic communications to handle events, such as alarm conditions. LAS has a list of the alloted transmission times for each device for cyclic communications. Acyclic communication is achieved by token passing.

Thus LAS is in charge of access to the bus. A link master is a device that contains LAS. Thus any fieldbus contains one or more link master and many basic devices.(fig 3.2)

Communication stack is again divided in two sub-layers. The stack's Fieldbus-access sublayer FAS provides an interface between the data-link layer and layer seven. The FAS provides communication services, such as client/server, publisher/subscriber, and event distribution. The stack's Fieldbus-messaging-specification FMS layer defines a model for applications to interact over the fieldbus. The object dictionary and the virtual field

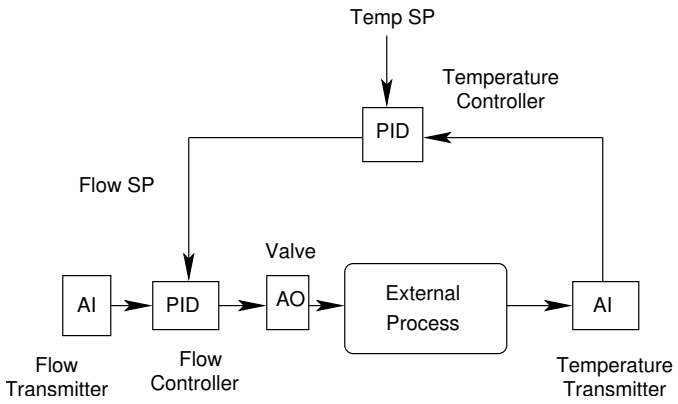

Figure 3.3: Example of Fieldbus Functional blocks

device are important features of this model. The object dictionary is a fieldbus-device structure that describes data that can appear on the fieldbus. You can think of the object dictionary as a look-up table that supplies information, such as the data type of values that you read from or write to a device. The virtual field device is a model for remotely viewing data described in the object dictionary.

3.4 Fieldbus User layer

Foundation fieldbus defines an eighth layer-user layer. Foundation Fieldbus user interacts with devices through a set of blocks that define device capabilities in a standardized way. Resource blocks, function blocks, and transducer blocks are the general types of blocks. Resource blocks describe the characteristics of a device, such as name, manufacturer, and serial number. Function blocks provide the control and I/O behavior of a device. Transducer blocks decouple function blocks from the functions that read and write local inputs and outputs.

Function blocks are the core components with which a user specifies a control system's behavior. Foundation Fieldbus defines standard sets of function blocks. Connecting the inputs and outputs of individual function blocks specifies communication of data on the bus. Even more important, you can precisely schedule a function block's execution, allowing direct execution of control loops over the network. The function blocks reside in individual devices, but the network schedules the overall execution of the function. An example of a simple control loop has analog-input (AI), proportional/integral/derivative (PID), and analog-output (AO) function blocks. (fig 3.3)

Chapter 4

Profibus-DP-A high speed I/O network

Profibus comprises of compatible protocol-stack variations–Profibus-FMS, Profibus-DP, and Profibus-PA. Profibus-FMS handles high-level, non-real-time communications among devices. Profibus-DP targets time-critical communication between controllers and distributed peripherals. German national standard DIN 19245 and European Fieldbus Standard EN50170 both specify Profibus-FMS and DP. Profibus-PA targets process-control applications, especially those requiring intrinsically safe operation.

Profibus uses twisted pair transmission medium and industry standard RS485 in manufacturing applications or IEC 1158-2 in process control. Profibus can also use Ethernet or TCP/IP as shown in figure. Profibus-DP is the profibus running on RS485. Profibus PA is profibus superimposed on standard IEC 1158-2. IEC 1158-2 allows data communication and power transmission over the network using two wire technology used in explosive environment.

Profibus-DP provides high-speed data transfer at the sensor and actuator levels. Controllers, such as PLCs, exchange data with their distributed peripherals using a fast serial link. Profibus defines master and slave devices. Master devices, or active stations, can control the bus and transfer messages without a remote request. Slave devices are simple peripherals, such as sensors and actuators. Slaves, or passive stations, cannot access the bus except at the request of a master. In a Profibus-DP system, data exchange is mainly cyclic, with a master reading input information from slaves and sending output information back to the slaves.

4.1 Profibus and OSI model

Profibus-DP does not use OSI layers three to six. The direct-data-link mapper maps the data-link-layer functions for the user interface. The user interface specifies the system

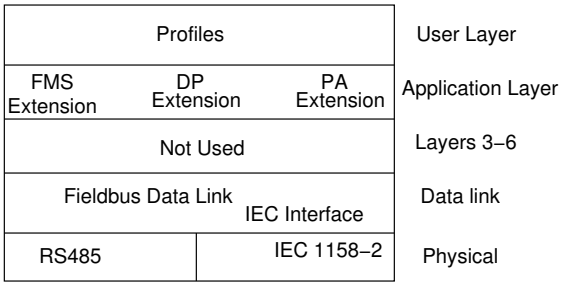

Figure 4.1: Profibus layers

and device behaviour of Profibus-DP devices. (fig 4.1)

4.2 Physical Layer

The Profibus standard defines two physical layers with appropriate medium-access protocols for different transmission techniques. The base version of the physical layer uses copper wire in accordance with US standard EIA RS-485. This physical medium uses a two-conductor twisted-pair cable. A second physical medium is fiber-optic, which greatly extends the bus length at high transmission speeds. Fiber-optic versions of Profibus operate at transmission speeds as high as 12 Mbps.

4.3 Data link Layer

The Profibus data-link layer is designated as the fieldbus data link. The MAC defines when a station can transmit data and ensures that only one station has the right to transmit data at any time.

The Profibus medium-access protocol includes the token-passing method for communication between complex stations (masters) and the master-slave method for communication between complex stations and simple peripherals (slaves). This combined method is called hybrid medium access.

Profibus-DP is a low-level network that targets high-performance I/O scanning. A Profibus-DP network can include the DP-Master Class 1 (DPM1), which is the central controller. A DPM1 exchanges information with the decentralized stations (DP-slaves) in a defined message cycle. Typical devices are PLCs, computer numerical controllers, and robot controllers. The network can also include DP-Master Class 2 (DPM2) devices, which are used for programming, configuration, or diagnostics, and DP-slaves, which are I/O devices that provide input information and issue output information to the system.

Typical DP-slaves are discrete inputs or outputs for 24V dc or 230V ac, analog inputs, and analog outputs. The amount of input and output data is device-dependent and has a maximum of 246 bytes.

A typical Profibus-DP system configuration comprises one or more DP-slaves assigned to a DPM1. Interaction between the DPM1 and a DP-slave has parameterization, configuration, and data-transfer phases. In the parameterization and configuration phases, the DPM1 sends configuration data to the DP-slave.

The Profibus specification requires the vendors of every DP-slave and DPM1 to document the device's characteristics in a device data sheet and database file that follow a standard structure, content, and coding format. The standardized approach permits common configuration tools to configure any Profibus-DP device.

Chapter 5

MODBUS

MODBUS Protocol is a messaging structure, widely used to establish master-slave communication between intelligent devices. Modbus devices communicate over a serial network in a master/slave (request/response) type relationship using one of two transmission modes: ASCII (American Standard Code for Information Interchange) mode or RTU (Remote Terminal Unit) mode. A MODBUS message sent from a master to a slave contains the address of the slave, the "command" (e.g. "read register" or "write register"), the data, and a check sum (LRC or CRC). MODBUS is traditionally implemented using RS232, RS422, or RS485 over a variety of media (e.g. fiber, radio, cellular, etc). MODBUS TCP/IP uses TCP/IP and Ethernet to carry the MODBUS messaging structure.

The MODBUS protocol comes in 2 flavours: ASCII transmission mode and RTU transmission mode. In ASCII mode, eight-bit bytes of information are sent as two ASCII characters. The primary advantage of ASCII mode is the flexibility of the timing sequence. Up to a one second interval can occur between character transmissions without causing communication errors. ASCII mode uses only ASCII character for data coding and can be used with any dummy modem like communication interface, even ones with with 7 bit communication channel. In RTU mode, data is sent as two four-bit, hexadecimal characters, providing for higher throughput than in ASCII mode for the same baud rate.

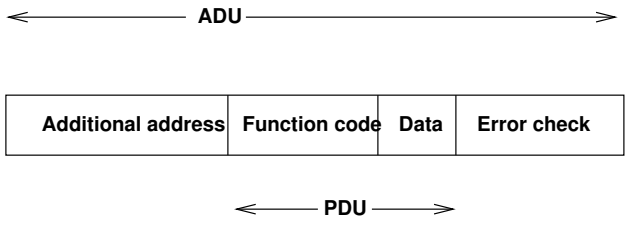

Figure 5.1: MODBUS Messaging Frame

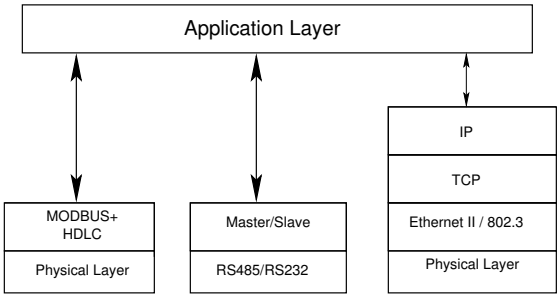

Figure 5.2: Modbus Protocol Architecture

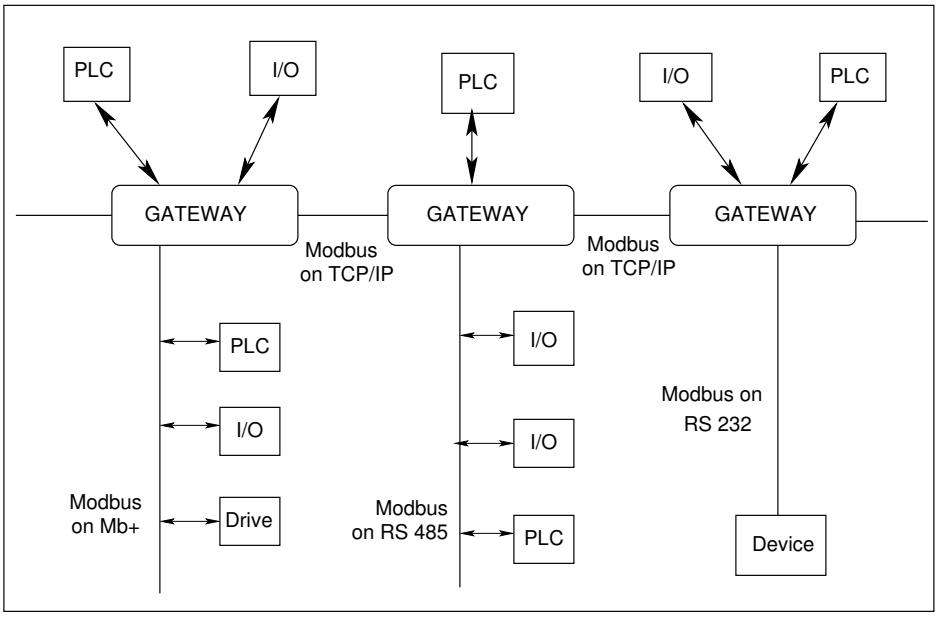

Figure 5.3: Modbus as an Interface

Modbus RTU is a binary protocol and more time delay critical than the ASCII protocol. Both ASCII and RTU work nicely with direct wire connection and with 2/4-wire short haul modems. For long distance connections where there are lots of communication devices in the communication route, the ASCII protocol is preferred, because it is less sensitive to delays and can be transported also over 7-bit communication channels.

Thus MODBUS is essentially an application layer protocol defined in layer 7 of the OSI model. (fig 5.2)

MODBUS allows easy communication within all types of network architectures. (fig 5.3)

Modbus defines basic architecture of the protocol Data unit (PDU) independent of the underlying layer. Some additional fields for addresses depending on network structure used

and PDU make an Application Data Unit. (fig 5.1) In a message from client to server, a function code specifies what action to perform. The data field may contain additional information that server uses to carry out the action performed eg. register addresses, quantity of items to be handled etc.

Chapter 6

Devicenet and CANOpen

DeviceNet and CANOpen both are CAN based manufacturing buses used as higher level protocols on the basic stack defined in CAN- an abbreviation for Controller Area Network.

6.1 DeviceNet

Developed by Allen-Bradley, DeviceNet is now the responsibility of an independent supplier organization, the Open DeviceNet Vendors Association. ODVA controls the DeviceNet specification. DeviceNet is a low-level network that connects industrial devices, such as sensors and actuators, to higher level devices, such as controllers. DeviceNet focuses especially on the interchangeability of low-cost, simple devices often used in manufacturing applications. Examples are limit switches, photoelectric sensors, motor starters, bar-code readers, variable-frequency drives, and operator interfaces. Part of the goal of DeviceNet is to achieve the same level of interchangeability for 120/220V-ac and 24V-dc discrete devices using digital communications, as is possible with hard-wired I/O devices.

The basis of DeviceNet is CAN. The CAN offers fast response and good reliability under adverse environmental and electrical conditions. Apart from these attributes of CAN, cost was also major factor for choosing CAN. The availability of CAN chips helps achieving improved performance at decreased prices. Part 2 of this document explains the working of CAN in detail.

6.1.1 DeviceNet and OSI model

Devicenet adds two more layers media and application layer to the basic stack of physical and data link layer defined by CAN. Hence,in short, CAN defines the form of data movement, whereas the DeviceNet application layer defines the data's meaning.

A DeviceNet network can have as many as 64 node addresses. Because it uses the CAN data-link layer, DeviceNet is inherently a peer-to-peer network, though many applications use a master/slave architecture.

In DeviceNet, two entities on the network must establish a connection before communication can occur. Explicit-message and I/O are the basic types of connections. The network assigns transmissions associated with a connection, an identification value, or a connection ID. An explicit message connection provides a generic, multipurpose communication path between two devices and provides the means for performing typical request/response functions, such as device configuration. The explicit messaging protocol indicates how a device should interpret a message. An I/O connection is a dedicated, special-purpose communication path between a producing device and one or more consuming devices. The connection ID implies the content of the associated I/O message. No defined protocol exists for I/O-message data. The devices at either end of the connection must know the general form of I/O messages. I/O connections carry time-critical, control-oriented data.

You can choose the I/O mechanism that yields the most efficient data transfer. The predefined master/slave connection set defines the communication mechanism for a basic network comprising a master (for example, a PLC) and a set of simple devices (for example, on/off switches or motor starters).

6.1.2 Devicenet objects and device profiles

The object model provides a template for organizing and implementing the attributes (data), services (methods or procedures), and behaviours of a DeviceNet product's components. For each attribute, the model provides an addressing scheme that comprises the node address or medium-access-control identifier (MAC ID), the object-class identifier, the instance number, and the attribute number. There are four different classes of objects- an identity object, connection object, parameter object and application object. An identity object's attributes include vendor ID, device type, product code, revision status, serial number, product name, and state. A connection object represents one end of a virtual connection (explicit or I/O) between two nodes on a DeviceNet network. Configuration options that are attributes of the parameter object include parameter values, ranges, text strings, and limits. Also, a device usually includes at least one application object other than those from the assembly or parameter class.

To facilitate compatibility and interoperability, DeviceNet defines standard device profiles. As One need not implement all the objects and all the attributes within an

object, a device profile contains the definition of the device's object model. The device profile and electronic data sheet describe the objects in a device and thus the device's function as a user sees it.

6.2 CANOpen

As the existing standards based on CAN allow designer to use off-the-shelf components and still have freedom to optimize the performance to meet specific requirements. In all the applications where customization is required CANOpen is the better option.

CANOpen was developed by ASPIC Esprit project. In 1994, the CANOpen specification was handled to CiA-CAN in Automation users and manufacturers group. The protocol is now standardized and it is EN 50325-4.

CANOpen standardizes the way in which data is structured and exchanged in CAN environment. Device profiles of CANOpen specify the data sets and communication models supported by various modules such as I/O devices, drivers, encoders etc. It offers standardized communication objects for real time data transfer and special objects like synchronization messages, emergency messages and management data like Error control, NMT message.

The core of any CANOpen node is the object dictionary, a lookup table with 16 bit index and 8 bit subindex. All process and communication related information is stored at predefined entries in the object dictionary. From any node object dictionary data of any node can be accessed by sending read/write requests. Each node has an unique ID which gets embedded into these requests which allows peer to peer communication. Thus messages are categorized into requests and replies referred to as Receive service data objects and transmit service data objects (RSDO and TSDO). Separate identifiers are maintained at each node in the object dictionary for these two.

Thus CANOpen tries to overcome limitations imposed by CAN by allowing data more than 8 bits, peer to peer communication, variable length data etc. Availability of various tools from many manufacturers for development, testing, configuration and maintenance makes this standard practically implemented in various industry applications.

Chapter 7

Controller Area Network: A closer Look

CAN an (abbreviation for Controller Area Network) protocol is an ISO standard for serial data communication originally developed by Bosch Technology. CAN is getting widespread use in industrial automation as well as automotive mobile machines.

7.1 Features of CAN

- Digital serial transmission
- Multi-master communication
- Privatization of messages
- Simple transmission medium
- Hardware implementation of the protocol
- Excellent error handling
- Fine fault confinement

7.2 Protocol Basic Characteristics

1. It uses non destructive bit wise arbitration to control access to the bus.
2. The messages are small and protected by checksum.

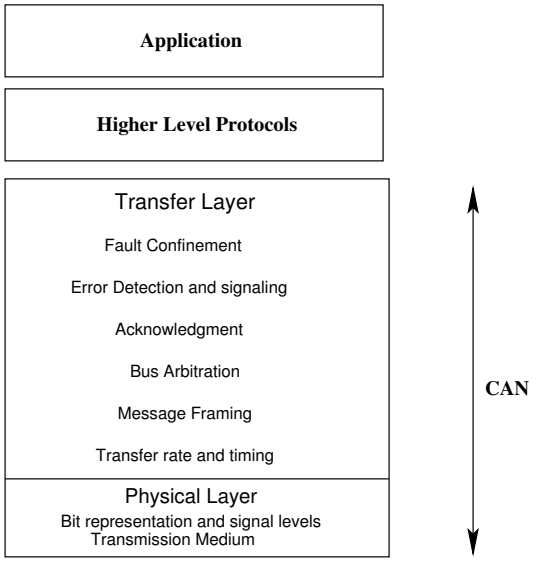

Figure 7.1: CAN and OSI model

3. There is no explicit address in the messages, instead each message carries a numeric value which controls its priority on the bus.

4. Multi-cast communication: It is a broadcast bus, no of nodes can receive the message simultaneously and can act upon it.

5. System Flexibility: Number of nodes can be added or removed from the bus without any change in hardware/software of any node or in the application layer.

6. An elaborate error handling scheme that results in retransmitted messages when they are not properly received.

7. There are effective means for isolating faults and removing faulty nodes from the bus.

7.3 CAN and the OSI model

The CAN standard specifies the physical and data-link layer (also referred to as transfer layer).

Physical layer defines electrical specifications of the transmission and the channel i.e. it defines how data is physically transmitted over the medium.

Transfer layer is actually the kernel of the protocol. It defines a few different message types, their formats, arbitration rules for bus access, acknowledgment etc. It is also

responsible for bit timing and synchronization, error detection and fault confinement. (fig 7.1)

7.4 CAN Physical Layer

7.4.1 Signal Levels

The CAN bus uses Non-Return To Zero (NRZ) coding with bit stuffing. There are two different signaling states: dominant (logic 0) and recessive (logic 1). These correspond to certain electrical levels depending on physical layer used. The modules are connected together in wired AND fashion. So, if just one node drives the bus into dominant mode the whole bus goes to dominant state regardless of several other nodes transmitting recessive state.

7.4.2 Physical medium

There are various types of physical media in use.

1. The most common specified by ISO CAN standard 11898. It uses differential transmission on twisted pair wire. It is sometimes known as *High Speed CAN*
2. Another ISO standard, std 11519, defines another two wire balanced signaling scheme for lower bus speeds. It is fault tolerant, so the signaling can continue even it the bus wire is cut or shorted to ground. It is sometimes called *High Speed CAN*
3. SAE J2411 defines a single-wire physical layer. It is used cheaply in cars and other locomotives.

7.4.3 Maximum Bus Speed

The maximum speed of the CAN bus, according to standard, is 1Mbit/s. Low speed CAN bus (referred ISO 11519 as above) can go up to 125 kbit/s. Single wire CAN bus can go up to around 50 to 100 kbit/s.

7.4.4 Minimum Bus Speed

Minimum bus speed is not specified in the standard, but some transceivers specify minimum bus speed also eg. using TJA1050 data rate below 50kbit/s.

7.4.5 Maximum Cable Length

At a speed of 1 Mbit/s, maximum cable length of 130 ft (about 40 meters) can be used. The length is limited as the signal wavefront should travel to most remote node and back before the next bit sample. Other maximum cable lengths are 100 meters at 500kbit/s, 6 kilometers at 10kbit/s.

7.4.6 Bus Termination

An ISO 11898 CAN bus must be terminated. This is done by a resistor of 120 ohms in each end of the bus. Termination serves two purposes-

- Remove the signal reflections at the end of the bus.
- Ensure the bus gets correct dc levels.

7.4.7 The Cable

The ISO 11898 prescribes that cable impedance should be 120 ohms, but an impedance in the interval (108-132) ohms is permitted. The standard is defined for twisted pair cable, shielded or unshielded.

7.4.8 CAN connectors

There is no standard for CAN connectors, but each higher level protocol defines one or few preferred connector types:

- 9 pin DSUB proposed by CiA
- 5 pin Mini-C and/or Micro-C used by DeviceNet.

If power is supplied, it shall be in the range +7 to +13V, 100mA.

7.5 CAN Data link Layer

It specifies-

- Message framing
- Bus Arbitration
- Acknowledgment

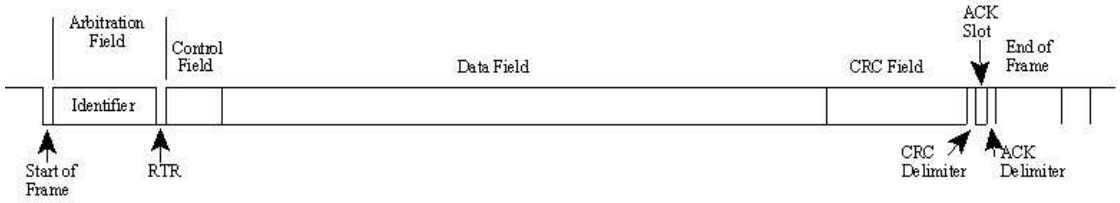

Figure 7.2: CAN Data Frame

- Error Detection and Signaling
- Fault Confinement
- Transfer rate and Timing.

7.5.1 Message Framing

The CAN Messages The CAN is a broadcast type of bus i.e. all nodes can hear all transmissions and there is no way to send a message to a specific node. The higher level protocols however provide local filtering so that a node responds only to the interesting messages. There is no explicit address in the messages and the messages are said to be content addressed, that is, their contents implicitly determine their addresses.

There are four types of messages-

- The Data Frame
- The Remote Frame
- The Error Frame
- The Overload Frame
- Data Frame

  Data frame carries information from transmitter to receiver. It contains seven different fields. (fig 7.2)

  - Start Of Frame: It marks the beginning of data frames and remote frames. It consists of single dominant bit. A station is allowed to transmit only when the bus is idle. All the stations synchronize to the leading edge caused by start of frame.

    The arbitration field consists of identifier and the RTR bit.

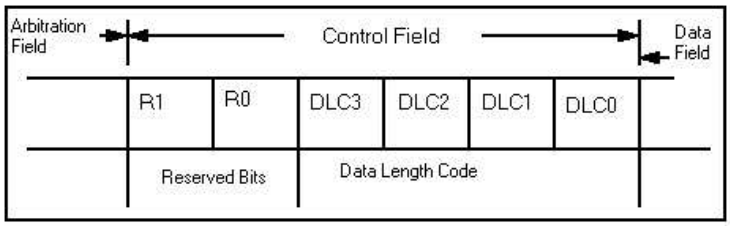

Figure 7.3: Control Field

- IDENTIFIER: The identifier's length is 11 bits (Std CAN). These bits are transmitted in the order from ID-10 to ID-0. The seven most significant bits (ID10 to ID4) must not be all recessive.
- RTR bit: Remote transmission request bit. It distinguishes a data frame and a remote frame.
- Control Field: It consists of 6 bits. It includes the data length code (4 bits) and two bits reserved for future expansion. The data length code indicates the no of bytes in data field. (fig 7.3)
- Data Field: The data field consists of the data to be transferred within a data frame. It can extend from 0 to 8 bits. MSB is transmitted first.
- CRC Field: It contains CRC sequence and CRC delimiter. This is a frame check sequence based on cyclic redundancy check. The CRC delimiter is single recessive bit at the end of CRC sequence.
- ACK Slot: The transmitter gives a recessive bit in this slot. The receiver may respond by superscribing dominant bit if it has received the message correctly.
- ACK Delimiter: ACK delimiter is a recessive bit after ACK slot.
- End Of Frame: Each Data frame and remote frame ends with a sequence of 8 recessive bits that marks the end of frame.

- Remote Frame

  It is a request to produce data frame of labeled X where X is a value of the arbitration field in both the frames. Remote frame has two main differences as compared to data frame-

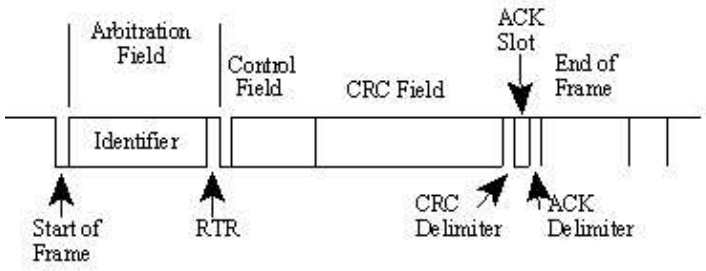

Figure 7.4: The Remote Frame

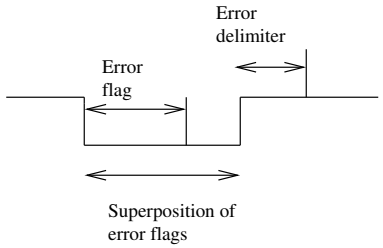

Figure 7.5: The Error Frame

1. Remote frame has RTR bit in the arbitration field recessive.
2. There is no data field in the message.

The purpose of the remote frame is to solicit the transmission of the corresponding frame. If say node A sends remote frame on the bus with X arbitration field, the node B may respond to it by sending a data frame also containing arbitration field X. (fig 7.4)

- The Error Frame

  The error frame consists of two fields-error flag and error delimiter. The error flag is the superposition of error flags given by all nodes. There are two forms of error flag-active error flag and passive error flag.
  Active error flag contains six consecutive dominant bits. Passive error flag contains six consecutive recessive bits unless it is overwritten by dominant bits by any other node. An error delimiter has 8 recessive bits. It provides some space in which other nodes can send their error flags when they detect the first error flag. (fig 7.5)The mechanism of error detection is explained later in section 7.5.5.

- The Overload Frame

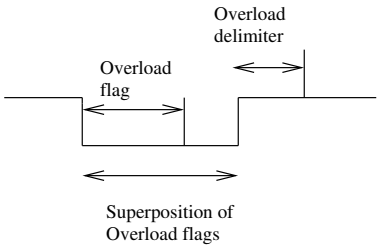

Figure 7.6: The Overload Frame

Overload frame is transmitted when receiver requires a delay of the next data frame or remote frame. It contains overflow flag and overflow delimiter.

- Overload Flag

  The overload flag contains six dominant bits. The overflow flag form destroys the fixed form of intermission. As a consequence all other stations also detect an overload condition and start transmitting an overload flag. (fig 7.6)

- Overload delimiter

  It has the same form as error delimiter. After transmission of overflow flag station monitors the bus until it detects a transition from dominant to recessive bit, at that point all the nodes have stopped transmitting overload flags and all the nodes get synchronized by sending 6 recessive bits of the overload delimiter.

7.5.2 Bus arbitration and Message Priority

The bus arbitration is the process in which two or more CAN nodes agree on who is to use the bus.

Any CAN controller may start transmission when it has detected an idle bus. This may result in two or more controllers starting a message at the same time. This is resolved in the following manner. The transmitting nodes monitor the bus while they are sending. If a node detects a dominant level when it is sending recessive level itself, it will immediately quit the arbitration process and become the receiver instead of sender. The arbitration is performed for the whole of the arbitration field and when that entire field has been sent, exactly one transmitter remains on the bus. No time is lost in the arbitration process.

Since the bus is wired and a dominant bit is logically 0, it follows that the node having numerically lower arbitration field will have the higher priority.

7.5.3 Message Addressing and Identification

It is worth noting that there is no explicit address in the CAN message. The contents of a message are identified by an identifier which is located somewhere in the message. Hence the messages are said to be contents-addressed. A conventional message address will be used like "Here is message for node X" and a contents addressed message will be "Here is a message containing data labeled X". The difference between the two is less apparent but significant. The standard does not say that the arbitration field must be used as a message identifier.

7.5.4 Message Coding

The frame segments START OF FRAME, ARBITRATION FIELD, DATA FIELD and CRC SEQUENCE are coded by method of bit stuffing. Whenever transmitter detects five consecutive identical bits in a message to be transmitted it automatically inserts a complementary bit in the message. The remaining fields and other frames such as overload frame and error frame are of fixed format. The individual bits are coded by NRZ (Non-Return-to-Zero) type of coding.

7.5.5 Error Handling

Error handling is built into CAN protocol and is of great importance for the performance of the CAN bus. The error handling aims at detection of errors in message appearing on the CAN bus so that transmitter can retransmit the message.

- Types of Errors

  There are five different types of errors defined- bit error, stuff error, CRC error, form error and acknowledgment error.

- The Error flag

  Every node will try to detect the errors in the transmitted message. If an error is found the discovering node will transmit an error flag destroying the bus traffic. The other nodes will detect the error flag and take appropriate action. For an error active node it is an active error flag and for a passive node it is a passive error flag. Active and passive nodes are described in next section.

7.5.6 Error Detection Mechanisms

There are five different mechanisms of detecting an error. First two of these work at bit level and others work at message level.

- Bit Monitoring

  Each transmitter on the bus monitors the transmitted signal level. If the bit level is different from that transmitted bit error is signaled. No bit error is raised during arbitration process.

- Bit Stuffing

  If more than five consecutive bits of the same level occur on the bus for fields on which stuffing is done, a stuff error is signaled.

- Frame Check

  Some fields of message have fixed format (eg. CRC delimiter, Ack Delimiter, End of frame etc). If a CAN controller detects invalid value in this field form error is signaled.

- Acknowledgment Check

  All the nodes which correctly receive the message are expected to send a dominant level in the ack slot. If a transmitter cannot detect dominant level in the ack slot, ack error is signaled.

- Cyclic Redundancy Check

  Each message contains a 15 bit CRC field. Any node that detects a different CRC field than what it has calculated, it will generate a CRC error.

7.5.7 Fault Confinement

Fault confinement is the management entity in the media access sublayer of the CAN protocol which distinguishes short disturbances from permanent failures. With respect to fault confinement the node may be in any of the three modes-

- Error passive
- Error active
- Bus off

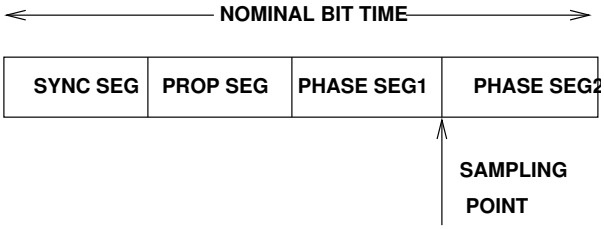

Figure 7.7: Bit Duration

An error active unit can normally take part in bus communication and sends active error flag when error is detected. A passive node sends a passive error flag. Also after an error, error passive unit will wait before initiating further transmission. A node which is in bus off state won't transmit anything on the bus at all.

For fault confinement two counts are implemented in every bus unit.

- Transmit error count
- Receive error count

There are several rules governing how these counters are incremented or decremented. In essence, a transmitter detecting fault increments its transmit error counter faster than the listening nodes will increment their receive error counters. Any of the nodes starts in error active mode. When any of the error counters raises above 127, the node will enter a state known as error passive and when the transmit error counter raises above 255, the node will enter the Bus Off State.

7.6 Bit Timing and Synchronization

Nominal bit rate is the no of bits per second transmitted by a transmitter in the absence of resynchronization. Nominal bit time is the reciprocal of nominal bit rate. The Nominal bit time can be divided into following segments- (fig 7.7)

- Synchronization segment $Sync - seg$
- Propagation time segment $prop - seg$
- Phase buffer segment1 $Phase - seg1$
- Phase buffer segment2 $Phase - seg2$

- Synchronization Segment

  This part of the bit time is used to synchronize the various nodes on the bus. An edge is expected to lie within this segment.

- Propagation Segment

  This part of the bit time is used to compensate for the physical delay times within the network.

- Phase-seg1,Phase-seg2

  These phase buffer segments are used to compensate for edge phase errors.

- Sample Point

  The Sample Point is the point of time at which the bit is sampled and read as the value of respective bit.

7.7 Higher Level Protocols

CAN protocol just specifies how small packets of data can be transmitted from node A to node B using shared communication medium. It contains no information about flow control, transportation of data larger than 8 bits, node addresses, establishment of communication. These topics are covered in higher level protocols which belong to higher layers of OSI model. Most commonly used higher level protocols based on CAN are CANOpen and Devicenet as explained in chapter 6.

7.8 Some of CAN controller chips

- Infineon 81C90 DPRAM type controller which supports CAN 2.0A. It has 16 message buffers and a global buffer and two 8 bit parallel ports.

- Intel 82527 This chip is successor of first CAN chip 82526 in the market. It supports CAN2.0 and provides DPRAM type interface with up to 15 message objects, two 8 bit parallel ports.

- Microchip 2515 Full CAN 2.0B, 1 Mbps, 3 Tx buffers, 2 Rx buffers.

- Philips SJA1000 This is advanced version of 82C200 supports CAN2.0 active and a receive FIFO of 64 bytes, complete access to error counters and error diagnostics.

Chapter 8

Conclusion

The best of buses for one industrial-automation application can be the worst of buses for another. Wide-ranging requirements have spurred the creation of specialized networks. Understanding the forces driving each of these networks can help making a better choice for the next system.

Fieldbus integrates digital control making it the best choice for process control. Foundation fieldbus enables the integration of a manufacturer's plant and global enterprise through the high Speed Ethernet (HSE) backbone.

Several different methods are now being used for communication between industrial devices and the world around them such as TCP/IP, CAN etc. The simplest communication interfaces for industrial equipment are serial interfaces such as RS-232, RS-422 and RS-485, used for communication among controllers, simple sensors, servos, valves and actuators. Some of the protocols that are used today on top of RS-232, RS-422 and RS-485 are Profibus and Modbus and Interbus. Profibus PA allows communication in hazardous industrial environment. However, MODBUS TCP/IP and Ethernet give easier interface on the internet. In addition, because of networking topology they require a great deal of effort for maintenance and network upgrades.

The CAN protocol is a serial bus system that comes to industrial design from automotive applications. Therefore, its cables and connectors are already ruggedized, and the equipment is relatively inexpensive and widely available. CAN's disadvantages are a small data payload up to 8 bytes per transmission and low-speed data transfers, at a maximum of 1 Mbit/second. It is a simple protocol well-suited for command-and-control-type interfaces. Its sophisticated error detection mechanisms and retransmission of faulty messages guarantee data integrity. In industrial design, it is best employed for communication among systems, sensors, devices and actuators, including printed-circuit boards.

A number of higher-level protocols based on CAN have been developed for industrial

control and automation, including CANOpen and Devicenet. CANOpen was designed for motion-oriented machine control networks such as handling systems. It is found in many applications, including medical equipment, maritime electronics, public transportation and building automation. DeviceNet is an open, low-level network that provides connections between simple industrial devices (such as sensors and actuators) and such higher-level devices as programmable logic controllers and computers. This network employs the Common Industrial Protocol to provide control, configuration and data collection capabilities for industrial devices.

With the help of communication protocols that are mostly new to the industrial environment, industrial control and automation equipment is being integrated with back-office and front-office systems, resulting in greater efficiency and productivity, as well as lower operating costs. Work still needs to be done, however, on reducing redundancy among competing industry standards.

References

1. "Foundation Fieldbus Primer" by fieldbus inc., revision 1.1, June 24, 2004, http://www.fieldbus.org/Primer 1_1.pdf

2. "Modbus Application Protocol Specification version 1.1, MODBUS.ORG", http://www.modbus.org/ModbusApplicationProtocol_v1_1.pdf

3. http://www.can-cia.de/

4. "CAN Specification version 2.0" by Bosch, http://www.can.bosch.com/can2spec.pdf

5. "A tale of three buses", Mike Santori, National Instruments, EDN Access October 23, 1997, http://www.reed-electronics.com/ednmag/archives/1997/102397/