Generation of clock signals on the I²C-bus is always the responsibility of master devices; each master generates its own clock signals when transferring data on the bus. Bus clock signals from a master can only be altered when they are stretched by a slow-slave device holding-down the clock line, or by another master when arbitration occurs.

5 GENERAL CHARACTERISTICS

Both SDA and SCL are bi-directional lines, connected to a positive supply voltage via a current-source or pull-up resistor (see Fig.3). When the bus is free, both lines are HIGH. The output stages of devices connected to the bus must have an open-drain or open-collector to perform the wired-AND function. Data on the I²C-bus can be transferred at rates of up to 100 kbit/s in the Standard-mode, up to 400 kbit/s in the Fast-mode, or up to 3.4 Mbit/s in the High-speed mode. The number of interfaces connected to the bus is solely dependent on the bus capacitance limit of 400 pF. For information on High-speed mode master devices, see Section 13.

6 BIT TRANSFER

Due to the variety of different technology devices (CMOS, NMOS, bipolar) which can be connected to the I²C-bus, the levels of the logical ‘0’ (LOW) and ‘1’ (HIGH) are not fixed and depend on the associated level of V_{DD} (see Section 15 for electrical specifications). One clock pulse is generated for each data bit transferred.

6.1 Data validity

The data on the SDA line must be stable during the HIGH period of the clock. The HIGH or LOW state of the data line can only change when the clock signal on the SCL line is LOW (see Fig.4).

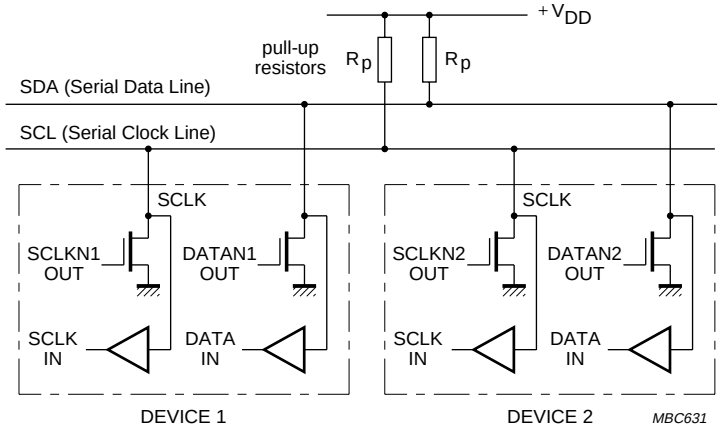


Fig.3 Connection of Standard- and Fast-mode devices to the I²C-bus.

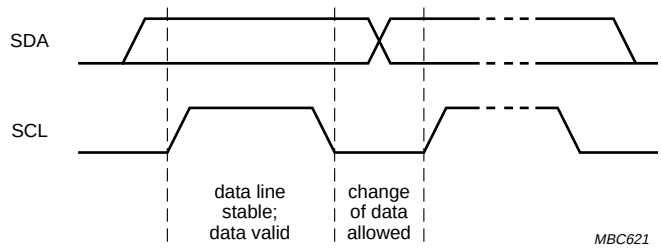


Fig.4 Bit transfer on the I²C-bus.

6.2 START and STOP conditions

Within the procedure of the I²C-bus, unique situations arise which are defined as START (S) and STOP (P) conditions (see Fig.5).

A HIGH to LOW transition on the SDA line while SCL is HIGH is one such unique case. This situation indicates a START condition.

A LOW to HIGH transition on the SDA line while SCL is HIGH defines a STOP condition.

START and STOP conditions are always generated by the master. The bus is considered to be busy after the START condition. The bus is considered to be free again a certain time after the STOP condition. This bus free situation is specified in Section 15.

The bus stays busy if a repeated START (Sr) is generated instead of a STOP condition. In this respect, the START (S) and repeated START (Sr) conditions are functionally identical (see Fig.6). For the remainder of this document, therefore, the S symbol will be used as a generic term to represent both the START and repeated START conditions, unless Sr is particularly relevant.

Detection of START and STOP conditions by devices connected to the bus is easy if they incorporate the necessary interfacing hardware. However, microcontrollers with no such interface have to sample the SDA line at least twice per clock period to sense the transition.

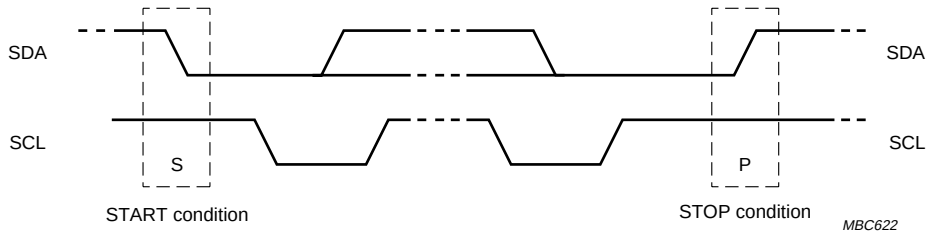


Fig.5 START and STOP conditions.

7 TRANSFERRING DATA

7.1 Byte format

Every byte put on the SDA line must be 8-bits long. The number of bytes that can be transmitted per transfer is unrestricted. Each byte has to be followed by an acknowledge bit. Data is transferred with the most significant bit (MSB) first (see Fig.6). If a slave can't receive or transmit another complete byte of data until it has performed some other function, for example servicing an internal interrupt, it can hold the clock line SCL LOW to force the master into a wait state. Data transfer then continues when the slave is ready for another byte of data and releases clock line SCL.

In some cases, it's permitted to use a different format from the I²C-bus format (for CBUS compatible devices for example). A message which starts with such an address can be terminated by generation of a STOP condition, even during the transmission of a byte. In this case, no acknowledge is generated (see Section 10.1.3).

7.2 Acknowledge

Data transfer with acknowledge is obligatory. The acknowledge-related clock pulse is generated by the master. The transmitter releases the SDA line (HIGH) during the acknowledge clock pulse.

The receiver must pull down the SDA line during the acknowledge clock pulse so that it remains stable LOW during the HIGH period of this clock pulse (see Fig.7). Of course, set-up and hold times (specified in Section 15) must also be taken into account.

Usually, a receiver which has been addressed is obliged to generate an acknowledge after each byte has been received, except when the message starts with a CBUS address (see Section 10.1.3).

When a slave doesn't acknowledge the slave address (for example, it's unable to receive or transmit because it's performing some real-time function), the data line must be left HIGH by the slave. The master can then generate either a STOP condition to abort the transfer, or a repeated START condition to start a new transfer.

If a slave-receiver does acknowledge the slave address but, some time later in the transfer cannot receive any more data bytes, the master must again abort the transfer. This is indicated by the slave generating the not-acknowledge on the first byte to follow. The slave leaves the data line HIGH and the master generates a STOP or a repeated START condition.

If a master-receiver is involved in a transfer, it must signal the end of data to the slave- transmitter by not generating an acknowledge on the last byte that was clocked out of the slave. The slave-transmitter must release the data line to allow the master to generate a STOP or repeated START condition.

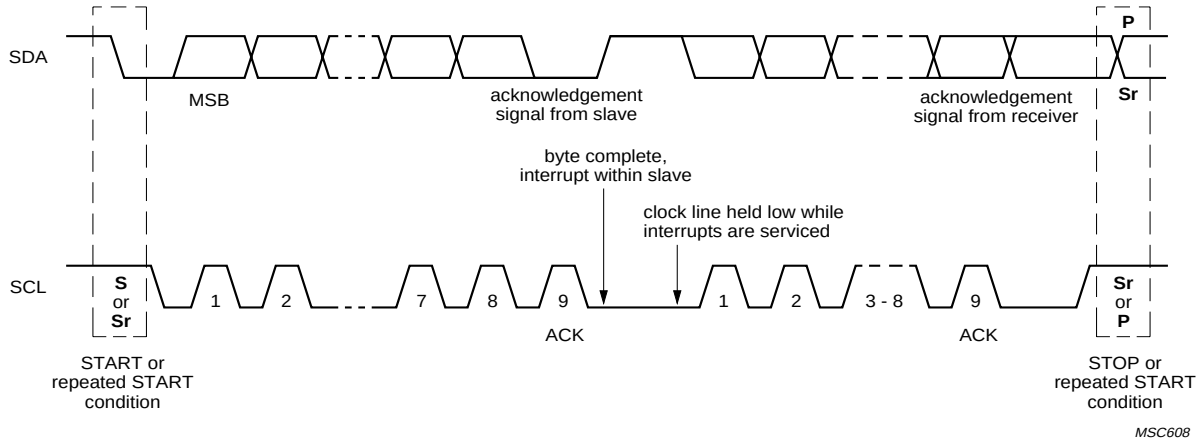


Fig.6 Data transfer on the I²C-bus.

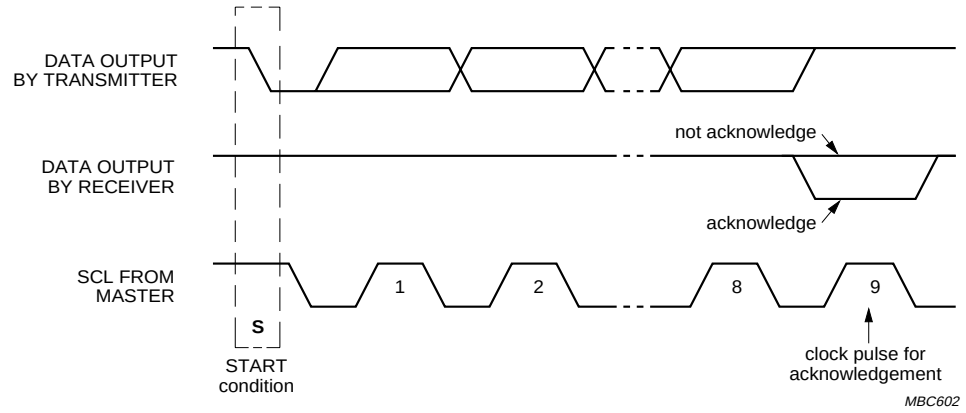


Fig.7 Acknowledge on the I²C-bus.

8 ARBITRATION AND CLOCK GENERATION

8.1 Synchronization

All masters generate their own clock on the SCL line to transfer messages on the I²C-bus. Data is only valid during the HIGH period of the clock. A defined clock is therefore needed for the bit-by-bit arbitration procedure to take place.

Clock synchronization is performed using the wired-AND connection of I²C interfaces to the SCL line. This means that a HIGH to LOW transition on the SCL line will cause the devices concerned to start counting off their LOW period and, once a device clock has gone LOW, it will hold the SCL line in that state until the clock HIGH state is reached (see Fig.8). However, the LOW to HIGH transition of this clock may not change the state of the SCL line if another clock is still within its LOW period. The SCL line will therefore be held LOW by the device with the longest LOW period. Devices with shorter LOW periods enter a HIGH wait-state during this time.

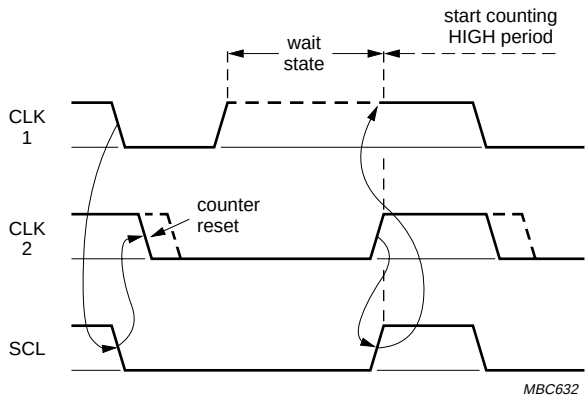


Fig.8 Clock synchronization during the arbitration procedure.

When all devices concerned have counted off their LOW period, the clock line will be released and go HIGH. There will then be no difference between the device clocks and the state of the SCL line, and all the devices will start counting their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronized SCL clock is generated with its LOW period determined by the device with the longest clock LOW period, and its HIGH period determined by the one with the shortest clock HIGH period.

8.2 Arbitration

A master may start a transfer only if the bus is free. Two or more masters may generate a START condition within the minimum hold time ($t_{HD;STA}$) of the START condition which results in a defined START condition to the bus.

Arbitration takes place on the SDA line, while the SCL line is at the HIGH level, in such a way that the master which transmits a HIGH level, while another master is transmitting a LOW level will switch off its DATA output stage because the level on the bus doesn't correspond to its own level.

Arbitration can continue for many bits. Its first stage is comparison of the address bits (addressing information is given in Sections 10 and 14). If the masters are each trying to address the same device, arbitration continues with comparison of the data-bits if they are master-transmitter, or acknowledge-bits if they are master-receiver. Because address and data information on the I²C-bus is determined by the winning master, no information is lost during the arbitration process.

A master that loses the arbitration can generate clock pulses until the end of the byte in which it loses the arbitration.

As an Hs-mode master has a unique 8-bit master code, it will always finish the arbitration during the first byte (see Section 13).

If a master also incorporates a slave function and it loses arbitration during the addressing stage, it's possible that the winning master is trying to address it. The losing master must therefore switch over immediately to its slave mode.

Figure 9 shows the arbitration procedure for two masters. Of course, more may be involved (depending on how many masters are connected to the bus). The moment there is a difference between the internal data level of the master generating DATA 1 and the actual level on the SDA line, its data output is switched off, which means that a HIGH output level is then connected to the bus. This will not affect the data transfer initiated by the winning master.

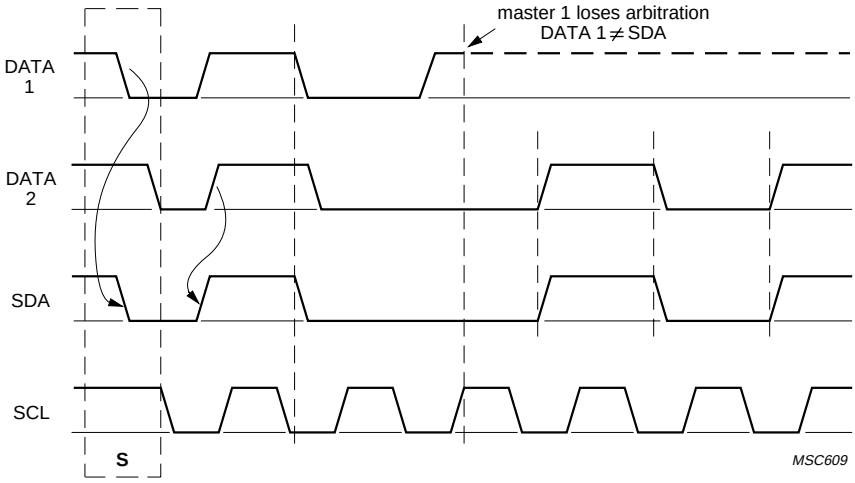


Fig.9 Arbitration procedure of two masters.

Since control of the I²C-bus is decided solely on the address or master code and data sent by competing masters, there is no central master, nor any order of priority on the bus.

Special attention must be paid if, during a serial transfer, the arbitration procedure is still in progress at the moment when a repeated START condition or a STOP condition is transmitted to the I²C-bus. If it's possible for such a situation to occur, the masters involved must send this repeated START condition or STOP condition at the same position in the format frame. In other words, arbitration isn't allowed between:

- A repeated START condition and a data bit
- A STOP condition and a data bit
- A repeated START condition and a STOP condition.

Slaves are not involved in the arbitration procedure.

8.3 Use of the clock synchronizing mechanism as a handshake

In addition to being used during the arbitration procedure, the clock synchronization mechanism can be used to enable receivers to cope with fast data transfers, on either a byte level or a bit level.

On the byte level, a device may be able to receive bytes of data at a fast rate, but needs more time to store a received byte or prepare another byte to be transmitted. Slaves can then hold the SCL line LOW after reception and acknowledgment of a byte to force the master into a wait state until the slave is ready for the next byte transfer in a type of handshake procedure (see Fig.6).

On the bit level, a device such as a microcontroller with or without limited hardware for the I²C-bus, can slow down the bus clock by extending each clock LOW period. The speed of any master is thereby adapted to the internal operating rate of this device.

In Hs-mode, this handshake feature can only be used on byte level (see Section 13).

9 FORMATS WITH 7-BIT ADDRESSES

Data transfers follow the format shown in Fig.10. After the START condition (S), a slave address is sent. This address is 7 bits long followed by an eighth bit which is a data direction bit (R/$\overline{W}$) - a 'zero' indicates a transmission (WRITE), a 'one' indicates a request for data (READ). A data transfer is always terminated by a STOP condition (P) generated by the master. However, if a master still wishes to communicate on the bus, it can generate a repeated START condition (Sr) and address another slave without first generating a STOP condition. Various combinations of read/write formats are then possible within such a transfer.

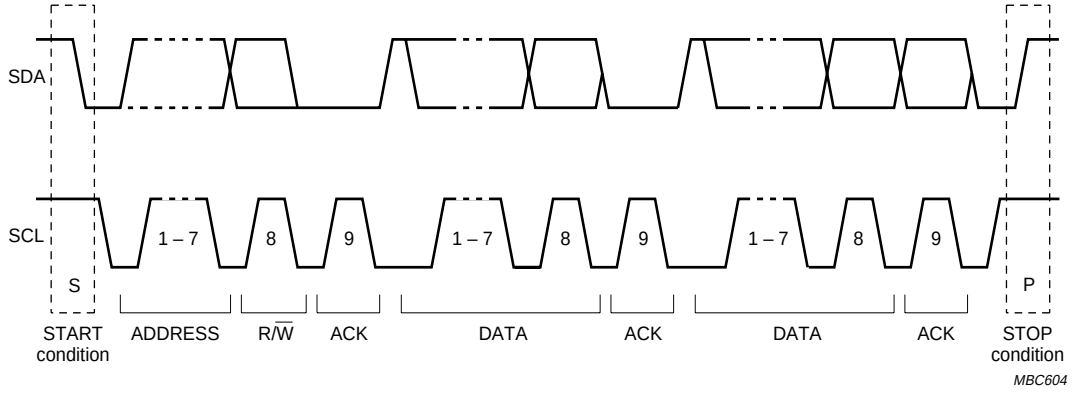


Fig.10 A complete data transfer.

Possible data transfer formats are:

- Master-transmitter transmits to slave-receiver. The transfer direction is not changed (see Fig.11).
- Master reads slave immediately after first byte (see Fig.12). At the moment of the first acknowledge, the master- transmitter becomes a master- receiver and the slave-receiver becomes a slave-transmitter. This first acknowledge is still generated by the slave. The STOP condition is generated by the master, which has previously sent a not-acknowledge ($\overline{A}$).
- Combined format (see Fig.13). During a change of direction within a transfer, the START condition and the slave address are both repeated, but with the R/$\overline{W}$ bit reversed. If a master receiver sends a repeated START condition, it has previously sent a not-acknowledge ($\overline{A}$).

NOTES:

1. Combined formats can be used, for example, to control a serial memory. During the first data byte, the internal memory location has to be written. After the START condition and slave address is repeated, data can be transferred.
2. All decisions on auto-increment or decrement of previously accessed memory locations etc. are taken by the designer of the device.
3. Each byte is followed by an acknowledgment bit as indicated by the A or $\overline{A}$ blocks in the sequence.
4. I²C-bus compatible devices must reset their bus logic on receipt of a START or repeated START condition such that they all anticipate the sending of a slave address, even if these START conditions are not positioned according to the proper format.
5. A START condition immediately followed by a STOP condition (void message) is an illegal format.

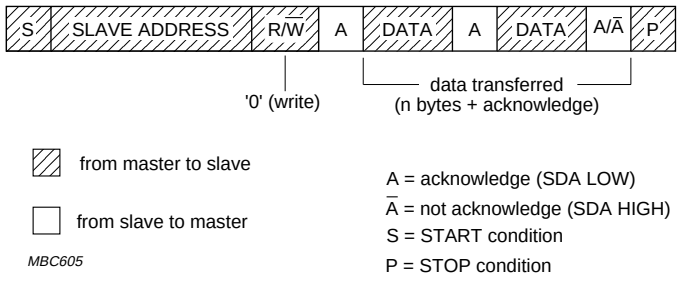


Fig.11 A master-transmitter addressing a slave receiver with a 7-bit address. The transfer direction is not changed.

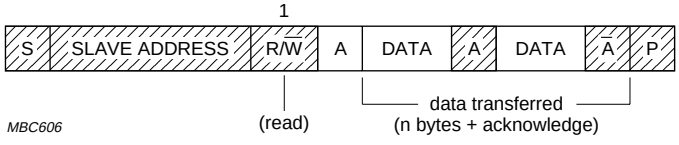


Fig.12 A master reads a slave immediately after the first byte.

1 INTRODUCTION

The Controller Area Network (CAN) is a serial communications protocol which efficiently supports distributed realtime control with a very high level of security.
Its domain of application ranges from high speed networks to low cost multiplex wiring. In automotive electronics, engine control units, sensors, anti-skid-systems, etc. are connected using CAN with bitrates up to 1 Mbit/s. At the same time it is cost effective to build into vehicle body electronics, e.g. lamp clusters, electric windows etc. to replace the wiring harness otherwise required.
The intention of this specification is to achieve compatibility between any two CAN implementations. Compatibility, however, has different aspects regarding e.g. electrical features and the interpretation of data to be transferred. To achieve design transparency and implementation flexibility CAN has been subdivided into different layers.

- the (CAN-) object layer
- the (CAN-) transfer layer
- the physical layer

The object layer and the transfer layer comprise all services and functions of the data link layer defined by the ISO/OSI model. The scope of the object layer includes

- finding which messages are to be transmitted
- deciding which messages received by the transfer layer are actually to be used,
- providing an interface to the application layer related hardware.

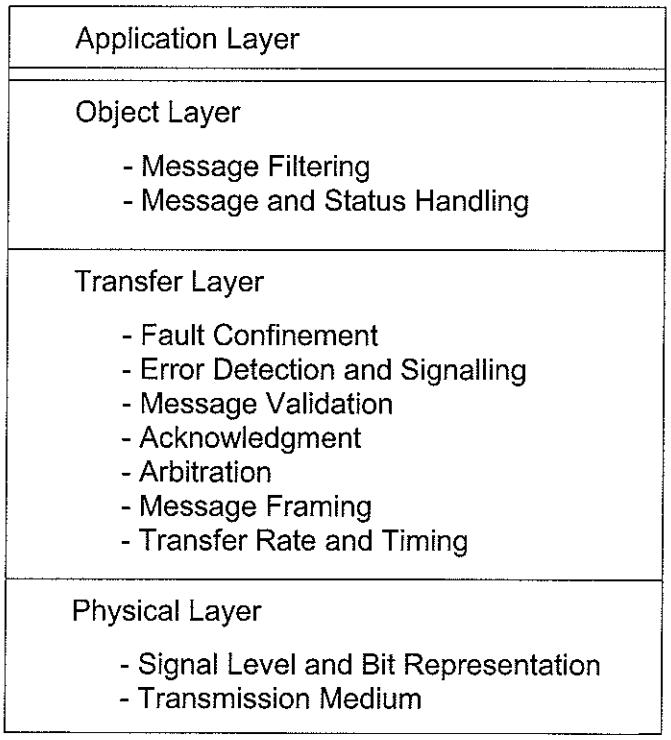
There is much freedom in defining object handling. The scope of the transfer layer mainly is the transfer protocol, i.e. controlling the framing, performing arbitration, error checking, error signalling and fault confinement. Within the transfer layer it is decided whether the bus is free for starting a new transmission or whether a reception is just starting. Also some general features of the bit timing are regarded as part of the transfer layer. It is in the nature of the transfer layer that there is no freedom for modifications.
The scope of the physical layer is the actual transfer of the bits between the different nodes with respect to all electrical properties. Within one network the physical layer, of course, has to be the same for all nodes. There may be, however, much freedom in selecting a physical layer.
The scope of this specification is to define the transfer layer and the consequences of the CAN protocol on the surrounding layers.

2 BASIC CONCEPTS

CAN has the following properties

- prioritization of messages
- guarantee of latency times
- configuration flexibility
- multicast reception with time synchronization
- system wide data consistency
- multimaster
- error detection and signalling
- automatic retransmission of corrupted messages as soon as the bus is idle again
- distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes

Layered Structure of a CAN Node



- The Physical Layer defines how signals are actually transmitted. Within this specification the physical layer is not defined so as to allow transmission medium and signal level implementations to be optimized for their application.
- The Transfer Layer represents the kernel of the CAN protocol. It presents messages received to the object layer and accepts messages to be transmitted from the object layer. The transfer layer is responsible for bit timing and synchronization, message framing, arbitration, acknowledgment, error detection and signalling, and fault confinement.
- The Object Layer is concerned with message filtering as well as status and message handling.

The scope of this specification is to define the transfer layer and the consequences of the CAN protocol on the surrounding layers.

Messages
Information on the bus is sent in fixed format messages of different but limited length (see section 3: Message Transfer). When the bus is free any connected unit may start to transmit a new message.

Information Routing
In CAN systems a CAN node does not make use of any information about the system configuration (e.g. station addresses). This has several important consequences.

System Flexibility: Nodes can be added to the CAN network without requiring any change in the software or hardware of any node and application layer.

Message Routing: The content of a message is named by an IDENTIFIER. The IDENTIFIER does not indicate the destination of the message, but describes the meaning of the data, so that all nodes in the network are able to decide by MESSAGE FILTERING whether the data is to be acted upon by them or not.

Multicast: As a consequence of the concept of MESSAGE FILTERING any number of nodes can receive and simultaneously act upon the same message.

Data Consistency: Within a CAN network it is guaranteed that a message is simultaneously accepted either by all nodes or by no node. Thus data consistency of a system is achieved by the concepts of multicast and by error handling.

Bit rate
The speed of CAN may be different in different systems. However, in a given system the bitrate is uniform and fixed.

Priorities
The IDENTIFIER defines a static message priority during bus access.

Remote Data Request
By sending a REMOTE FRAME a node requiring data may request another node to send the corresponding DATA FRAME. The DATA FRAME and the corresponding REMOTE FRAME are named by the same IDENTIFIER.

Multimaster
When the bus is free any unit may start to transmit a message. The unit with the message of higher priority to be transmitted gains bus access.

Arbitration
Whenever the bus is free, any unit may start to transmit a message. If 2 or more units start transmitting messages at the same time, the bus access conflict is resolved by bitwise arbitration using the IDENTIFIER. The mechanism of arbitration guarantees that neither information nor time is lost. If a DATA FRAME and a REMOTE FRAME with the same IDENTIFIER are initiated at the same time, the DATA FRAME prevails over the REMOTE FRAME. During arbitration every transmitter compares the level of the bit transmitted with the level that is monitored on the bus. If these levels are equal the unit may continue to send. When a 'recessive' level is sent and a 'dominant' level is monitored (see Bus Values), the unit has lost arbitration and must withdraw without sending one more bit.

Safety
In order to achieve the utmost safety of data transfer, powerful measures for error detection, signalling and self-checking are implemented in every CAN node.

Error Detection
For detecting errors the following measures have been taken:
- Monitoring (transmitters compare the bit levels to be transmitted with the bit levels detected on the bus)
- Cyclic Redundancy Check
- Bit Stuffing
- Message Frame Check

Performance of Error Detection
The error detection mechanisms have the following properties:

- all global errors are detected.
- all local errors at transmitters are detected.
- up to 5 randomly distributed errors in a message are detected.
- burst errors of length less than 15 in a message are detected.
- errors of any odd number in a message are detected.

Total residual error probability for undetected corrupted messages: less than

$$\text{message error rate} * 4.7 * 10^{-11}.$$

Error Signalling and Recovery Time
Corrupted messages are flagged by any node detecting an error. Such messages are aborted and will be retransmitted automatically. The recovery time from detecting an error until the start of the next message is at most 29 bit times, if there is no further error.

Fault Confinement
CAN nodes are able to distinguish short disturbances from permanent failures. Defective nodes are switched off.

Connections
The CAN serial communication link is a bus to which a number of units may be connected. This number has no theoretical limit. Practically the total number of units will be limited by delay times and/or electrical loads on the bus line.

Single Channel
The bus consists of a single channel that carries bits. From this data resynchronization information can be derived. The way in which this channel is implemented is not fixed in this specification. E.g. single wire (plus ground), two differential wires, optical fibres, etc.

Bus values
The bus can have one of two complementary logical values: 'dominant' or 'recessive'. During simultaneous transmission of 'dominant' and 'recessive' bits, the resulting bus value will be 'dominant'. For example, in case of a wired-AND implementation of the bus, the 'dominant' level would be represented by a logical '0' and the 'recessive' level by a logical '1'. Physical states (e.g. electrical voltage, light) that represent the logical levels are not given in this specification.

Acknowledgment

All receivers check the consistency of the message being received and will acknowledge a consistent message and flag an inconsistent message.

Sleep Mode / Wake-up

To reduce the system's power consumption, a CAN-device may be set into sleep mode without any internal activity and with disconnected bus drivers. The sleep mode is finished with a wake-up by any bus activity or by internal conditions of the system. On wake-up, the internal activity is restarted, although the transfer layer will be waiting for the system's oscillator to stabilize and it will then wait until it has synchronized itself to the bus activity (by checking for eleven consecutive 'recessive' bits), before the bus drivers are set to "on-bus" again.

In order to wake up other nodes of the system, which are in sleep-mode, a special wake-up message with the dedicated, lowest possible IDENTIFIER (rrr rrrd rrrr; r = 'recessive' d = 'dominant') may be used.

3 MESSAGE TRANSFER

3.1 Frame Types

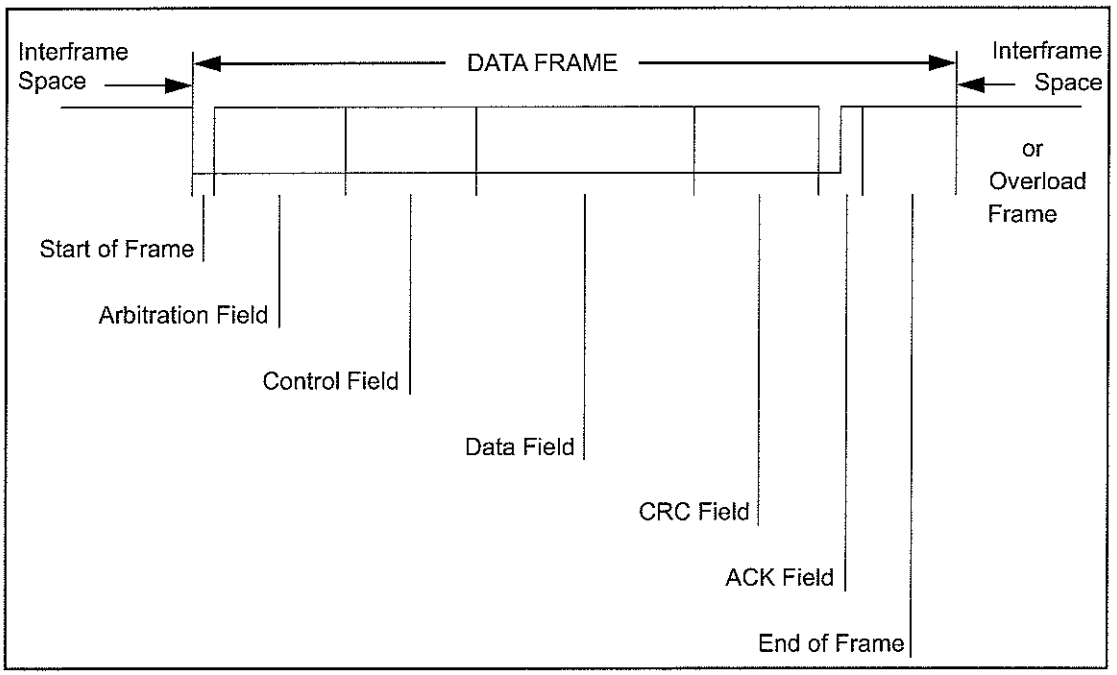
Message transfer is manifested and controlled by four different frame types:

A DATA FRAME carries data from a transmitter to the receivers.
A REMOTE FRAME is transmitted by a bus unit to request the transmission of the DATA FRAME with the same IDENTIFIER.
An ERROR FRAME is transmitted by any unit on detecting a bus error.
An OVERLOAD FRAME is used to provide for an extra delay between the preceding and the succeeding DATA or REMOTE FRAMEs.

DATA FRAMEs and REMOTE FRAMEs are separated from preceding frames by an INTERFRAME SPACE.

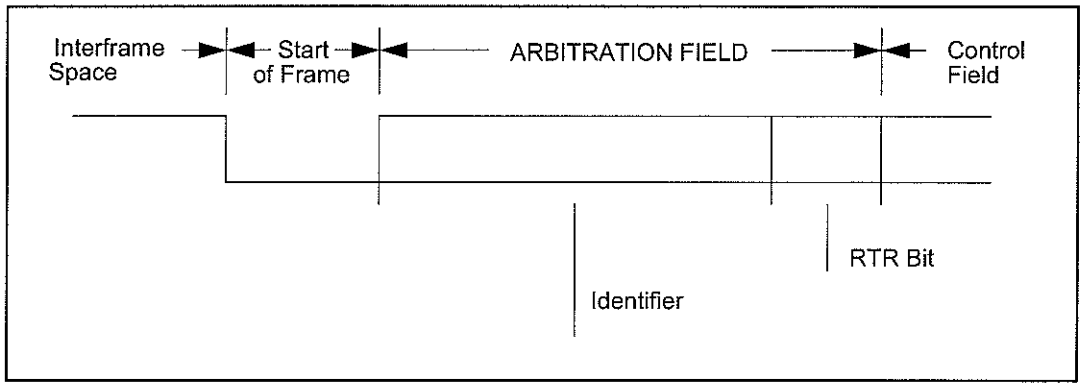
3.1.1 DATA FRAME

A DATA FRAME is composed of seven different bit fields:
START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD, CRC FIELD, ACK FIELD, END OF FRAME. The DATA FIELD can be of length zero.



START OF FRAME
marks the beginning of DATA FRAMES and REMOTE FRAMEs. It consists of a single 'dominant' bit.
A station is only allowed to start transmission when the bus is idle (see BUS IDLE). All stations have to synchronize to the leading edge caused by START OF FRAME (see 'HARD SYNCHRONIZATION') of the station starting transmission first.

ARBITRATION FIELD
The ARBITRATION FIELD consists of the IDENTIFIER and the RTR-BIT.

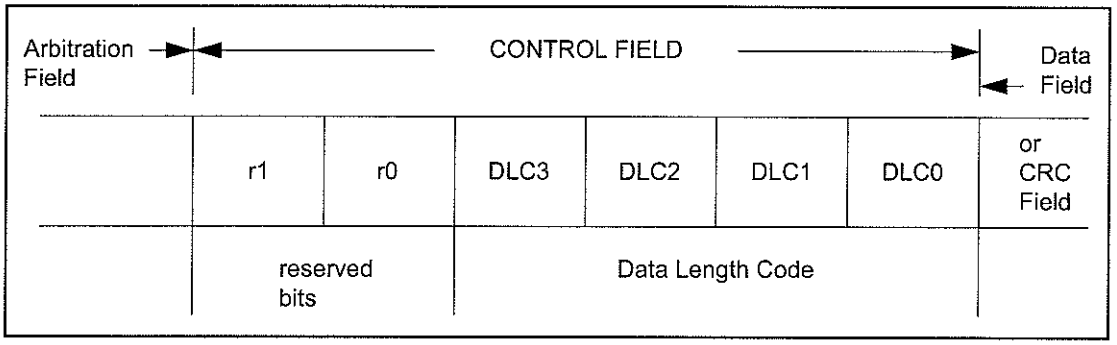


IDENTIFIER
The IDENTIFIER's length is 11 bits. These bits are transmitted in the order from ID-10 to ID-0. The least significant bit is ID-0. The 7 most significant bits (ID-10 - ID-4) must not be all 'recessive'.

RTR BIT
Remote Transmission Request BIT
In DATA FRAMEs the RTR BIT has to be 'dominant'. Within a REMOTE FRAME the RTR BIT has to be 'recessive'.

CONTROL FIELD
The CONTROL FIELD consists of six bits. It includes the DATA LENGTH CODE and two bits reserved for future expansion. The reserved bits have to be sent 'dominant'. Receivers accept 'dominant' and 'recessive' bits in all combinations.

DATA LENGTH CODE
The number of bytes in the DATA FIELD is indicated by the DATA LENGTH CODE. This DATA LENGTH CODE is 4 bits wide and is transmitted within the CONTROL FIELD.



Coding of the number of data bytes by the DATA LENGTH CODE

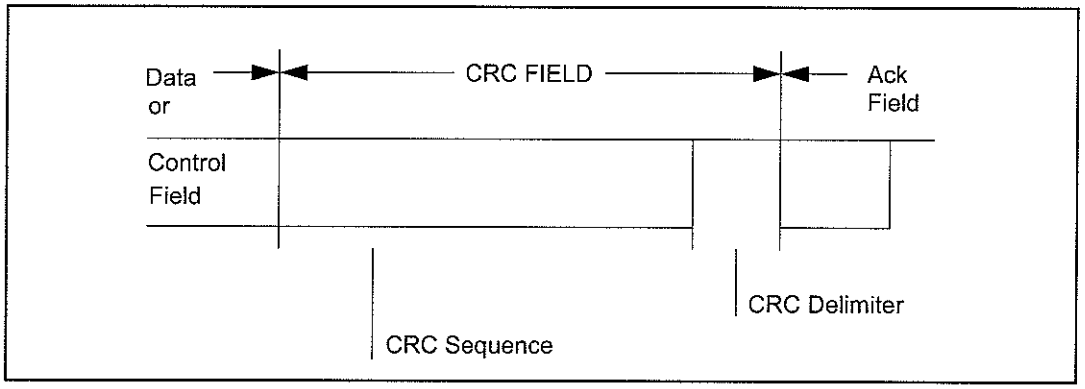
abbreviations: d 'dominant'
r 'recessive'

Number of Data Bytes	Data Length Code			
	DLC3	DLC2	DLC1	DLC0
0	d	d	d	d
1	d	d	d	r
2	d	d	r	d
3	d	d	r	r
4	d	r	d	d
5	d	r	d	r
6	d	r	r	d
7	d	r	r	r
8	r	d	d	d

DATA FRAME: admissible numbers of data bytes: {0,1,....,7,8}.
Other values may not be used.

DATA FIELD
The DATA FIELD consists of the data to be transferred within a DATA FRAME. It can contain from 0 to 8 bytes, which each contain 8 bits which are transferred MSB first.

CRC FIELD
contains the CRC SEQUENCE followed by a CRC DELIMITER.



CRC SEQUENCE
The frame check sequence is derived from a cyclic redundancy code best suited for frames with bit counts less than 127 bits (BCH Code).
In order to carry out the CRC calculation the polynomial to be divided is defined as the polynomial, the coefficients of which are given by the destuffed bit stream consisting of START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, DATA FIELD (if present) and, for the 15 lowest coefficients, by 0. This polynomial is divided (the coefficients are calculated modulo-2) by the generator-polynomial:

$$X^{15} + X^{14} + X^{10} + X^8 + X^7 + X^4 + X^3 + 1.$$

The remainder of this polynomial division is the CRC SEQUENCE transmitted over the bus. In order to implement this function, a 15 bit shift register CRC_RG(14:0) can be used. If NXTBIT denotes the next bit of the bit stream, given by the destuffed bit sequence from START OF FRAME until the end of the DATA FIELD, the CRC SEQUENCE is calculated as follows:

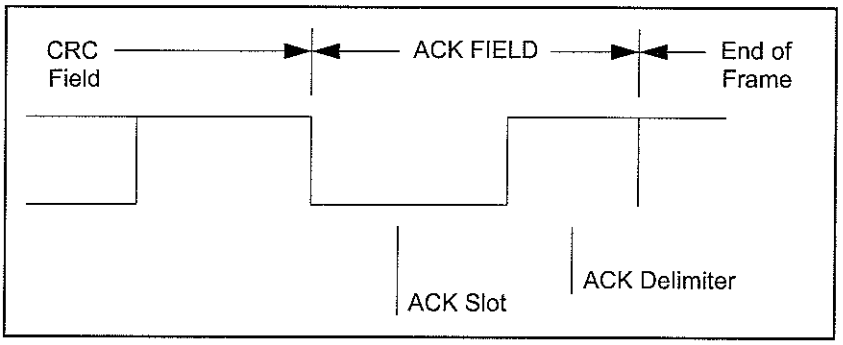
```
CRC_RG = 0;                                  // initialize shift register
REPEAT
        CRCNXT = NXTBIT EXOR CRC_RG(14);
        CRC_RG(14:1) = CRC_RG(13:0);         // shift left by
        CRC_RG(0) = 0;                       // 1 position
```

```
            IF CRCNXT THEN
                        CRC_RG(14:0) = CRC_RG(14:0) EXOR (4599hex);
            ENDIF
UNTIL (CRC SEQUENCE starts or there is an ERROR condition)
```

After the transmission / reception of the last bit of the DATA FIELD, CRC_RG contains the CRC sequence.

CRC DELIMITER
The CRC SEQUENCE is followed by the CRC DELIMITER which consists of a single 'recessive' bit.

ACK FIELD
The ACK FIELD is two bits long and contains the ACK SLOT and the ACK DELIMITER. In the ACK FIELD the transmitting station sends two 'recessive' bits.
A RECEIVER which has received a valid message correctly, reports this to the TRANSMITTER by sending a 'dominant' bit during the ACK SLOT (it sends 'ACK').



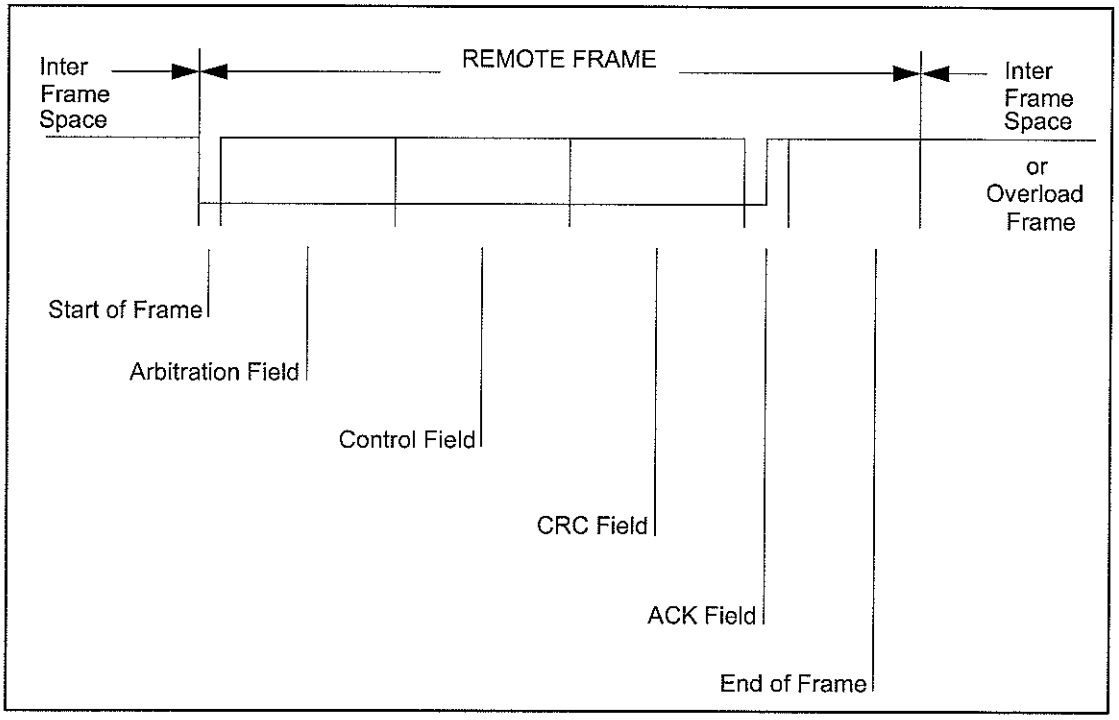
ACK SLOT
All stations having received the matching CRC SEQUENCE report this within the ACK SLOT by superscribing the 'recessive' bit of the TRANSMITTER by a 'dominant' bit.

ACK DELIMITER
The ACK DELIMITER is the second bit of the ACK FIELD and has to be a 'recessive' bit. As a consequence, the ACK SLOT is surrounded by two 'recessive' bits (CRC DELIMITER, ACK DELIMITER).

END OF FRAME
Each DATA FRAME and REMOTE FRAME is delimited by a flag sequence consisting of seven 'recessive' bits.

3.1.2 REMOTE FRAME

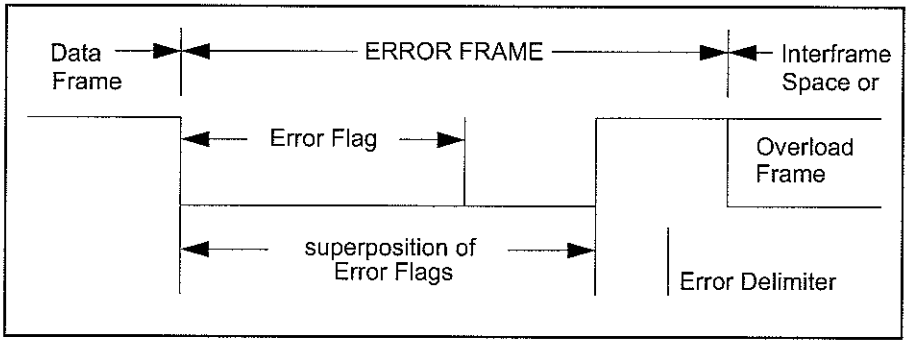
A station acting as a RECEIVER for certain data can initiate the transmission of the respective data by its source node by sending a REMOTE FRAME.
A REMOTE FRAME is composed of six different bit fields:
START OF FRAME, ARBITRATION FIELD, CONTROL FIELD, CRC FIELD, ACK FIELD, END OF FRAME.
Contrary to DATA FRAMEs, the RTR bit of REMOTE FRAMEs is 'recessive'. There is no DATA FIELD, independent of the values of the DATA LENGTH CODE which may be signed any value within the admissible range 0...8. The value is the DATA LENGTH CODE of the corresponding DATA FRAME.



The polarity of the RTR bit indicates whether a transmitted frame is a DATA FRAME (RTR bit 'dominant') or a REMOTE FRAME (RTR bit 'recessive').

3.1.3 ERROR FRAME

The ERROR FRAME consists of two different fields. The first field is given by the superposition of ERROR FLAGs contributed from different stations. The following second field is the ERROR DELIMITER.



In order to terminate an ERROR FRAME correctly, an 'error passive' node may need the bus to be 'bus idle' for at least 3 bit times (if there is a local error at an 'error passive' receiver). Therefore the bus should not be loaded to 100%.

ERROR FLAG
There are 2 forms of an ERROR FLAG: an ACTIVE ERROR FLAG and a PASSIVE ERROR FLAG.

1. The ACTIVE ERROR FLAG consists of six consecutive 'dominant' bits.

2. The PASSIVE ERROR FLAG consists of six consecutive 'recessive' bits unless it is overwritten by 'dominant' bits from other nodes.

An 'error active' station detecting an error condition signals this by transmission of an ACTIVE ERROR FLAG. The ERROR FLAG's form violates the law of bit stuffing (see CODING) applied to all fields from START OF FRAME to CRC DELIMITER or destroys the fixed form ACK FIELD or END OF FRAME field. As a consequence, all other stations detect an error condition and on their part start transmission of an ERROR FLAG. So the sequence of 'dominant' bits which actually can be monitored on the bus results from a superposition of different ERROR FLAGs transmitted by individual stations. The total length of this sequence varies between a minimum of six and a maximum of twelve bits.
An 'error passive' station detecting an error condition tries to signal this by transmission of a PASSIVE ERROR FLAG. The 'error passive' station waits for six consecutive bits

of equal polarity, beginning at the start of the PASSIVE ERROR FLAG. The PASSIVE ERROR FLAG is complete when these 6 equal bits have been detected.

ERROR DELIMITER
The ERROR DELIMITER consists of eight 'recessive' bits.
After transmission of an ERROR FLAG each station sends 'recessive' bits and monitors the bus until it detects a 'recessive' bit. Afterwards it starts transmitting seven more 'recessive' bits.

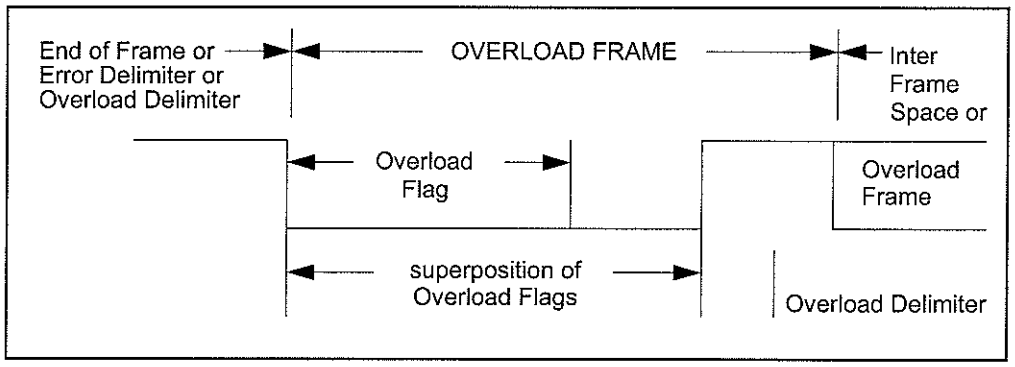
3.1.4 OVERLOAD FRAME

The OVERLOAD FRAME contains the two bit fields OVERLOAD FLAG and OVERLOAD DELIMITER.
There are two kinds of OVERLOAD conditions, which both 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 OVERLOAD condition 1 is only allowed to be started at the first bit time of an expected INTERMISSION, whereas OVERLOAD FRAMEs due to OVERLOAD condition 2 start one bit after detecting the 'dominant' bit.



At most two OVERLOAD FRAMEs may be generated to delay the next DATA or REMOTE FRAME.

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. (In case that there is a 'dominant' bit detected during the 3rd bit of INTERMISSION locally at some node, the other nodes will not interpret the OVERLOAD FLAG correctly, but interpret the first of these six 'dominant' bits as START OF FRAME. The sixth 'dominant' bit violates the rule of bit stuffing causing an error condition).

OVERLOAD DELIMITER
consists of eight 'recessive' bits.

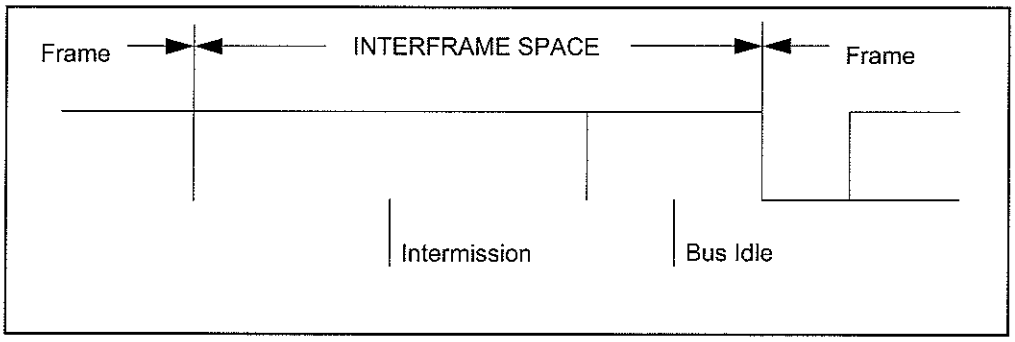
The OVERLOAD DELIMITER is of the same form as the ERROR DELIMITER. After transmission of an OVERLOAD FLAG the station monitors the bus until it detects a transition from a 'dominant' to a 'recessive' bit. At this point of time every bus station has finished sending its OVERLOAD FLAG and all stations start transmission of seven more 'recessive' bits in coincidence.

3.1.5 INTERFRAME SPACING

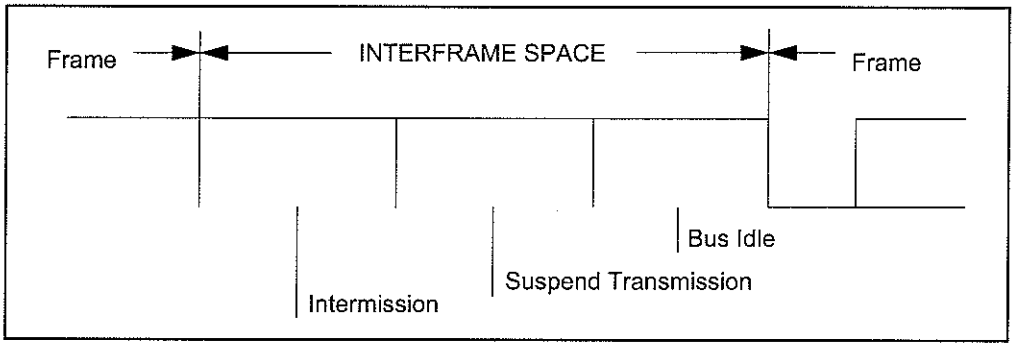
DATA FRAMEs and REMOTE FRAMEs are separated from preceding frames whatever type they are (DATA FRAME, REMOTE FRAME, ERROR FRAME, OVERLOAD FRAME) by a bit field called INTERFRAME SPACE. In contrast, OVERLOAD FRAMEs and ERROR FRAMEs are not preceded by an INTERFRAME SPACE and multiple OVERLOAD FRAMEs are not separated by an INTERFRAME SPACE.

INTERFRAME SPACE
contains the bit fields INTERMISSION and BUS IDLE and, for 'error passive' stations, which have been TRANSMITTER of the previous message, SUSPEND TRANSMISSION.

For stations which are not 'error passive' or have been RECEIVER of the previous message:



For 'error passive' stations which have been TRANSMITTER of the previous message:



INTERMISSION
consists of three 'recessive' bits.

During INTERMISSION no station is allowed to start transmission of a DATA FRAME or REMOTE FRAME. The only action to be taken is signalling an OVERLOAD condition.

BUS IDLE
The period of BUS IDLE may be of arbitrary length. The bus is recognized to be free and any station having something to transmit can access the bus. A message, which is pending for transmission during the transmission of another message, is started in the first bit following INTERMISSION.
The detection of a 'dominant' bit on the bus is interpreted as a START OF FRAME.

SUSPEND TRANSMISSION
After an 'error passive' station has transmitted a message, it sends eight 'recessive' bits following INTERMISSION, before starting to transmit a further message or recognizing the bus to be idle. If meanwhile a transmission (caused by another station) starts, the station will become receiver of this message.

3.2 Definition of TRANSMITTER / RECEIVER

TRANSMITTER
A unit originating a message is called "TRANSMITTER" of that message. The unit stays TRANSMITTER until the bus is idle or the unit loses ARBITRATION.

RECEIVER
A unit is called "RECEIVER" of a message, if it is not TRANSMITTER of that message and the bus is not idle.

The ATM Forum

BEGINNER'S OVERVIEW

GO

MEMBER LOGIN

SECTION TITLE ► LEARN ABOUT ATM

Contact Us

ABOUT ATM TECHNOLOGY
ATM STANDARDS
MEETINGS & EVENTS
LEARN ABOUT ATM
ATM IN THE NEWS
ABOUT THE FORUM
HOME

Beginners' Overview Of Asynchronous Transfer Mode (ATM)

What is ATM Technology?

Asynchronous Transfer Mode (ATM) is the world's most widely deployed backbone technology. This standards-based transport medium is widely used within the core--at the access and in the edge of telecommunications systems to send data, video and voice at ultra high speeds.

ATM is best known for its easy integration with other technologies and for its sophisticated management features that allow carriers to guarantee quality of service. These features are built into the different layers of ATM, giving the protocol an inherently robust set of controls.

Sometimes referred to as cell relay, ATM uses short, fixed-length packets called cells for transport. Information is divided among these cells, transmitted and then re-assembled at their final destination.

How does ATM fit in the telecommunications infrastructure?

A telecommunications network is designed in a series of layers. A typical configuration may have utilized a mix of time division multiplexing, Frame Relay, ATM and/or IP. Within a network, carriers often extend the characteristic strengths of ATM by blending it other technologies, such as ATM over SONET/SDH or DSL over ATM. By doing so, they extend the management features of ATM to other platforms in a very cost-effective manner.

ATM itself consists of a series of layers. The first layer - known as the adaptation layer - holds the bulk of the transmission. This 48-byte payload divides the data into different types. The ATM layer contains five bytes of additional information, referred to as overhead. This section directs the transmission. Lastly, the physical layer attaches the electrical elements and network interfaces.

How is ATM used as the backbone for other networks?

The vast majority (roughly 80 percent) of the world's carriers use ATM in the core of their networks. ATM has been widely adopted because of its unmatched flexibility in supporting the broadest array of technologies, including DSL, IP Ethernet, Frame Relay, SONET/SDH and wireless platforms. It also

acts a unique bridge between legacy equipment and the new generation of operating systems and platforms. ATM freely and easily communicates with both, allowing carriers to maximize their infrastructure investment.

ATM in the LAN (Local Area Network)

The LAN environment of a campus or building appears sheltered from the headaches associated with high-volumes of traffic that deluge larger networks. But the changes of LAN interconnection and performance are no less critical.

The ATM/LAN relationship recently took a giant step forward when a prominent U.S. vendor announced a patent for its approach to extending ATM's quality of service to the LAN. The filing signals another birth in a long lineage of applications that prove the staying power and adaptability of ATM.

ATM is a proven technology that is now in its fourth generation of switches. Its maturity alone is not its greatest asset. Its strength is in its ability to anticipate the market and quickly respond, doing so with the full confidence of the industry behind it.

ATM in the WAN (Wide Area Network)

A blend of ATM, IP and Ethernet options abound in the wide area network. But no other technology can replicate ATM's mix of universal support and enviable management features. Carriers inevitably turn to ATM when they need high-speed transport in the core coupled with the security of a guaranteed level of quality of service. When those same carriers expand to the WAN, the vast majority does so with an ATM layer.

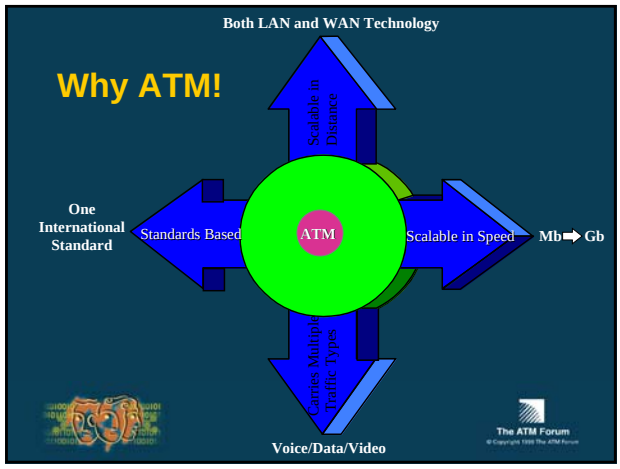
Distance can be a problem for some high-speed platforms. Not so with ATM. The integrity of the transport signal is maintained even when different kinds of traffic are traversing the same network. And because of its ability to scale up to OC-48, different services can be offered at varying speeds and at a range of performance levels.

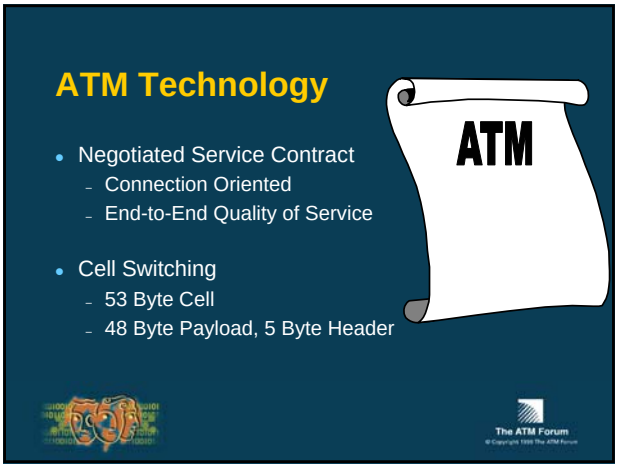
ATM in the MAN (Metropolitan Area Network)

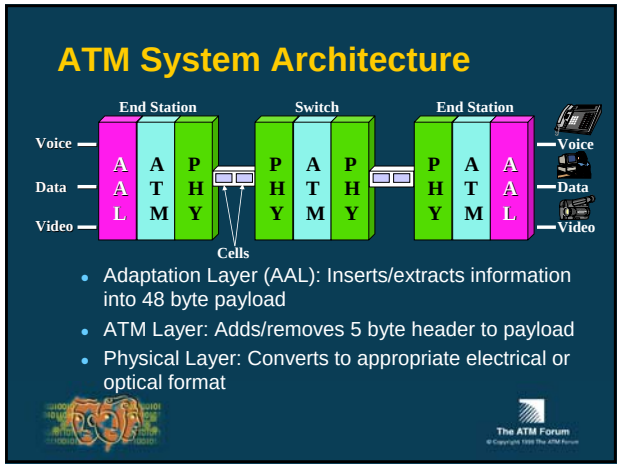
The MAN is one of the hottest growing areas in data and telecommunications. Traffic may not travel more than a few miles within a MAN, but it's generally doing so over leading edge technologies and at faster-than-lightening speeds.

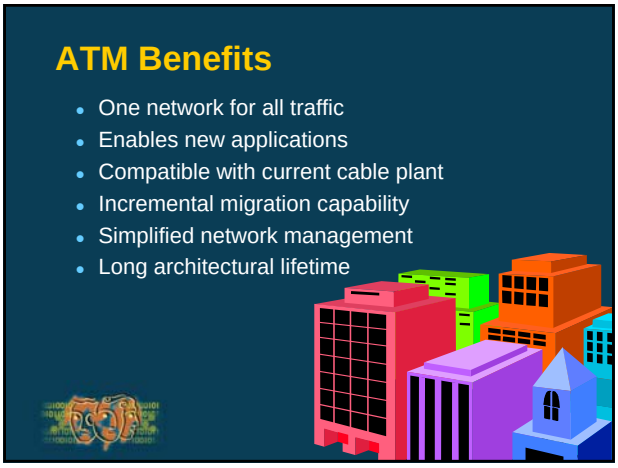
The typical MAN configuration is a point of convergence for many different types of traffic that are generated by many different sources. The beauty of ATM in the MAN is that it easily accommodates these divergent transmissions, often times bridging legacy equipment with ultra high-speed networks. Today, ATM scales form T-1 to OC-48 at speeds that average 2.5 Gb/s in operation, 10 Gb/s in limited use and spanning up to 40 Gb/s in trials.

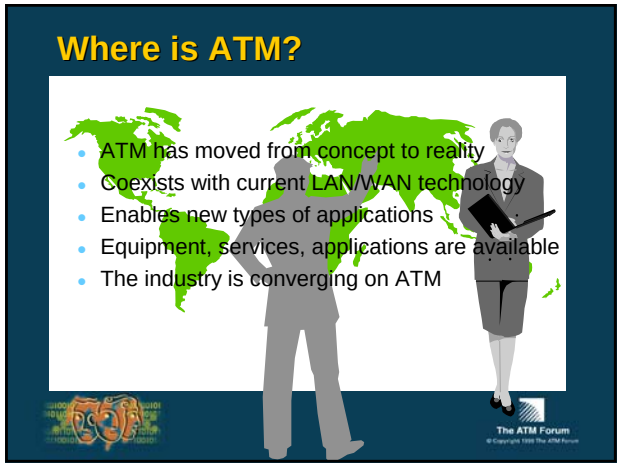

About ATM
The ATM Forum


Both LAN and WAN Technology
Why ATM!
Scalable in Distance
One International Standard
Standards Based
ATM
Scalable in Speed
Mb Gb
Carries Multiple Traffic Types
Voice/Data/Video
The ATM Forum


ATM Technology
Negotiated Service Contract
Connection Oriented
End-to-End Quality of Service
Cell Switching
53 Byte Cell
48 Byte Payload, 5 Byte Header
ATM
The ATM Forum


ATM System Architecture
End Station
Switch
End Station
Voice
Data
Video
AAL
ATM
PHY
Cells
Adaptation Layer (AAL): Inserts/extracts information into 48 byte payload
ATM Layer: Adds/removes 5 byte header to payload
Physical Layer: Converts to appropriate electrical or optical format
The ATM Forum


ATM Benefits
One network for all traffic
Enables new applications
Compatible with current cable plant
Incremental migration capability
Simplified network management
Long architectural lifetime


Where is ATM?
ATM has moved from concept to reality
Coexists with current LAN/WAN technology
Enables new types of applications
Equipment, services, applications are available
The industry is converging on ATM
The ATM Forum

1. Introduction

1.1 Document Purpose

This document specifies the *Universal Test & Operations PHY Interface for ATM* (UTOPIA) data path interface. UTOPIA defines the interface between the *Physical Layer* (PHY) and upper layer modules such as the ATM Layer, and various management entities. The definition allows a common PHY interface in ATM subsystems across a wide range of speeds and media types. This definition covers connection to devices supporting ATM PHY specifications from sub-100 Mbps to 155 Mbps, and provides guidelines for 622 Mbps.

1.2 Document Scope

The UTOPIA *data path* specification defines two interface signal groups and general ATM and PHY layer device capabilities. The two interface signal groups are: Transmit and Receive. The device capability specification defines minimum capabilities primarily of the PHY layer device, but secondarily the ATM (SAR) device also. The definition of minimum device capabilities is required to allow different PHY layer devices to interface to a common ATM layer device. Figure 1 shows the relationship of this interface to ATM and PHY components of an ATM subsystem. The shaded interface groups are defined by this specification.

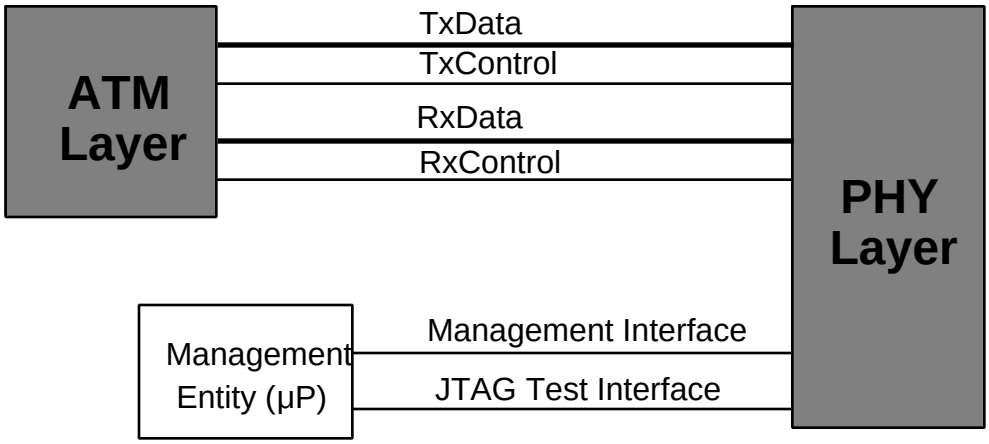


Figure 1. UTOPIA Interface Diagram

1.3 Current Level

This level of the specification covers the following two scenarios.

1. an 8-bit wide data path, using an octet-level handshake, operating up to 25 MHz, with a single PHY device
2. an 8-bit wide data path, using a cell-level handshake, operating up to 25 MHz, with a single PHY device

2. Motivation and Goals

The fundamental goal of this specification is to define a common, standard interface between ATM and PHY layers of ATM subsystems. The motivations for such a proposal are given below. All ATM developers wish to leverage their investment cost and development effort. There are multiple PHY layers defined for ATM (defined by ANSI, the ITU and the ATM Forum). Each of the PHY layers is potentially supportable by a common ATM layer. Consequently, one motivation of this proposal is to leverage ATM development across multiple PHY types.

The ATM-PHY interface is not a UNI (User Network Interface) nor NNI (Network Network Interface) specification, so is not directly relevant to interoperability. Therefore, no one is directly required to adhere to any specific interface definition. However, the economic advantages of multiple PHY layer parts that are interface compatible cannot be ignored. Correspondingly, another motivation for this is to encourage the adoption of this data path interface, implementable as a chip-internal, chip-to-chip, or even board-to-board interface.

A second goal is to allow expansion of the interface FIFO's using industry standard devices. Due to the differing data rates of the various ATM PHY layers, it is necessary to provide rate matching buffers (i.e. FIFO's) between ATM and PHY layers.

A third goal is to define an interface that supports a streaming mode of operation, where both the ATM and PHY layers have the capability to throttle the actual transfer rate.

A fourth goal is to support rates from sub-100 Mbps, up to 155 Mbps with a common interface, requiring only an 8-bit wide data path. There is also an intention to provide support for higher rates (e.g. 622 Mbps) by extending the data path width and increasing the transfer rate. A proposed method for such an extension is described in section 6[1].

[1]It is recognized that this document (Level 1) does not fully address this issue; the text is for guidance only.

Chapter 3
Background

This chapter presents a brief description of the background of the Universal Serial Bus (USB), including design goals, features of the bus, and existing technologies.

3.1 Goals for the Universal Serial Bus

The USB is specified to be an industry-standard extension to the PC architecture with a focus on PC peripherals that enable consumer and business applications. The following criteria were applied in defining the architecture for the USB:

- Ease-of-use for PC peripheral expansion
- Low-cost solution that supports transfer rates up to 480 Mb/s
- Full support for real-time data for voice, audio, and video
- Protocol flexibility for mixed-mode isochronous data transfers and asynchronous messaging
- Integration in commodity device technology
- Comprehension of various PC configurations and form factors
- Provision of a standard interface capable of quick diffusion into product
- Enabling new classes of devices that augment the PC's capability
- Full backward compatibility of USB 2.0 for devices built to previous versions of the specification

3.2 Taxonomy of Application Space

Figure 3-1 describes a taxonomy for the range of data traffic workloads that can be serviced over a USB. As can be seen, a 480 Mb/s bus comprehends the high-speed, full-speed, and low-speed data ranges. Typically, high-speed and full-speed data types may be isochronous, while low-speed data comes from interactive devices. The USB is primarily a PC bus but can be readily applied to other host-centric computing devices. The software architecture allows for future extension of the USB by providing support for multiple USB Host Controllers.

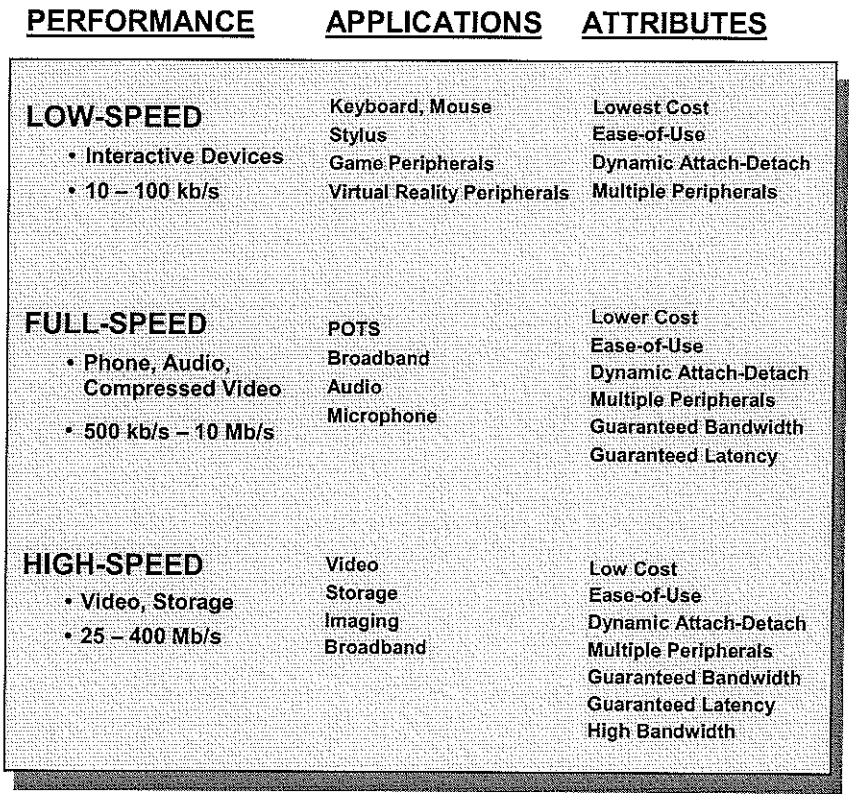


Figure 3-1. Application Space Taxonomy

3.3 Feature List

The USB Specification provides a selection of attributes that can achieve multiple price/performance integration points and can enable functions that allow differentiation at the system and component level. Features are categorized by the following benefits:

Easy to use for end user

- Single model for cabling and connectors
- Electrical details isolated from end user (e.g., bus terminations)
- Self-identifying peripherals, automatic mapping of function to driver and configuration
- Dynamically attachable and reconfigurable peripherals

Wide range of workloads and applications

- Suitable for device bandwidths ranging from a few kb/s to several hundred Mb/s
- Supports isochronous as well as asynchronous transfer types over the same set of wires
- Supports concurrent operation of many devices (multiple connections)
- Supports up to 127 physical devices
- Supports transfer of multiple data and message streams between the host and devices
- Allows compound devices (i.e., peripherals composed of many functions)
- Lower protocol overhead, resulting in high bus utilization

Isochronous bandwidth

- Guaranteed bandwidth and low latencies appropriate for telephony, audio, video, etc.

Flexibility

- Supports a wide range of packet sizes, which allows a range of device buffering options
- Allows a wide range of device data rates by accommodating packet buffer size and latencies
- Flow control for buffer handling is built into the protocol

Robustness

- Error handling/fault recovery mechanism is built into the protocol
- Dynamic insertion and removal of devices is identified in user-perceived real-time
- Supports identification of faulty devices

Synergy with PC industry

- Protocol is simple to implement and integrate
- Consistent with the PC plug-and-play architecture
- Leverages existing operating system interfaces

Low-cost implementation

- Low-cost subchannel at 1.5 Mb/s
- Optimized for integration in peripheral and host hardware
- Suitable for development of low-cost peripherals
- Low-cost cables and connectors
- Uses commodity technologies

Upgrade path

- Architecture upgradeable to support multiple USB Host Controllers in a system

Chapter 4
Architectural Overview

This chapter presents an overview of the Universal Serial Bus (USB) architecture and key concepts. The USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host-scheduled, token-based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation.

Later chapters describe the various components of the USB in greater detail.

4.1 USB System Description

A USB system is described by three definitional areas:

- USB interconnect
- USB devices
- USB host

The USB interconnect is the manner in which USB devices are connected to and communicate with the host. This includes the following:

- Bus Topology: Connection model between USB devices and the host.
- Inter-layer Relationships: In terms of a capability stack, the USB tasks that are performed at each layer in the system.
- Data Flow Models: The manner in which data moves in the system over the USB between producers and consumers.
- USB Schedule: The USB provides a shared interconnect. Access to the interconnect is scheduled in order to support isochronous data transfers and to eliminate arbitration overhead.

USB devices and the USB host are described in detail in subsequent sections.

4.1.1 Bus Topology

The USB connects USB devices with the USB host. The USB physical interconnect is a tiered star topology. A hub is at the center of each star. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or function. Figure 4-1 illustrates the topology of the USB.

Due to timing constraints allowed for hub and cable propagation times, the maximum number of tiers allowed is seven (including the root tier). Note that in seven tiers, five non-root hubs maximum can be supported in a communication path between the host and any device. A compound device (see Figure 4-1) occupies two tiers; therefore, it cannot be enabled if attached at tier level seven. Only functions can be enabled in tier seven.

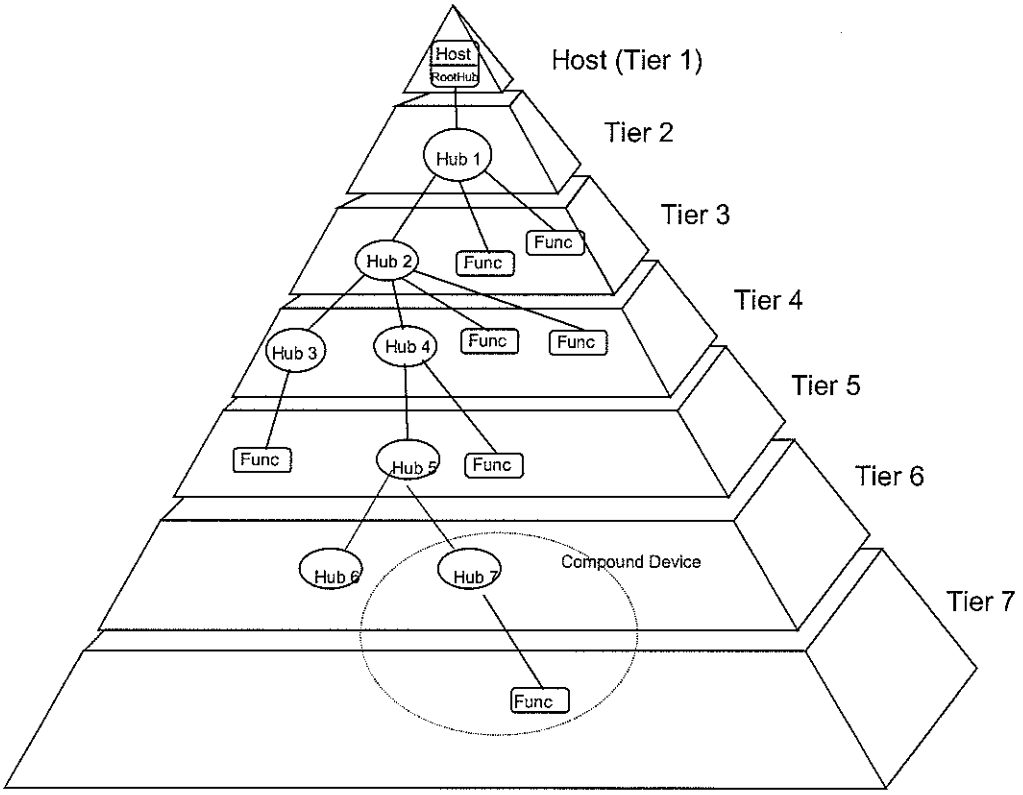


Figure 4-1. Bus Topology

4.1.1.1 USB Host

There is only one host in any USB system. The USB interface to the host computer system is referred to as the Host Controller. The Host Controller may be implemented in a combination of hardware, firmware, or software. A root hub is integrated within the host system to provide one or more attachment points.

Additional information concerning the host may be found in Section 4.9 and in Chapter 10.

4.1.1.2 USB Devices

USB devices are one of the following:

- Hubs, which provide additional attachment points to the USB
- Functions, which provide capabilities to the system, such as an ISDN connection, a digital joystick, or speakers

USB devices present a standard USB interface in terms of the following:

- Their comprehension of the USB protocol
- Their response to standard USB operations, such as configuration and reset
- Their standard capability descriptive information

Additional information concerning USB devices may be found in Section 4.8 and in Chapter 9.

4.2 Physical Interface

The physical interface of the USB is described in the electrical (Chapter 7) and mechanical (Chapter 6) specifications for the bus.

4.2.1 Electrical

The USB transfers signal and power over a four-wire cable, shown in Figure 4-2. The signaling occurs over two wires on each point-to-point segment.

There are three data rates:

- The USB high-speed signaling bit rate is 480 Mb/s.
- The USB full-speed signaling bit rate is 12 Mb/s.
- A limited capability low-speed signaling mode is also defined at 1.5 Mb/s.

USB 2.0 host controllers and hubs provide capabilities so that full-speed and low-speed data can be transmitted at high-speed between the host controller and the hub, but transmitted between the hub and the device at full-speed or low-speed. This capability minimizes the impact that full-speed and low-speed devices have upon the bandwidth available for high-speed devices.

The low-speed mode is defined to support a limited number of low-bandwidth devices, such as mice, because more general use would degrade bus utilization.

The clock is transmitted, encoded along with the differential data. The clock encoding scheme is NRZI with bit stuffing to ensure adequate transitions. A SYNC field precedes each packet to allow the receiver(s) to synchronize their bit recovery clocks.

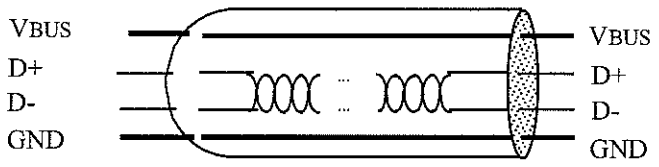


Figure 4-2. USB Cable

The cable also carries VBUS and GND wires on each segment to deliver power to devices. VBUS is nominally +5 V at the source. The USB allows cable segments of variable lengths, up to several meters, by choosing the appropriate conductor gauge to match the specified IR drop and other attributes such as device power budget and cable flexibility. In order to provide guaranteed input voltage levels and proper termination impedance, biased terminations are used at each end of the cable. The terminations also permit the detection of attach and detach at each port and differentiate between high/full-speed and low-speed devices.

4.2.2 Mechanical

The mechanical specifications for cables and connectors are provided in Chapter 6. All devices have an upstream connection. Upstream and downstream connectors are not mechanically interchangeable, thus eliminating illegal loopback connections at hubs. The cable has four conductors: a twisted signal pair of standard gauge and a power pair in a range of permitted gauges. The connector is four-position, with shielded housing, specified robustness, and ease of attach-detach characteristics.

4.3 Power

The specification covers two aspects of power:

- Power distribution over the USB deals with the issues of how USB devices consume power provided by the host over the USB.
- Power management deals with how the USB System Software and devices fit into the host-based power management system.

4.3.1 Power Distribution

Each USB segment provides a limited amount of power over the cable. The host supplies power for use by USB devices that are directly connected. In addition, any USB device may have its own power supply. USB devices that rely totally on power from the cable are called bus-powered devices. In contrast, those that have an alternate source of power are called self-powered devices. A hub also supplies power for its connected USB devices. The architecture permits bus-powered hubs within certain constraints of topology that are discussed later in Chapter 11.

4.3.2 Power Management

A USB host may have a power management system that is independent of the USB. The USB System Software interacts with the host's power management system to handle system power events such as suspend or resume. Additionally, USB devices typically implement additional power management features that allow them to be power managed by system software.

The power distribution and power management features of the USB allow it to be designed into power-sensitive systems such as battery-based notebook computers.

4.4 Bus Protocol

The USB is a polled bus. The Host Controller initiates all data transfers.

Most bus transactions involve the transmission of up to three packets. Each transaction begins when the Host Controller, on a scheduled basis, sends a USB packet describing the type and direction of transaction, the USB device address, and endpoint number. This packet is referred to as the "token packet." The USB device that is addressed selects itself by decoding the appropriate address fields. In a given transaction, data is transferred either from the host to a device or from a device to the host. The direction of data transfer is specified in the token packet. The source of the transaction then sends a data packet or indicates it has no

data to transfer. The destination, in general, responds with a handshake packet indicating whether the transfer was successful.

Some bus transactions between host controllers and hubs involve the transmission of four packets. These types of transactions are used to manage the data transfers between the host and full-/low- speed devices.

The USB data transfer model between a source or destination on the host and an endpoint on a device is referred to as a pipe. There are two types of pipes: stream and message. Stream data has no USB-defined structure, while message data does. Additionally, pipes have associations of data bandwidth, transfer service type, and endpoint characteristics like directionality and buffer sizes. Most pipes come into existence when a USB device is configured. One message pipe, the Default Control Pipe, always exists once a device is powered, in order to provide access to the device's configuration, status, and control information.

The transaction schedule allows flow control for some stream pipes. At the hardware level, this prevents buffers from underrun or overrun situations by using a NAK handshake to throttle the data rate. When NAKed, a transaction is retried when bus time is available. The flow control mechanism permits the construction of flexible schedules that accommodate concurrent servicing of a heterogeneous mix of stream pipes. Thus, multiple stream pipes can be serviced at different intervals and with packets of different sizes.

4.5 Robustness

There are several attributes of the USB that contribute to its robustness:

- Signal integrity using differential drivers, receivers, and shielding
- CRC protection over control and data fields
- Detection of attach and detach and system-level configuration of resources
- Self-recovery in protocol, using timeouts for lost or corrupted packets
- Flow control for streaming data to ensure isochrony and hardware buffer management
- Data and control pipe constructs for ensuring independence from adverse interactions between functions

4.5.1 Error Detection

The core bit error rate of the USB medium is expected to be close to that of a backplane and any glitches will very likely be transient in nature. To provide protection against such transients, each packet includes error protection fields. When data integrity is required, such as with lossless data devices, an error recovery procedure may be invoked in hardware or software.

The protocol includes separate CRCs for control and data fields of each packet. A failed CRC is considered to indicate a corrupted packet. The CRC gives 100% coverage on single- and double-bit errors.

4.5.2 Error Handling

The protocol allows for error handling in hardware or software. Hardware error handling includes reporting and retry of failed transfers. A USB Host Controller will try a transmission that encounters errors up to three times before informing the client software of the failure. The client software can recover in an implementation-specific way.

4.6 System Configuration

The USB supports USB devices attaching to and detaching from the USB at any time. Consequently, system software must accommodate dynamic changes in the physical bus topology.

4.6.1 Attachment of USB Devices

All USB devices attach to the USB through ports on specialized USB devices known as hubs. Hubs have status bits that are used to report the attachment or removal of a USB device on one of its ports. The host queries the hub to retrieve these bits. In the case of an attachment, the host enables the port and addresses the USB device through the device's control pipe at the default address.

The host assigns a unique USB address to the device and then determines if the newly attached USB device is a hub or a function. The host establishes its end of the control pipe for the USB device using the assigned USB address and endpoint number zero.

If the attached USB device is a hub and USB devices are attached to its ports, then the above procedure is followed for each of the attached USB devices.

If the attached USB device is a function, then attachment notifications will be handled by host software that is appropriate for the function.

4.6.2 Removal of USB Devices

When a USB device has been removed from one of a hub's ports, the hub disables the port and provides an indication of device removal to the host. The removal indication is then handled by appropriate USB System Software. If the removed USB device is a hub, the USB System Software must handle the removal of both the hub and of all of the USB devices that were previously attached to the system through the hub.

4.6.3 Bus Enumeration

Bus enumeration is the activity that identifies and assigns unique addresses to devices attached to a bus. Because the USB allows USB devices to attach to or detach from the USB at any time, bus enumeration is an on-going activity for the USB System Software. Additionally, bus enumeration for the USB also includes the detection and processing of removals.

4.7 Data Flow Types

The USB supports functional data and control exchange between the USB host and a USB device as a set of either uni-directional or bi-directional pipes. USB data transfers take place between host software and a particular endpoint on a USB device. Such associations between the host software and a USB device endpoint are called pipes. In general, data movement though one pipe is independent from the data flow in any other pipe. A given USB device may have many pipes. As an example, a given USB device could have an endpoint that supports a pipe for transporting data to the USB device and another endpoint that supports a pipe for transporting data from the USB device.

The USB architecture comprehends four basic types of data transfers:

- Control Transfers: Used to configure a device at attach time and can be used for other device-specific purposes, including control of other pipes on the device.
- Bulk Data Transfers: Generated or consumed in relatively large and bursty quantities and have wide dynamic latitude in transmission constraints.
- Interrupt Data Transfers: Used for timely but reliable delivery of data, for example, characters or coordinates with human-perceptible echo or feedback response characteristics.
- Isochronous Data Transfers: Occupy a prenegotiated amount of USB bandwidth with a prenegotiated delivery latency. (Also called streaming real time transfers).

A pipe supports only one of the types of transfers described above for any given device configuration. The USB data flow model is described in more detail in Chapter 5.

4.7.1 Control Transfers

Control data is used by the USB System Software to configure devices when they are first attached. Other driver software can choose to use control transfers in implementation-specific ways. Data delivery is lossless.

4.7.2 Bulk Transfers

Bulk data typically consists of larger amounts of data, such as that used for printers or scanners. Bulk data is sequential. Reliable exchange of data is ensured at the hardware level by using error detection in hardware and invoking a limited number of retries in hardware. Also, the bandwidth taken up by bulk data can vary, depending on other bus activities.

4.7.3 Interrupt Transfers

A limited-latency transfer to or from a device is referred to as interrupt data. Such data may be presented for transfer by a device at any time and is delivered by the USB at a rate no slower than is specified by the device.

Interrupt data typically consists of event notification, characters, or coordinates that are organized as one or more bytes. An example of interrupt data is the coordinates from a pointing device. Although an explicit timing rate is not required, interactive data may have response time bounds that the USB must support.

4.7.4 Isochronous Transfers

Isochronous data is continuous and real-time in creation, delivery, and consumption. Timing-related information is implied by the steady rate at which isochronous data is received and transferred. Isochronous data must be delivered at the rate received to maintain its timing. In addition to delivery rate, isochronous data may also be sensitive to delivery delays. For isochronous pipes, the bandwidth required is typically based upon the sampling characteristics of the associated function. The latency required is related to the buffering available at each endpoint.

A typical example of isochronous data is voice. If the delivery rate of these data streams is not maintained, drop-outs in the data stream will occur due to buffer or frame underruns or overruns. Even if data is delivered at the appropriate rate by USB hardware, delivery delays introduced by software may degrade applications requiring real-time turn-around, such as telephony-based audio conferencing.

The timely delivery of isochronous data is ensured at the expense of potential transient losses in the data stream. In other words, any error in electrical transmission is not corrected by hardware mechanisms such as retries. In practice, the core bit error rate of the USB is expected to be small enough not to be an issue. USB isochronous data streams are allocated a dedicated portion of USB bandwidth to ensure that data can be delivered at the desired rate. The USB is also designed for minimal delay of isochronous data transfers.

4.7.5 Allocating USB Bandwidth

USB bandwidth is allocated among pipes. The USB allocates bandwidth for some pipes when a pipe is established. USB devices are required to provide some buffering of data. It is assumed that USB devices requiring more bandwidth are capable of providing larger buffers. The goal for the USB architecture is to ensure that buffering-induced hardware delay is bounded to within a few milliseconds.

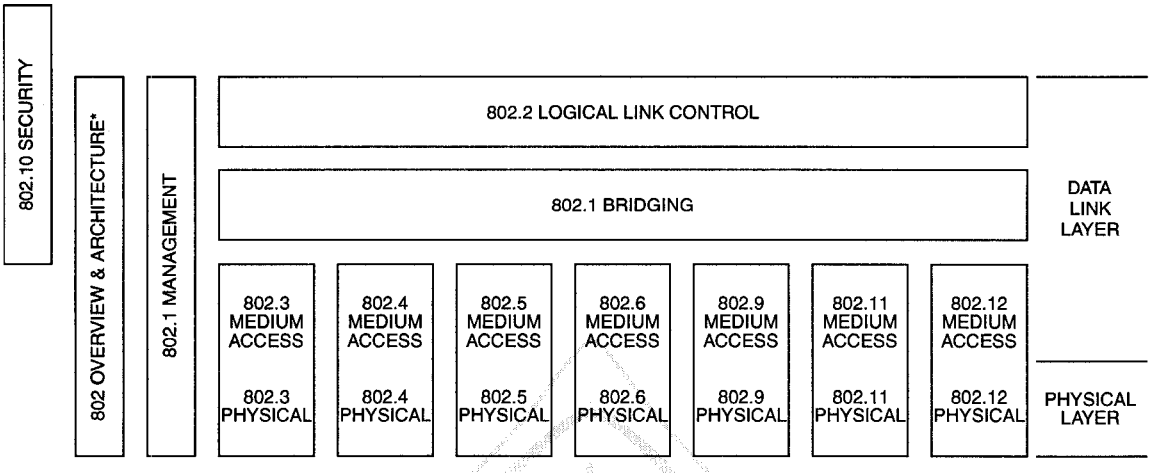
The USB's bandwidth capacity can be allocated among many different data streams. This allows a wide range of devices to be attached to the USB. Further, different device bit rates, with a wide dynamic range, can be concurrently supported.

The USB Specification defines the rules for how each transfer type is allowed access to the bus.

Introduction

(This introduction is not part of IEEE Std 802.11-1997, but is included for information only.)

This standard is part of a family of standards for local and metropolitan area networks. The relationship between the standard and other members of the family is shown below. (The numbers in the figure refer to IEEE standard numbers.)



* Formerly IEEE Std 802.1A.

This family of standards deals with the physical and data link layers as defined by the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Open Systems Interconnection Basic Reference Model (ISO/IEC 7498-1: 1994). The access standards define several types of medium access technologies and associated physical media, each appropriate for particular applications or system objectives. Other types are under investigation.

The standards defining the access technologies are as follows:

- IEEE Std 802 — *Overview and Architecture.* This standard provides an overview to the family of IEEE 802 Standards. This document forms part of the 802.1 scope of work.
- ANSI/IEEE Std 802.1B and 802.1k [ISO/IEC 15802-2] — *LAN/MAN Management.* Defines an Open Systems Interconnection (OSI) management-compatible architecture, and services and protocol elements for use in a LAN/MAN environment for performing remote management.
- ANSI/IEEE Std 802.1D [ISO/IEC 10038] — *MAC Bridging.* Specifies an architecture and protocol for the interconnection of IEEE 802 LANs below the MAC service boundary.
- ANSI/IEEE Std 802.1E [ISO/IEC 15802-4] — *System Load Protocol.* Specifies a set of services and protocol for those aspects of management concerned with the loading of systems on IEEE 802 LANs.
- ANSI/IEEE Std 802.2 [ISO/IEC 8802-2] — *Logical Link Control*
- ANSI/IEEE Std 802.3 [ISO/IEC 8802-3] — *CSMA/CD Access Method and Physical Layer Specifications*
- ANSI/IEEE Std 802.4 [ISO/IEC 8802-4] — *Token Passing Bus Access Method and Physical Layer Specifications*

5. General description

5.1 General description of the architecture

This subclause presents the concepts and terminology used within IEEE Std 802.11-1997 (referred to throughout the text as IEEE 802.11). Specific terms are defined in Clause 3. Illustrations convey key IEEE 802.11 concepts and the interrelationships of the architectural components. IEEE 802.11 uses an architecture to describe functional components of an IEEE 802.11 LAN. The architectural descriptions are not intended to represent any specific physical implementation of IEEE 802.11.

5.1.1 How wireless LAN systems are different

Wireless networks have fundamental characteristics that make them significantly different from traditional wired LANs.

5.1.1.1 Destination address does not equal destination location

In wired LANs, an address is equivalent to a physical location. This is implicitly assumed in the design of wired LANs. In IEEE 802.11, the addressable unit is a station (STA). The STA is a message destination, but not (in general) a fixed location.

5.1.1.2 The media impact the design

The physical layers used in IEEE 802.11 are fundamentally different from wired media. Thus IEEE 802.11 PHYs

a) Use a medium that has neither absolute nor readily observable boundaries outside of which stations with conformant PHY transceivers are known to be unable to receive network frames.
b) Are unprotected from outside signals.
c) Communicate over a medium significantly less reliable than wired PHYs.
d) Have dynamic topologies.
e) Lack full connectivity, and therefore the assumption normally made that every STA can hear every other STA is invalid (that is, STAs may be "hidden" from each other).
f) Have time-varying and asymmetric propagation properties.

Because of limitations on wireless PHY ranges, wireless LANs intended to cover reasonable geographic distances may be built from basic coverage building blocks.

5.1.1.3 The impact of handling mobile stations

One of the requirements of IEEE 802.11 is to handle *mobile* as well as *portable* stations. A *portable* station is one that is moved from location to location, but is only used while at a fixed location. *Mobile* stations actually access the LAN while in motion.

For technical reasons, it is not sufficient to handle only portable stations. Propagation effects blur the distinction between portable and mobile stations; stationary stations often appear to be mobile due to propagation effects.

Another aspect of mobile stations is that they may often be battery powered. Hence power management is an important consideration. For example, it cannot be presumed that a station's receiver will always be powered on.

5.1.1.4 Interaction with other IEEE 802 layers

IEEE 802.11 is required to appear to higher layers [logical link control (LLC)] as a current style 802 LAN. This requires that the IEEE 802.11 network handle station mobility within the MAC sublayer. To meet reliability assumptions (that LLC makes about lower layers), it is necessary for IEEE 802.11 to incorporate functionality that is untraditional for MAC sublayers.

5.2 Components of the IEEE 802.11 architecture

The IEEE 802.11 architecture consists of several components that interact to provide a wireless LAN that supports station mobility transparently to upper layers.

The *basic service set* (BSS) is the basic building block of an IEEE 802.11 LAN. Figure 1 shows two BSSs, each of which has two stations that are members of the BSS.

It is useful to think of the ovals used to depict a BSS as the coverage area within which the member stations of the BSS may remain in communication. (The concept of area, while not precise, is often good enough.) If a station moves out of its BSS, it can no longer directly communicate with other members of the BSS.

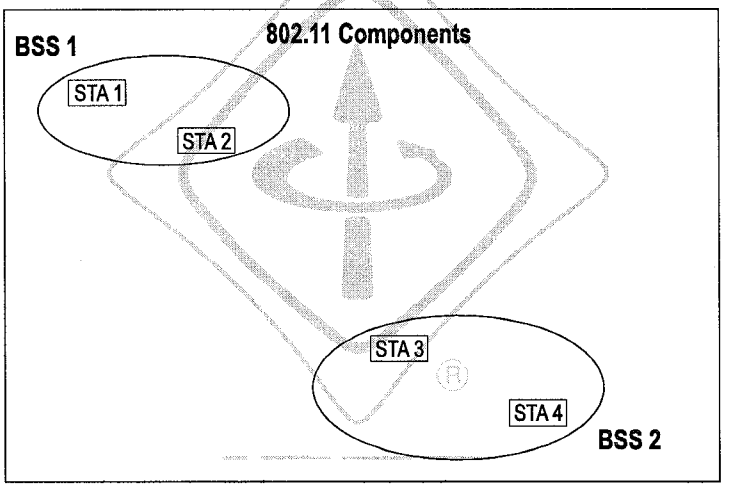


Figure 1—Basic service sets

5.2.1 The independent BSS as an ad hoc network

The independent BSS (IBSS) is the most basic type of IEEE 802.11 LAN. A minimum IEEE 802.11 LAN may consist of only two stations.

Figure 1 shows two IBSSs. This mode of operation is possible when IEEE 802.11 stations are able to communicate directly. Because this type of IEEE 802.11 LAN is often formed without pre-planning, for only as long as the LAN is needed, this type of operation is often referred to as an *ad hoc network.*

5.2.1.1 STA to BSS association is dynamic

The association between a STA and a BSS is dynamic (STAs turn on, turn off, come within range, and go out of range). To become a member of an infrastructure BSS, a station shall become "associated." These associations are dynamic and involve the use of the distribution system service (DSS), which is described later.

5.2.2 Distribution system concepts

PHY limitations determine the direct station-to-station distance that may be supported. For some networks this distance is sufficient; for other networks, increased coverage is required.

Instead of existing independently, a BSS may also form a component of an extended form of network that is built with multiple BSSs. The architectural component used to interconnect BSSs is the *distribution system* (DS).

IEEE 802.11 logically separates the wireless medium (WM) from the distribution system medium (DSM). Each logical medium is used for different purposes, by a different component of the architecture. The IEEE 802.11 definitions neither preclude, nor demand, that the multiple media be either the same or different.

Recognizing that the multiple media are *logically* different is key to understanding the flexibility of the architecture. The IEEE 802.11 LAN architecture is specified independently of the physical characteristics of any specific implementation.

The DS enables mobile device support by providing the logical services necessary to handle address to destination mapping and seamless integration of multiple BSSs.

An *access point* (AP) is a STA that provides access to the DS by providing DS services in addition to acting as a STA.

Figure 2 adds the DS and AP components to the IEEE 802.11 architecture picture.

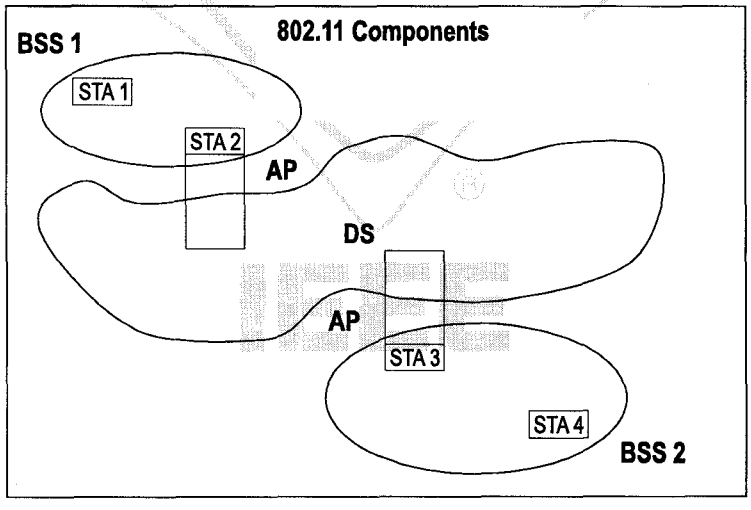


Figure 2—Distribution systems and access points

Data move between a BSS and the DS via an AP. Note that all APs are also STAs; thus they are addressable entities. The addresses used by an AP for communication on the WM and on the DSM are not necessarily the same.

5.2.2.1 Extended service set (ESS): The large coverage network

The DS and BSSs allow IEEE 802.11 to create a wireless network of arbitrary size and complexity. IEEE 802.11 refers to this type of network as the *extended service set* (ESS) network.

The key concept is that the ESS network appears the same to an LLC layer as an IBSS network. Stations within an ESS may communicate and mobile stations may move from one BSS to another (within the same ESS) transparently to LLC.

Nothing is assumed by IEEE 802.11 about the relative physical locations of the BSSs in Figure 3.

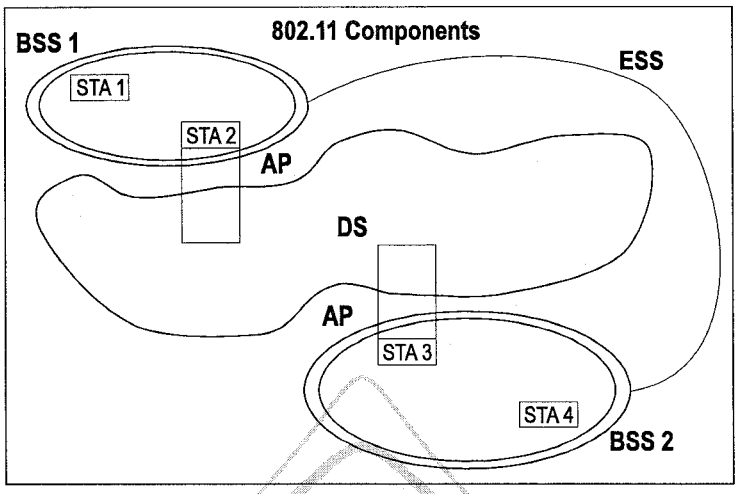


Figure 3—Extended service set

All of the following are possible:

a) The BSSs may partially overlap. This is commonly used to arrange contiguous coverage within a physical volume.

b) The BSSs could be physically disjointed. Logically there is no limit to the distance between BSSs.

c) The BSSs may be physically collocated. This may be done to provide redundancy.

d) One (or more) IBSS or ESS networks may be physically present in the same space as one (or more) ESS networks. This may arise for a number of reasons. Two of the most common are when an ad hoc network is operating in a location that also has an ESS network, and when physically overlapping IEEE 802.11 networks have been set up by different organizations.

5.2.3 Area concepts

For wireless PHYs, well-defined coverage areas simply do not exist. Propagation characteristics are dynamic and unpredictable. Small changes in position or direction may result in dramatic differences in signal strength. Similar effects occur whether a station is stationary or mobile (as moving objects may impact station-to-station propagation).

Figure 4 shows a signal strength map for a simple square room with a standard metal desk and an open doorway. Figure 4 is a static snapshot; the propagation patterns change dynamically as stations and objects in the environment move. In Figure 4 the dark (solid) blocks in the lower left are a metal desk and there is a doorway at the top right of the figure. The figure indicates relative differences in field strength with different intensities and indicates the variability of field strength even in a static environment.

While the architecture diagrams show sharp boundaries for BSSs, this is an artifact of the pictorial representation, not a physical reality. Since dynamic three-dimensional field strength pictures are difficult to draw, well-defined shapes are used by IEEE 802.11 architectural diagrams to represent the coverage of a BSS.

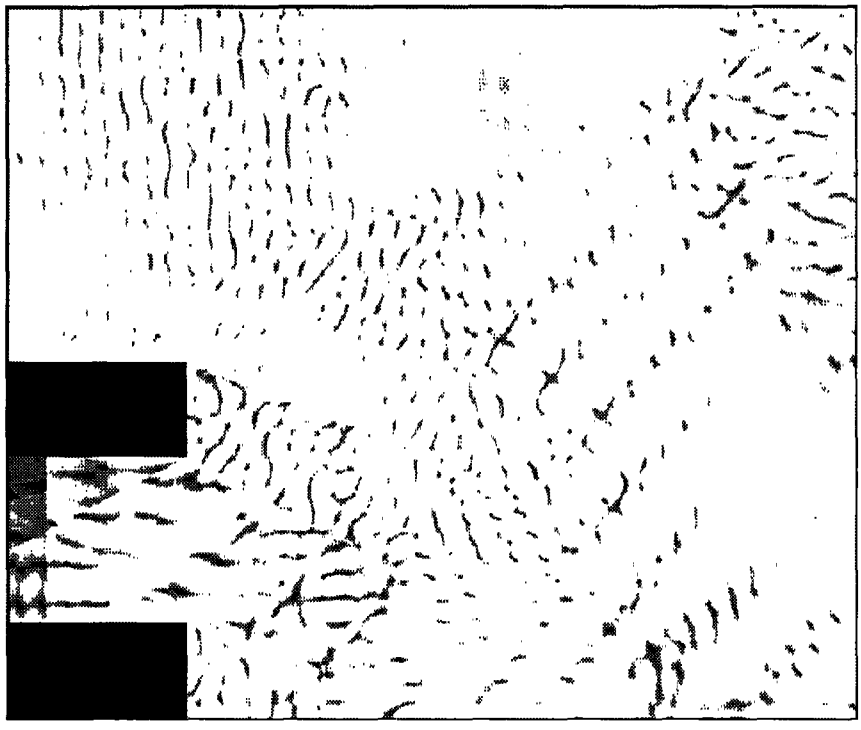


Figure 4—A representative signal intensity map

Further description difficulties arise when attempting to describe collocated coverage areas. Consider Figure 5, in which STA 6 could belong to BSS 2 or BSS 3.

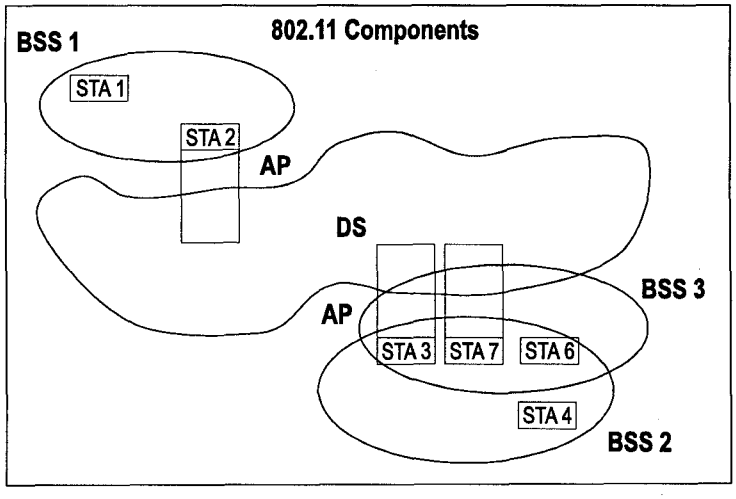


Figure 5—Collocated coverage areas

While the concept of sets of stations is correct, it is often convenient to talk about areas. For many topics the concept of area is sufficient. *Volume* is a more precise term than area, though still not technically correct. For historical reasons and convenience, this standard uses the common term *area*.

5.2.4 Integration with wired LANs

To integrate the IEEE 802.11 architecture with a traditional wired LAN, a final *logical* architectural component is introduced—a *portal.*

A portal is the logical point at which MSDUs from an integrated non-IEEE 802.11 LAN enter the IEEE 802.11 DS. For example, a portal is shown in Figure 6 connecting to a wired 802 LAN.

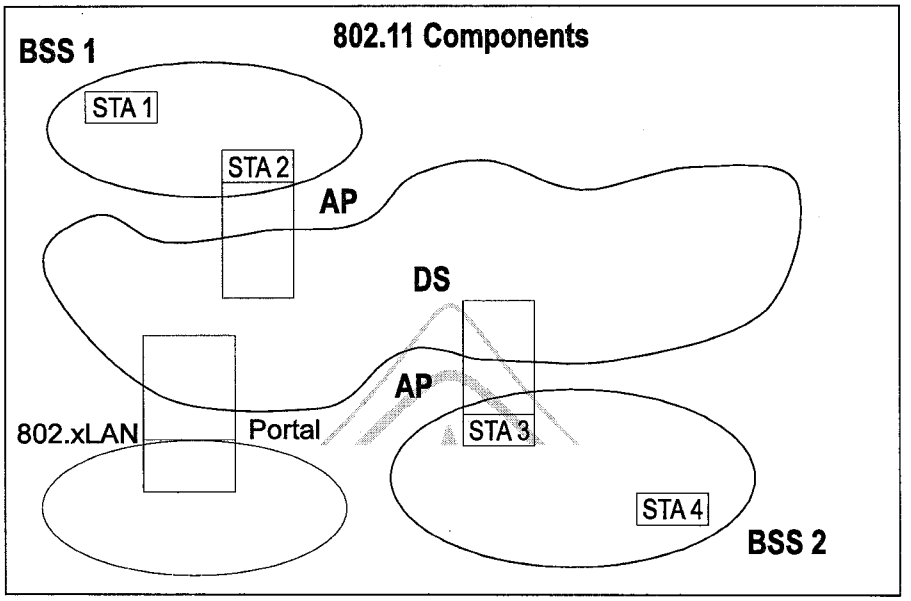


Figure 6—Connecting to other IEEE 802 LANs

All data from non-IEEE 802.11 LANs enter the 802.11 architecture via a portal. The portal provides logical integration between the IEEE 802.11 architecture and existing wired LANs. It is possible for one device to offer both the functions of an AP and a portal; this could be the case when a DS is implemented from 802 LAN components.

In IEEE 802.11, the ESS architecture (APs and the DS) provides traffic segmentation and range extension. Logical connections between IEEE 802.11 and other LANs are via the portal. Portals connect between the DSM and the LAN medium that is to be integrated.

5.3 Logical service interfaces

The IEEE 802.11 architecture allows for the possibility that the DS may not be identical to an existing wired LAN. A DS may be created from many different technologies including current 802 wired LANs. IEEE 802.11 does not constrain the DS to be either data link or network layer based. Nor does IEEE 802.11 constrain a DS to be either centralized or distributed in nature.

IEEE 802.11 explicitly does not specify the details of DS implementations. Instead, IEEE 802.11 specifies *services.* The services are associated with different components of the architecture. There are two categories of IEEE 802.11 service—the station service (SS) and the distribution system service (DSS). Both categories of service are used by the IEEE 802.11 MAC sublayer.

The complete set of IEEE 802.11 architectural services are as follows:

a) Authentication
b) Association
c) Deauthentication
d) Disassociation
e) Distribution
f) Integration
g) Privacy
h) Reassociation
i) MSDU delivery

This set of services is divided into two groups: those that are part of every station, and those that are part of a DS.

5.3.1 Station service (SS)

The service provided by stations is known as the *station service*.

The SS is present in every IEEE 802.11 station (including APs, as APs include station functionality). The SS is specified for use by MAC sublayer entities. All conformant stations provide SS.

The SS is as follows:

a) Authentication
b) Deauthentication
c) Privacy
d) MSDU delivery

5.3.2 Distribution system service (DSS)

The service provided by the DS is known as the *distribution system service*.

These services are represented in the IEEE 802.11 architecture by arrows within the APs, indicating that the services are used to cross media and address space logical boundaries. This is the convenient place to show the services in the picture. The physical embodiment of various services may or may not be within a physical AP.

The DSSs are provided by the DS. They are accessed via a STA that also provides DSSs. A STA that is providing access to DSS is an AP.

The DSSs are as follows:

a) Association
b) Disassociation
c) Distribution
d) Integration
e) Reassociation

DSSs are specified for use by MAC sublayer entities.

Figure 7 combines the components from previous figures with both types of services to show the complete IEEE 802.11 architecture.

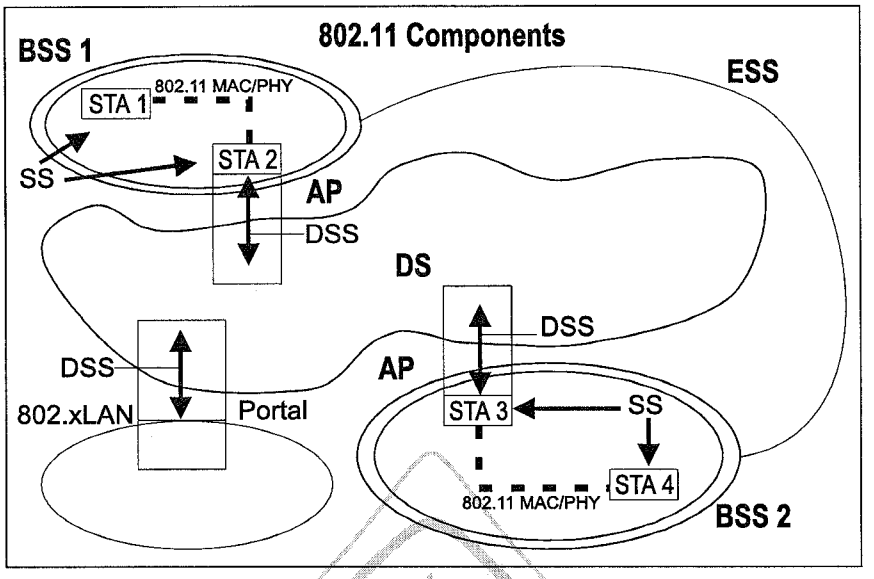


Figure 7—Complete IEEE 802.11 architecture

5.3.3 Multiple logical address spaces

Just as the IEEE 802.11 architecture allows for the possibility that the WM, DSM, and an integrated wired LAN may all be different physical media, it also allows for the possibility that each of these components may be operating within different address spaces. IEEE 802.11 only uses and specifies the use of the WM address space.

Each IEEE 802.11 PHY operates in a single medium—the WM. The IEEE 802.11 MAC operates in a single address space. MAC addresses are used on the WM in the IEEE 802.11 architecture. Therefore, it is unnecessary for the standard to explicitly specify that its addresses are "WM addresses." This is assumed throughout the IEEE 802.11 standard.

IEEE 802.11 has chosen to use the IEEE 802 48-bit address space (see 7.1.3.3.1). Thus IEEE 802.11 addresses are compatible with the address space used by the 802 LAN family.

The IEEE 802.11 choice of address space implies that for many instantiations of the IEEE 802.11 architecture, the wired LAN MAC address space and the IEEE 802.11 MAC address space may be the same. In those situations where a DS that uses MAC level 802 addressing is appropriate, all three of the logical address spaces used within a system could be identical. While this is a common case, it is not the only combination allowed by the architecture. The IEEE 802.11 architecture allows for all three logical address spaces to be distinct.

A multiple address space example is one where the DS implementation uses network layer addressing. In this case, the WM address space and the DS address space would be different.

The ability of the architecture to handle multiple logical media and address spaces is key to the ability of IEEE 802.11 to be independent of the DS implementation and to interface cleanly with network layer mobility approaches. The implementation of the DS is unspecified and is beyond the scope of this standard.

1 GENERAL DESCRIPTION

Bluetooth wireless technology is a short-range communications system intended to replace the cable(s) connecting portable and/or fixed electronic devices. Key features are robustness, low power, and low cost. Many features of the core specification are optional, allowing product differentiation.

The Bluetooth core system consists of an RF transceiver, baseband, and protocol stack. The system offers services that enable the connection of devices and the exchange of a variety of classes of data between these devices.

This chapter of the specification provides an overview of the Bluetooth system architecture, communication topologies and data transport features. The text in this chapter of the specification should be treated as informational and used as a background and for context-setting.

1.1 OVERVIEW OF OPERATION

The Bluetooth RF (physical layer) operates in the unlicensed ISM band at 2.4 GHz. The system employs a frequency hop transceiver to combat interference and fading and provides many FHSS carriers. RF operation uses a shaped, binary FM modulation to minimize transceiver complexity. The symbol rate is 1 Megasymbol per second (Ms/s) supporting the bit rate of 1 Megabit per second (Mb/s).

During typical operation a physical radio channel is shared by a group of devices that are synchronized to a common clock and frequency hopping pattern. One device provides the synchronization reference and is known as the master. All other devices are known as slaves. A group of devices synchronized in this fashion form a piconet. This is the fundamental form of communication in the Bluetooth wireless technology.

Devices in a piconet use a specific frequency hopping pattern, which is algorithmically determined by certain fields in the Bluetooth address and clock of the master. The basic hopping pattern is a pseudo-random ordering of the 79 frequencies in the ISM band. The hopping pattern may be adapted to exclude a portion of the frequencies that are used by interferring devices. The adaptive hopping technique improves Bluetooth co-existence with static (non-hopping) ISM systems when these are co-located.

The physical channel is sub-divided into time units known as slots. Data is transmitted between Bluetooth devices in packets, that are positioned in these slots. When circumstances permit, a number of consecutive slots may be allocated to a single packet. Frequency hopping takes place between the transmission or reception of packets. Bluetooth technology provides the effect of full duplex transmission through the use of a Time-Division Duplex (TDD) scheme.

Above the physical channel there is a layering of links and channels and associated control protocols. The hierarchy of channels and links from the physical channel upwards is physical channel, physical link, logical transport, logical link and L2CAP channel. These are discussed in more detail in Section 3.3 on page 32 - Section 3.6 on page 48 but are introduced here to aid the understanding of the remainder of this section.

Within a physical channel, a physical link is formed between any two devices that transmit packets in either direction between them. In a piconet physical channel there are restrictions on which devices may form a physical link. There is a physical link between each slave and the master. Physical links are not formed directly between the slaves in a piconet.

The physical link is used as a transport for one or more logical links that support unicast synchronous, asynchronous and isochronous traffic, and broadcast traffic. Traffic on logical links is multiplexed onto the physical link by occupying slots assigned by a scheduling function in the resource manager.

A control protocol for the baseband and physical layers is carried over logical links in addition to user data. This is the link manager protocol (LMP). Devices that are active in a piconet have a default asynchronous connection-oriented logical transport that is used to transport the LMP protocol signalling. For historical reasons this is known as the ACL logical transport. The default ACL logical transport is the one that is created whenever a device joins a piconet. Additional logical transports may be created to transport synchronous data streams when this is required.

The Link Manager function uses LMP to control the operation of devices in the piconet and provide services to manage the lower architectural layers (radio layer and baseband layer). The LMP protocol is only carried on the default ACL logical transport and the default broadcast logical transport.

Above the baseband layer the L2CAP layer provides a channel-based abstraction to applications and services. It carries out segmentation and reassembly of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. L2CAP has a protocol control channel that is carried over the default ACL logical transport. Application data submitted to the L2CAP protocol may be carried on any logical link that supports the L2CAP protocol.

2 CORE SYSTEM ARCHITECTURE

The Bluetooth core system covers the four lowest layers and associated protocols defined by the Bluetooth specification as well as one common service layer protocol, the Service Discovery Protocol (SDP) and the overall profile requirements are specified in the Generic Access Profile (GAP). A complete Bluetooth application requires a number of additional service and higher layer protocols that are defined in the Bluetooth specification, but are not described here. The core system architecture is shown in Figure 2.1 on page 19 except for SDP that is not shown for clarity.

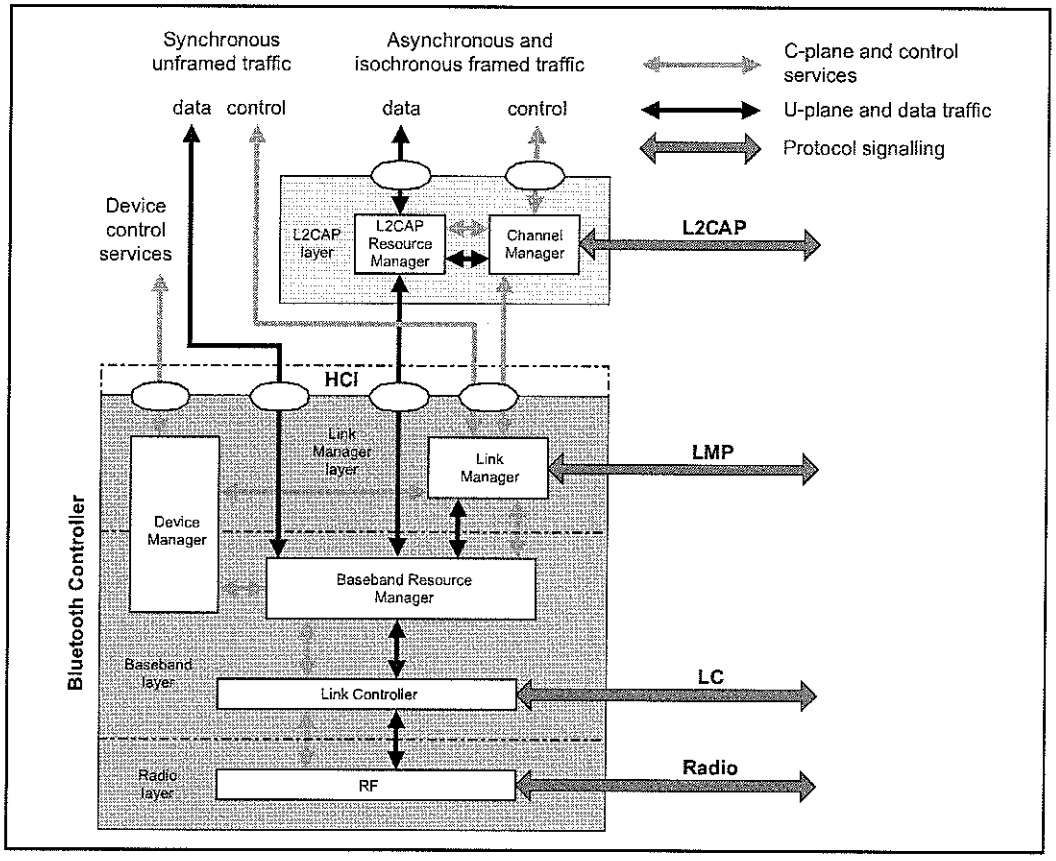


Figure 2.1: Bluetooth core system architecture

Figure 2.1 on page 19 shows the four lowest layers, each with its associated communication protocol. The lowest three layers are sometimes grouped into a subsystem known as the Bluetooth controller. This is a common implementation involving a standard physical communications interface between the Bluetooth controller and remainder of the Bluetooth system including the L2CAP, service and higher layers (known as the Bluetooth host.) Although this interface is optional the architecture is designed to allow for its existence and characteristics. The Bluetooth specification enables inter-operability between independent Bluetooth systems by defining the protocol messages exchanged between equivalent layers, and also inter-operability between independent

Bluetooth sub-systems by defining a common interface between Bluetooth controllers and Bluetooth hosts.

A number of functional blocks are shown and the path of services and data between these. The functional blocks shown in the diagram are informative; in general the Bluetooth specification does not define the details of implementations except where this is required for inter-operability. Thus the functional blocks in Figure 2.1 on page 19 are shown in order to aid description of the system behaviour. An implementation may be different from the system shown in Figure 2.1 on page 19.

Standard interactions are defined for all inter-device operation, where Bluetooth devices exchange protocol signalling according to the Bluetooth specification. The Bluetooth core system protocols are the Radio (RF) protocol, Link Control (LC) protocol, Link Manager (LM) protocol and Logical Link Control and Adaptation protocol (L2CAP), all of that are fully defined in subsequent parts of the Bluetooth specification. In addition the Service Discovery Protocol (SDP) is a service layer protocol required by all Bluetooth applications.

The Bluetooth core system offers services through a number of service access points that are shown in the diagram as ellipses. These services consist of the basic primitives that control the Bluetooth core system. The services can be split into three types. There are device control services that modify the behaviour and modes of a Bluetooth device, transport control services that create, modify and release traffic bearers (channels and links), and data services that are used to submit data for transmission over traffic bearers. It is common to consider the first two as belonging to the C-plane and the last as belonging to the U-plane.

A service interface to the Bluetooth controller sub-system is defined such that the Bluetooth controller may be considered a standard part. In this configuration the Bluetooth controller operates the lowest three layers and the L2CAP layer is contained with the rest of the Bluetooth application in a host system. The standard interface is called the Host to Controller Interface (HCI) and its service access points are represented by the ellipses on the upper edge of the Bluetooth controller sub-system in Figure 2.1 on page 19. Implementation of this standard service interface is optional.

As the Bluetooth architecture is defined with the possibility of separate host and controller communicating through an HCI, a number of general assumptions are made. The Bluetooth controller is assumed to have limited data buffering capabilities in comparison with the host. Therefore the L2CAP layer is expected to carry out some simple resource management when submitting L2CAP PDUs to the controller for transport to a peer device. This includes segmentation of L2CAP SDUs into more manageable PDUs and then the fragmentation of PDUs into start and continuation packets of a size suitable for the controller buffers, and management of the use of controller buffers to ensure availability for channels with Quality of Service (QoS) commitments.

The Baseband layer provides the basic ARQ protocol in Bluetooth. The L2CAP layer can optionally provide a further error detection and retransmission to the L2CAP PDUs. This feature is recommended for applications with requirements for a low probability of undetected errors in the user data. A further optional feature of L2CAP is a window-based flow control that can be used to manage buffer allocation in the receiving device. Both of these optional features augment the QoS performance in certain scenarios.

Although these assumptions may not be required for embedded Bluetooth implementations that combine all layers in a single system, the general architectural and QoS models are defined with these assumptions in mind, in effect a lowest common denominator.

Automated conformance testing of implementations of the Bluetooth core system is required. This is achieved by allowing the tester to control the implementation through the RF interface, which is common to all Bluetooth systems, and through the Test Control Interface (TCI), which is only required for conformance testing.

The tester uses exchanges with the Implementation Under Test (IUT) through the RF interface to ensure the correct responses to requests from remote devices. The tester controls the IUT through the TCI to cause the IUT to originate exchanges through the RF interface so that these can also be verified as conformant.

The TCI uses a different command-set (service interface) for the testing of each architectural layer and protocol. A subset of the HCI command-set is used as the TCI service interface for each of the layers and protocols within the Bluetooth Controller subsystem. A separate service interface is used for testing the L2CAP layer and protocol. As an L2CAP service interface is not defined in the Bluetooth core specification it is defined separately in the Test Control Interface specification. Implementation of the L2CAP service interface is only required for conformance testing.