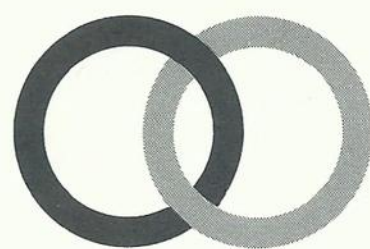

Serial Bus Structures For Microcomputers — Small Area Networks



Wescon/83

Electronic Show and Convention
November 8-11/San Francisco, California

Session 26

Serial Bus Structures for Microcomputers - Small
Area Networks

Session Organizer
Alex Goldberger
Exel Microelectronics
Milpitas, CA

Session Chairman
Cecil Kaplinsky
Signetics Corporation
Sunnyvale, CA

26/1 An Introduction to Small Area Networks

Cecil Kaplinsky
Signetics Corporation
Sunnyvale, CA

26/2 The D²B - A Digital Data Bus for Small Area
Networks

Charles Seaborg
Signetics Corporation
Sunnyvale, CA

26/3 Using the 8051's 9-Bit Serial Mode

Alan Steinberg and Ira Horden
Intel Corporation
Chandler, AZ

26/4 The I²C Bus - An Interconnect Structure for
Integrated Circuits

Ir. A.P.M.M. Moelands
Philips International B.V.
Eindhoven, The Netherlands

Archief
Ir. A.P.M.Moelands

AN INTRODUCTION TO SMALL AREA NETWORKS

Cecil Kaplinsky, Manager of Microprocessor Architecture
Signetics Corporation
811 E. Arques Ave.
Sunnyvale, CA 94086

INTRODUCTION

This paper is necessarily an introduction not a review, since Small Area Networks are just emerging into the marketplace. The concept of Small Area Networks has been in existence for some time, but the name itself is new and needs to be defined in a way that allows the topic to be discussed without ambiguity.

Before Small Area Networks can be distinguished from other types of area networks, it is necessary to define the term 'Area Network'. This definition is followed by a brief discussion of the relatively familiar Local Area Networks. Small Area Networks (SANs) will then be differentiated from LANs and their properties and applications will be reviewed.

AREA NETWORKS

Although the term 'Area Networks' has been defined implicitly rather than explicitly, it is possible to extract the essential features from the implicit definitions. Area networks work over a distance and although they could be implemented as a parallel bundle of wires carrying byte or word information, it is generally accepted that the only realistic transmission methods are those that carry the information serially. This means that all the information, both data and timing signals, is carried on one or two transmission channels and that Network Interface Units are needed to synchronize, serialize and de-serialize the information appropriately.

Networks can exist with different topologies, e.g. buses, rings, trees, etc. The definitions used here apply without differentiation to these different methods of connection.

Aside from their serial transmission of information, Area Networks are distinguished by two logical properties and one legal property. The logical properties are the most important for networks.

Area networks are modular; nodes can be added without disturbing existing units. The only way that a new node should affect an existing node is to reduce the bandwidth available to the latter for transmission of data, i.e. the total bandwidth is shared by an extra

node. Thus new nodes, that communicate only amongst themselves, can be attached to the communication channel without the knowledge of the pre-existing nodes and without interfering with their use.

The second property is that Area Networks are democratic. They allow all units to participate, on an equal basis, in the decision as to which node will gain control of the transmission medium if one or more nodes wish to use the network. However, this decision can be made in many ways and may not be fair, i.e. it may not give every node an equal chance to acquire the network. This democratic decision making process implies that there is no central or master unit which controls the network.

The legal property that defines a network is its ownership. Unlike telecom systems, area networks are owned and managed by their users and are not governed by the restrictions that are faced by public utilities. As a result they can be optimized to suit the needs of their users.

Area Networks, then, are modular serial communication channels which allocate the available bandwidth in a democratic fashion and are owned, or at least managed, by their users. This is to be contrasted with the traditional, serial data communication networks which employ autocratic control, using public utilities and which need an alteration, to at least the autocratic master, the exchange, when new units are added.

LOCAL AREA NETWORKS

Current definitions (1,2) of Local Area Networks tend to define the parameters like network length and speed rather than the logical properties of the network. This form of definition can lead to "religious wars" over what does or does not constitute a Local Area Network. In this paper we shall try and define a LAN logically rather than by its attributes, i.e. a Local Area Network is a modular, serial network, managed by its users which allocates its use in a democratic fashion and transmits its data in a packet.

The term packet is slightly misleading because it refers as much to the quantum mechanical concept of a wave packet as it does to the normal data communication concept of a

packet. The information on a LAN is spread out. Since electrical signals travel at a finite speed and since the network is relatively long compared to the wavelength of the signal, different nodes on the LAN 'see' different bits at the same time.

The principal is illustrated in Figure 1 which shows a bus; however the principal applies to all topologies. Node A is currently transmitting bit_n of a packet. While A is transmitting this bit, node B is receiving bit_r and node C is receiving bit₁. The other nodes have not yet noticed any activity on the network. The fact that a LAN transmits its data in a packet implies that the data rate is high. In practice the data rates tend to be in the range of 1-10 Mbits/sec.

The acknowledgement that these packets have been received, and understood, is not given synchronously with the sending of a packet and is not always given. When the arrival of a packet is acknowledged, this is done at a later time with a reference to the packet which is being acknowledged.

Currently LANs are used in EDP and process control systems and there is a trend towards extending them into the area of human communication, e.g. telephony. Data is transported in logical frames of data from one node to another. The protocol for transmitting frames is at a higher level than the level which controls the passing of the bitstream from one node to another, e.g. the PUP protocol⁽³⁾.

The implementation of LAN interface units is complex and relatively expensive employing a sophisticated VLSI I.C., which has a DMA interface into a memory and needs a processor to control the transfers, e.g. Ethernet⁽⁴⁾. The processor is needed since some form of intelligence is required for decoding messages and issuing responses.

Many local area networks have been defined and implemented. One of the first being the Ethernet⁽²⁾. Others exist and three of the most important LANs are currently being standardized by the IEEE⁽⁵⁾. All of these LANs have a data transmission rate of between 1-10 Mbits/sec, a length of between 500M and 2kM and can have several hundred attached nodes.

SMALL AREA NETWORKS

A Small Area Network has many of the same logical characteristics as a Local Area Network but differs in one essential aspect. It will be shown that this one aspect has a marked affect on the attributes and application areas of a SAN.

A Small Area Network is a modular, serial network, managed by its users, with an embedded protocol and which allocates its use in a democratic fashion and transmits its data bit by bit. The one big difference in the definition of a Small Area Network is the fact that bits are transmitted so that all nodes see the same bit simultaneously.

This apparently small difference has a profound effect on the properties of the network. The most apparent is that there is a more immediate correlation between the length of the network and the speed of transmission. The latter is constrained by the fact that, after a bit has been transmitted, the transmitter has to wait for it to propagate along the full length of the medium before it can start transmitting the next bit. This restriction has some positive aspects. The first and most important of which is that messages can be acknowledged synchronously. As all modules see the same bit simultaneously, the receiving unit can, if the protocol demands it, send an acknowledge signal as soon as it has received the message. This simplifies the logic needed to implement the interface unit.

This synchronous property also makes the second difference between SANs and LANs possible. Because a receiving node can respond synchronously to the transmitter, the protocol can be embedded in the network. If responses are asynchronous, then it is not productive to define the protocol on the medium as all transmitters and receivers need to have memory and logic to remember what has been acknowledged and what is faulty. However, synchronous response protocols do not need these mechanisms.

The fact that a bit is seen by all nodes simultaneously also increases the reliability of the network, as the transmitter is able to check that the correct value is on the medium at the time that it is seen by the receiver.

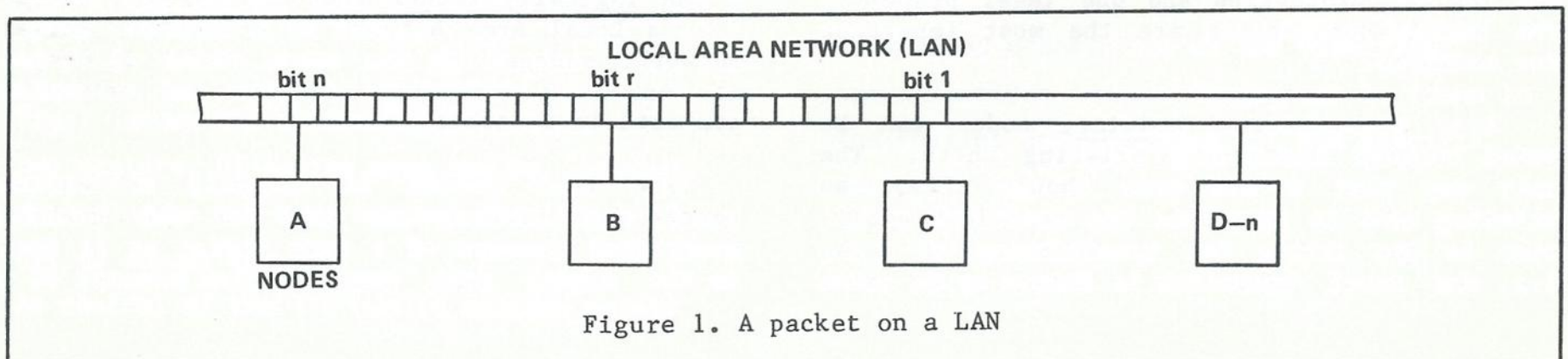


Figure 1. A packet on a LAN

These properties, thus favor the use of small messages over relatively short distances. The overhead of any message is large, since it normally contains timing information to synchronize the nodes, address information and checking information. Thus, if a message is used to acknowledge receipt of a previous short message, the effective overhead per message is doubled. On the other hand, if a bit is appended to a message to acknowledge receipt, then the effective overhead per message bit is smaller and there is a smaller penalty for short messages.

The embedded protocol can do more than carry a response. The messages can carry information marking them, for instance, as network messages, control messages or data messages. The receiving unit can respond, in a synchronous manner, with an acknowledgement of its ability or inability to accept or carry out the requested function.

Small Area Networks are thus used not only for the transmission of data but often carry information to control a system or part of a system. Control information tends to be short, less than 5 bytes, and requires an acknowledgement of safe delivery. This control can be done economically because SANs tend to have small packets and have a simple protocol that can be implemented in hardware. In some cases the protocol is so simple that it does not need a separate integrated circuit to implement it but can be implemented on a portion of an integrated circuit. Examples of these will be discussed in this session^(6,7).

APPLICATIONS OF SANs

SANs are used to interface electronic systems to humans and electromechanical devices, rather than to other electronic systems. Naturally, electronics are used to actually implement the interface. Consider the case of EDP systems; LANs can be used to interface two system modules which need to switch large amounts of data, e.g. files, quickly between themselves. However, it is unlikely that a LAN will ever be a cost effective solution to connecting a keyboard to a terminal or even a terminal to a CPU. Thus, one can conceive of a hierarchy of networks in the EDP field with a SAN tying together local units, e.g. printer, keyboard, display and other peripherals. This cluster, or perhaps a super cluster of these units, could be attached to the mainframe with a LAN.

Similar examples can be found in the process control field. A work area can be connected to a central controller using a LAN, e.g. a machine which will paint a car to a production control computer. Typically, the production control computer will announce the arrival of the car at the paint station, indicate its type, and what color or colors it should be painted. Within the paint station the connection from its own controller to all the nozzles could be performed most economically by a SAN. The controller would switch the nozzles on or off and control the nozzle trajectory in real time.

A myriad of examples can be found in personal articles both in and outside the home. Clearly either the all the systems in the home, or a portion of it such as the audio-visual system, can be connected together. This connection could be for reasons of control (e.g. a security system) or for ease of use. An example of this is a remote control used to control all the home entertainment units. Thus, if the VCR was turned on, the TV set could be automatically turned on, tuned to the VCR channel, and the speakers could be switched from the record player to the VCR.

CHOOSING A SAN

In assessing the different SANs it is important to consider some of the important parameters. Three obvious parameters are:

- (a) the length of the SAN,
- (b) the speed of transmission, and
- (c) the number of units that can be connected to it.

The need for speed is not an absolute; it depends on the use of the SAN. Different applications need different speeds. If the SAN is only to be used to transmit control signals from a person to a unit, e.g. on/off/change channel etc., then it need only operate at human speeds, i.e. 10-100 chars/sec. However if it is to be used to control an electromechanical device, or provide it with data, then it needs to be faster. Typical electromechanical devices, e.g. printers or motors, normally do not need control signals or data at a rate faster than 100-1000 chars/sec. Interfaces to humans (e.g. visual display units or speed synthesizers) and between humans (e.g. an intercom) seldom need a faster data rate than 8000 chars/sec.

The distance that a SAN must span can vary. It can vary from within a machine (1-10ft) to within a house or work-area (300-500ft). The choice of distance obviously varies with the application.

Other considerations are important. The electrical environment can have an effect on the error rate of the network, which in itself is an important characteristic. Clearly if the network is to work in a home or in a car it cannot be allowed to interfere with existing units like radios by radiating strong signals in the relevant wavelengths.

The final and perhaps most important characteristic in considering a Small Area Network is the price. This includes not only the cost of the logic but also the cost of the connectors and cables. If the cost of a connection is to be kept to a reasonable figure, say <2-3% of the module cost, then these costs will have to be kept below a few dollars if SANs are to be fully exploited.

CONCLUSION

Clearly the whole area of SANs is ripe for standardization. Perhaps the best initial filter for the possibilities currently being proposed is the market place. However, somebody needs to start considering the SANs that survive this test and provide standards so that different manufacturers can design compatible equipment.

This introduction to SANs has shown how they differ from local area networks and, in particular the way in which they offer a lower performance at a much lower cost. Small Area

Networks can be exploited in systems which connect humans and electromechanical devices together as opposed to LANs which connect electronic systems together at a higher cost.

1. Kenneth J. Thurber and Harvey A. Freeman, "Architecture Considerations for Local Computer Networks," Proc. First Int'l Conf. Distributed Computing Systems, Oct 1979, pp-131-142.

2. John F. Shoch, Yogen K. Dalal and David D. Redell, "Evolution of the Ethernet Local Computer Network", Computer, Vol 15, No. 8, August 1982, 10-27.

3. David R. Boggs, John F. Shoch, Edward A. Taft and Robert M. Metcalfe, "Pup: An Internetwork Architecture", IEEE Trans. on Communications, Vol COM-28, No. 4, April 1980, 612-624.

4. John Ketzner, Kelly Pan, Bob Beach and Darcy Nelson, "Communicating Computer Simplifies the System Integrator's Work," Electronics, Vol 56, No. 17, August 25, 1983, 121-126.

5. Ware Myers, "Towards a Local Network Standard," IEEE Micro, August 1982, 28-45.

6. Charles Seaborg, "The D²B - A Digital Data Bus for Small Area Networks," to be published in Proceedings of WESCON, November 1983.

7. A.P.P. Moelands, "I²C Bus - An Interconnect Structure for Integrated Circuits," to be published in Proceedings of WESCON, November 1983.

THE D²B - A DIGITAL DATA BUS FOR SMALL AREA NETWORKS

Charles Seaborg, Design Manager
Signetics Corp.
811 E. Arques Ave.
Sunnyvale, Ca. 94086

INTRODUCTION

The Digital Data Bus (D²B) is a small area network (SAN). It is designed to fill the gap between the relatively high performance of a local area network such as the Ethernet and the relative simplicity of a serial data link such as the RS-232 protocol. The D²B is intended to be used for multimaster control and communications where very high data rates are not necessary. Its protocol insures a high data integrity level through encoding of parity information and receiver acknowledges at multiple places within a complete transmission frame. In addition, the physical or electrical protocol layer has also been specified to maintain high levels of data integrity. As the name small area network implies, the D²B is limited to a maximum of 50 masters and slaves and a maximum length of 150 meters. The D²B is being designed onboard an 80C51 CMOS microcomputer.

PROTOCOL STANDARDIZATION

In order to simplify and standardize the D²B network, both the electrical layer and the control layer of protocol have been incorporated into the D²B definition. The control protocol layer defines the address, control, and data fields of a message, which the user sees, as well as invisible things such as start/stop bits, acknowledge and parity bit fields.

Predefining the control layer unburdens the user from the task of defining his own hardware control protocol, something he may not want to do or be able to do efficiently. The electrical layer has been defined so that supporting or 'glue' chips are not required for the bus interface to the microcomputer. The bus uses low voltage analog levels to minimize RF radiation from the twisted wire pair and differential sensing to reduce stray noise coupling onto the bus.

The bus drivers and receivers are shown in figure 1. The bus is terminated at each end by 125 ohm resistors. The receiver is a CMOS analog comparator built on the same chip with the D²B. The comparator functions in the differential sensing mode to help reject any noise coupled onto the twisted pair. The differential levels sensed are approximately 100 millivolts.

MODULARITY

The D²B is expandable in a modular fashion. The D²B does not require a central controlling master, so the addition of another unit to the bus is done just by physically connecting it onto the bus. For example, take a situation where the bus is performing some given set of functions A. By adding units to the bus it is possible to overlay a completely different set of functions B onto the original bus. The new units need not know anything about the original units. By adopting the concept of a multimaster, autonomus bus and discarding the central controlling master concept, the D²B allows simple cost effective modular expandability. A feature that results from the use of a multimaster network is increased reliability. No unit(s) can cause the entire network to fail. In the above example function A continues operating independent of the status of function B.

BUS ARBITRATION

The multimaster nature of the D²B requires an algorithm to decide, when two masters both try to use the bus simultaneously, which shall win bus arbitration. In keeping with the modular expandability of the D²B, the bus arbitration scheme used is distributed thruout the system. Every potential master unit incorporates into its control protocol the necessary arbiting ability.

The basic properties of the bus arbitration rely on two essential needs of the bus. The first is the bus must be capable of a wired 'OR' connection and the second is the need for all units on the bus to see the same information at the same point in time. For correct arbitration, all units must be given enough time to let the bus arbitration procedures propagate to every unit before any decisions are made. This is one of the essential properties of a SAN. This simultaneity puts a constraint on the speed versus length trade-off. It is possible to define higher speeds on the bus if the length is shortened. A benefit of the simultaneity constraint is a simplification of the arbitration procedure. See figure 1 for the bus interconnection.

When all drivers are off, the bus is defined to be at a logical '1'. If any or all drivers are on, the bus goes to a logical '0'.

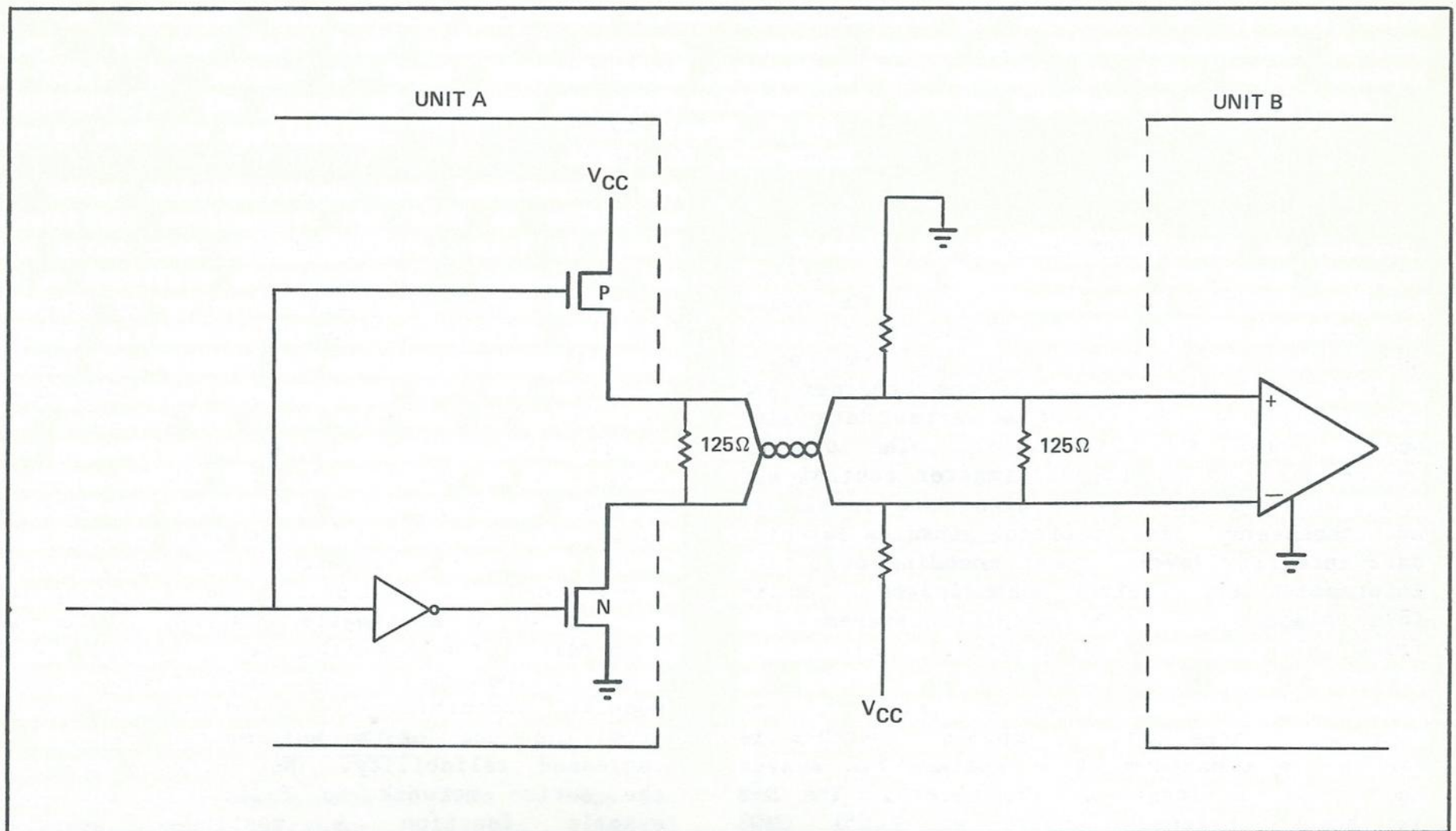


Figure 1. D²B Driver and Receiver Electrical Configuration

Since all transmitters also monitor the bus during arbitration, the units trying to send a '1' (passive drive) can determine if an active unit is overriding the '1' with a '0' at the same time. Through this property of a wired 'OR' structure, a simple arbitration scheme can be defined.

TRANSMISSION SPEEDS

In order to meet cost objectives for the bus, a simple low speed bus, mode 0, was defined to transmit simple data or control information to other units on the bus. Mode 0 will transfer about 110 bytes per second over the bus. Units that send and receive in mode 0 need only have a frequency tolerance of 25%, suitable for RC control. Mode 0 is also slow enough to allow an output port on an 80C48 or an 80C51 to be controlled by software and communicate on the bus through appropriate bus drivers and receivers.

There are also two other speeds or modes defined for the D²B. Mode 1 is a faster speed mode, 2.4K bytes per second, which requires 25%

clock tolerances. Mode 2 the fastest mode defined, can transfer 8.8K bytes per second and requires crystal control of .5% tolerance. Other higher speed modes can be defined in the future and still insure compatibility with current units that would not have the higher speed mode. All units are required to be able to communicate at all modes slower than their highest operable mode. Therefore, the units containing mode 3 or higher could talk to each other but would slow down to converse with slower units.

D²B MESSAGE FORMAT

A complete D²B message frame is shown in figure 2. A message frame is defined as one complete transaction between two units on the bus. A frame has a maximum length of time in the bus of approximately 9ms. In mode 0, 9ms allows one byte of data to be sent per frame, in mode 1, 22 bytes, and in mode 2, 75 bytes. Higher modes of operation shorten the overhead time of the protocol and allow more bytes of information to be transmitted within the 9ms maximum frame time.

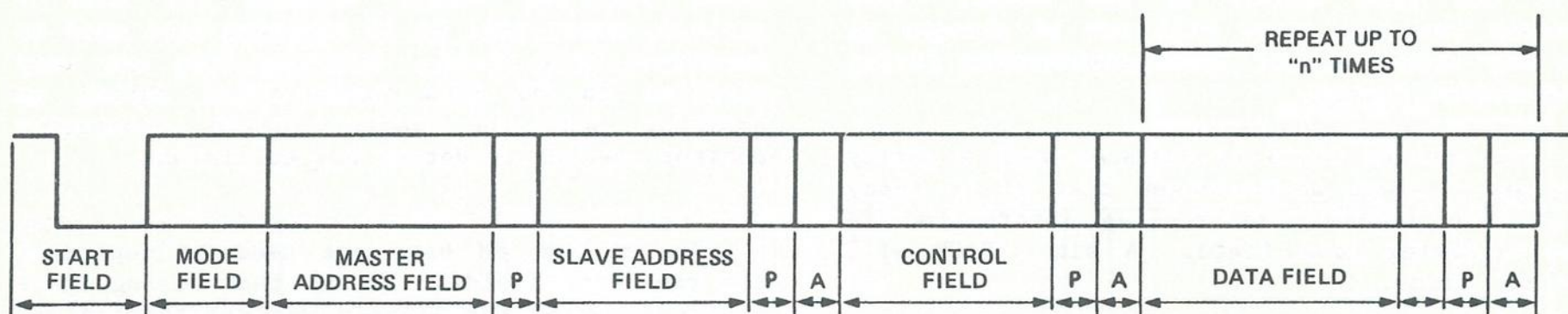


Figure 2. D²B Fields Within One Frame

The D²B units can be both master or slave devices, and transmitter or receiver devices at the same time. A master unit is one which has won arbitration on the bus and which controls bus timing or synchronization from then on through completion of the frame. A slave device is any unit which was not arbiting for the bus or a master unit which lost bus arbitration. A slave device does not control bus timing. It is required to synchronize itself to the bus master.

Masters and slaves can also be transmitters and receivers at certain times within a complete frame. A transmitter is that unit, master or slave, which is actively driving the bus sending address, control, or data but not synchronization information to the receiving unit. It is possible that a slave device can be requested by the master to send or transmit information back to the master. In this case the slave is the transmitter of the data and the master is the receiver although it still provides bus synchronization timing for the slave.

There are six basic fields which comprise one frame. They are in order of transmission: the start field, mode field, master address field, slave address field, control field, and the data field. Each frame always contains these six basic fields.

The start field always begins a new frame. The start field has two parts. During the first part, a master wishing to initiate a new frame must enter the start field in a passive mode listening for other transmitters on the bus. If transmissions are detected, the master loops in the listening mode until the bus is quiet. When the master detects the bus has been quiet for a certain time, the second half of the start field is entered where the master drives the bus for a unique period of time which is recognized by all units on the bus as a start field.

When a master begins a frame in the start field, the lowest frequency, mode 0, is assumed. In order to change to mode 2, the master must wait until entering the mode field, then the mode can be stepped up incrementally until the desired mode is reached. In order to go from mode 0 to mode 2, a master steps through mode 1 for one bit time. This transitory state is necessary for bus arbitration to function predictably. At the end of the mode field, the transmission speed for all units is defined for the rest of the frame.

Next is the master address field. The master outputs its address onto the bus, MSB first, thereby notifying all receivers it is transmitting as master for this frame. The master address is 12 bits long and is followed by a parity bit.

Now that the master has identified itself to all units on the bus, it enters into the slave address field. The 12-bit slave address is sent MSB first, followed by a parity bit. Concluding this field is a transmission from the addressed slave acknowledging it is on line and is correctly functioning. If the slave was not on line, the master would receive a negative acknowledge since the passive response on the bus is a NACK. A NACK would terminate the frame for the master, freeing up the bus for other units.

The purpose of the transmission is defined in the control field. It tells the slave whether it will be a transmitter or receiver in the following data field, and what kind of information will be sent at that time. Typically this information will be status, commands, address or data. A slave may also be dedicated or locked to a particular master using the control field. Locking is useful when a large amount of information must be sent and cannot all be sent in one frame. By locking the slave, a master insures that no other

unit can gain access to the slave. This insures that the slave will not be changed in any way if the master loses arbitration during consecutive frames.

The control field is finished with a parity bit and a slave acknowledges that it is ready either to transmit or receive information in the following data field. A slave NACK will end the frame.

The data field is used to transmit the information requested in the control field. Eight bits at a time are sent followed by an end of data bit which is set following the last data byte to be sent. Following the EOD bit is a parity bit and an acknowledge bit from the receiver that the byte was correctly received. If the receiver NACKS the previous byte of data, the transmitter will resend the byte until either; 1) the receiver acknowledges the byte, or 2) the maximum frame time of 9ms is reached, at which time the master ceases trying to send data to the slave and a bus error condition is generated.

Thruout the mode and master address fields, bus arbitration is also occurring. All masters trying to control the bus must also monitor the bus during these fields. If a certain master sends a '1' onto the bus during one of these fields (i.e. passive drive) and sees a '0' (someone else actively driving the bus), then that master will cease arbiting for the bus. Since all units on the bus must have a unique address, by the time the master address field is completed only one unit will remain as master. This method of arbitration allows a predictable outcome for every combination of masters. The unit wishing to send data in the lowest mode and having the lowest unit address of all units contending for the bus will always win bus arbitration. In order to keep any particular unit from hogging the bus a rest time is required for the unit which just exited from a frame. This rest time gives other units, which lost arbitration previously, to arbit among themselves for the bus.

Each of the six fields described above can be built up from combinations of four basic timing bits; 1) start bits, 2) arbiting bits, 3) master to slave bits (MS), and 4) slave to master bits (SM). These bit times are also part of the control protocol (see figure 3).

A start bit is only used during the start field as previously described.

Arbiting bits are used to construct the mode and master address fields. When a unit

sends out its master address (12 bits + parity), there will be 13 arbiting bits used to construct this field. Arbiting bits are very similiar to MS bits except that during an arbiting bit the master units constantly monitor the bus to detect bus collisions.

The MS and SM bits are used to construct the remaining fields. Notice that during these fields a master has already won arbitration and has assumed full control over the bus.

A typical bit time is shown in figure 4. The clock and data are time division multiplexed on the bus. Each bit consists of five parts or T-times:

- a. This is a rest time, insuring that all units see the high level before the bus goes low.
- b. The high to low edge between a and b is generated by the master which synchronizes all other receiving units on the bus.
- c. The transmitter places data on the bus during this time. For a start bit, the data is always a '0'.
- d. The receiver samples the data at the beginning of this time.
- e. Allowance for unit to unit clock tolerances.

Clock synchronization is done during every bit time and not just at the beginning of a frame. Looser clock tolerances result from bit level synchronization.

BUS TIMING EQUATIONS

The actual time duration of the above five parts or T-times for each bit time is derived from an analysis of all the worst case timing differences between masters and slaves communicating on the bus. Within each field the bit time(s) that comprise that field, contain T-times that have been carefully calculated to accomplish the purpose of the field. These are stored in ROM in the actual implementation.

See figure 5 for the system model of the timing calculations. The time taken by a signal at the input of unit A to be clocked into unit B, takes into account the following factors for the T-time calculations:

1. Delay time through unit A driver.
2. Bus propagation time.
3. Bus rise or fall time.
4. Clock phase differences between units A and B.
5. Clock frequency differences between units A and B.

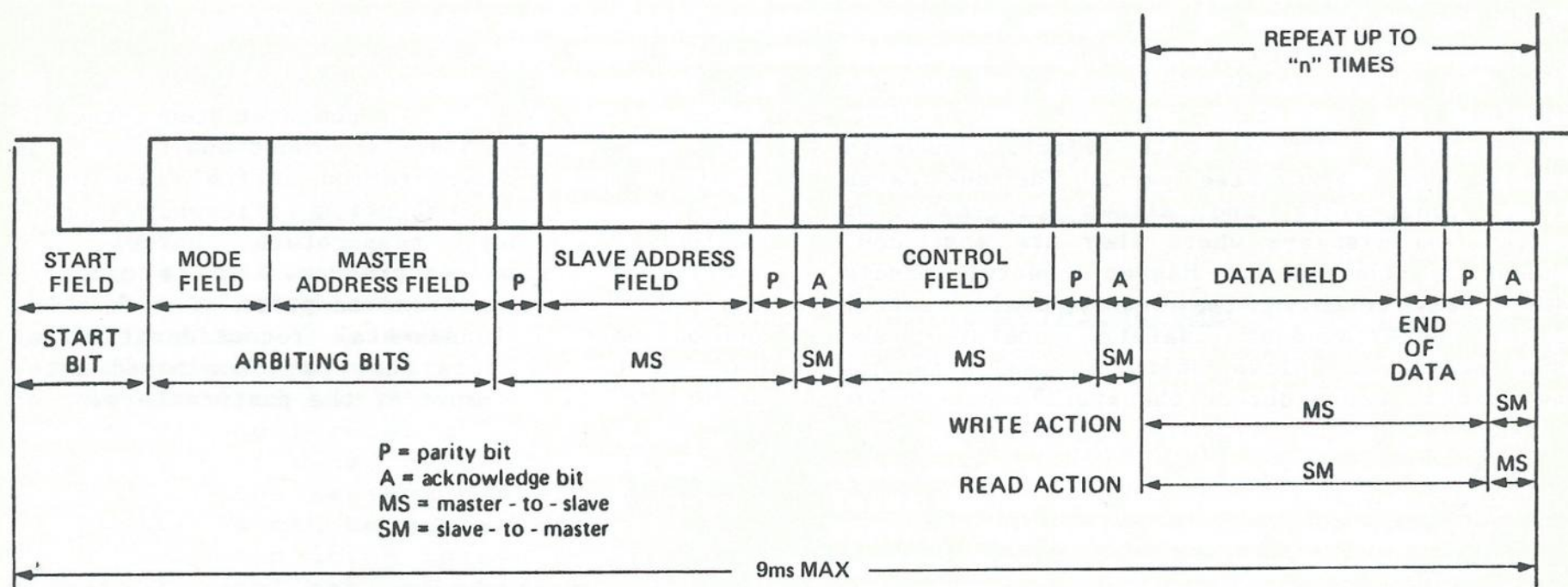


Figure 3. D²B Frame Format with Bit Times

The timing calculations also take into account the differences between a master to slave transmission and a slave to master transmission. In a master to slave transmission, the difference between the best and worst case of the master determines the timing. In the slave to master transmission the worst case combinations of the master and slave have to be taken into account, since the a to b synchronization edge in figure 3 is always generated by the master.

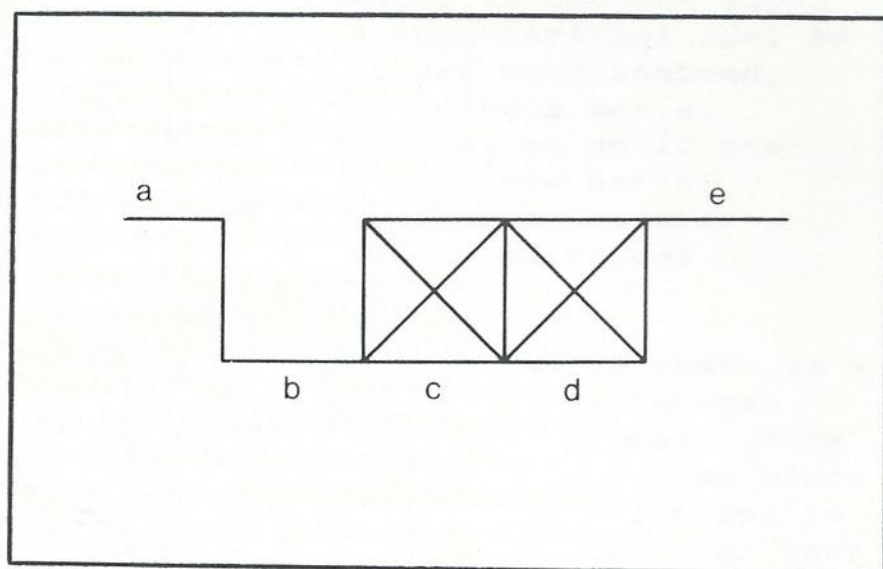


Figure 4. T-Times within a Bit Time

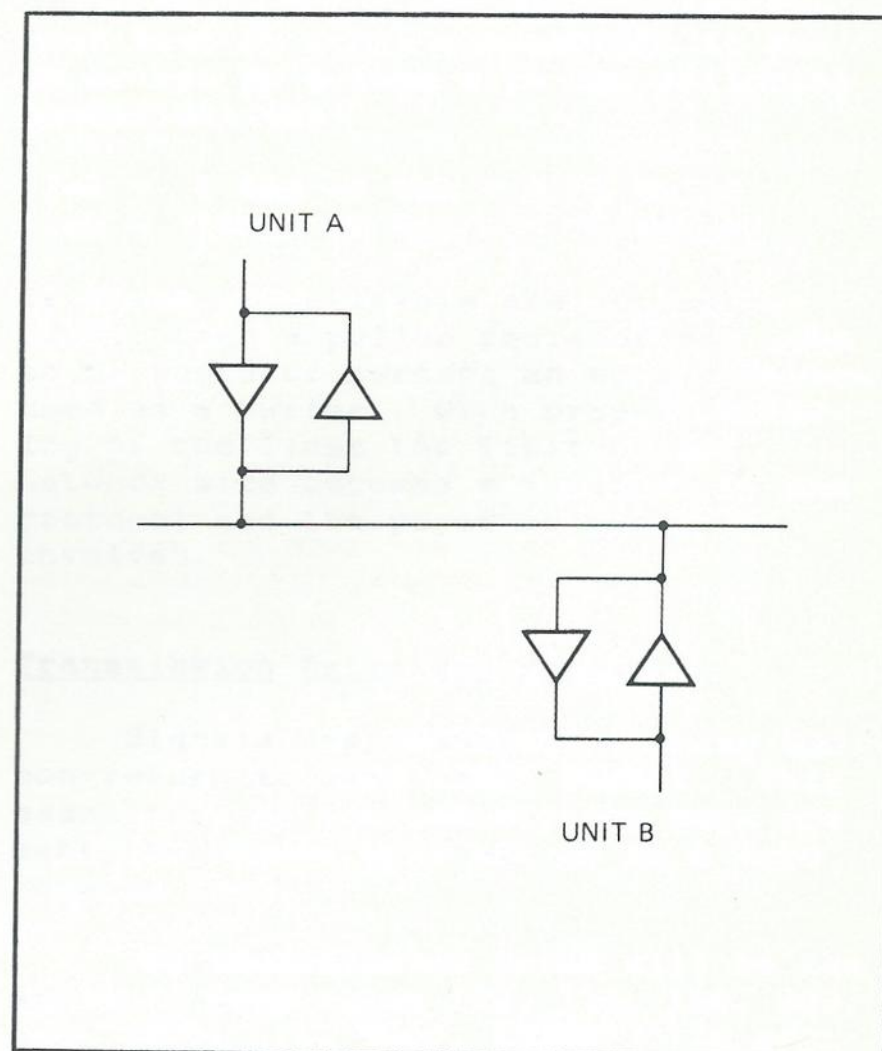


Figure 5. System Model for Timing Calculations

APPLICATION

An example of an application area for the D²B is the replacement of the wiring harness in an automobile with a twisted pair conductor D²B. The electrical specification for the bus minimizes any radiation and the bus differential sensing enables it to reject transients coupled onto the twisted pair. The bus eases wiring constraints and allows placement of control and displays where they are most convenient to the user. Master control panels within easy reach of the driver could control engine, entertainment, safety, comfort, and other services. Slave units at the sites of the services would decode the signals on the bus

and perform the required functions. Options and extras would become easier to add, a power-controlled antenna, for example, could be installed without extensive rewiring.

Diagnostic sensors could also be incorporated so that a computer connected to the bus could analyze functions such as ignition timing, carburation or fuel injection, fuel consumption, combustion efficiency, lubrication efficiency, temperature control, and electrical system performance. It is conceivable that the widespread adoption of such a bus could lead to fundamental reconsideration of many of the concepts that have dominated automobile design for most of the past century.

USING THE 8051'S 9-BIT SERIAL MODE

Alan Steinberg and Ira Horden
Intel Corporation
5000 West Williams Field Rd.
Chandler, AZ 85224

ABSTRACT

Intel has incorporated a special mode in its new products that have serial ports. This mode allows selective reception of a data stream based on the state of the 9th data bit. A simple protocol, referred to as uLANTM, has been established as a standard to ensure that the maximum number of manufacturers can be compatible with each other when using this hardware mode. The applications where a nine bit mode could be used as well as the description of the electrical and data transfer specifications will be covered in this paper.

INTRODUCTION

To produce the highest possible functionality in systems it is frequently necessary to transfer large amounts of information between parts of the system. If the whole system is on one board, or on a set of boards that are in close proximity, the data transfer can be performed over a parallel bus. When the parts of a system are physically separated it becomes expensive to create a long parallel bus, so serial data links are used instead. The low cost of placing a serial communications channel on an IC has made it economical to use serial instead of parallel data transfer schemes even between individual circuits.

In serial communication there is a trade-off that must be made between cost, data integrity, and speed. Many times this is not a major problem since cost sensitive equipment seldom has to handle large amounts of data. For many low to medium speed (up to 500K bits/second) applications the uLAN network is an ideal solution. The electrical characteristics of the network are compatible with many parts and the protocol is simple enough to be handled with little overhead. Data integrity can be confirmed using one or more of three methods: parity, checksums, and

an acknowledge scheme.

ELECTRICAL SPECIFICATIONS

Drive Levels

The hardware of the uLAN network must be compatible with the current Intel UARTs and microcontrollers with UARTs. For this reason the signal levels were chosen to be TTL compatible. This compatibility also has the benefit of being a low cost solution for small systems where no buffers are needed.

When a microcontroller is used directly to drive the serial link the load must be small enough that it can be driven by a line with the following specifications:

$$\begin{aligned} I_{ol} &= 1.6\text{mA at } V_{ol} = 0.45 \text{ Volts} \\ I_{oh} &= -100\mu\text{A at } V_{oh} = 2.40 \text{ Volts} \end{aligned}$$

If these drive levels are not sufficient, then a pullup resistor may have to be added or perhaps an active device used as a buffer. With proper buffering of the lines the limit on the network size becomes a function of the protocol and the power of the CPU involved.

Transmission Rates and Format

Signals are transmitted using the non-return to zero (NRZ) format. This means that if two consecutive ones are sent the signal will not go to a zero between them.

The transmission rates will be the standard baud rates (75, 110, 150, 300, 600, 1200, ..., 9600, 19200). In addition the rate of 62.5 Kbaud, a rate compatible with the Intel microcontrollers, will be used for fast data transfer. The microcontrollers can also function at the standard baud rates.

A maximum deviation of +/- 2.0% of the ideal frequency has been specified

to ensure that all network members can communicate with each other. For an 11-bit transmission, with a stop bit, start bit and 9 data bits, the worst case timing would be off by 20% in the last data bit time slot. If a transmitter and receiver were off by 20% in opposite directions the bit could be off by 40% of the time slot. Since the bit sampling takes place in the middle of the time-slot this 40% error will not cause problems.

DATA TRANSFER SPECIFICATIONS

The μ LAN network is a single master network with provisions for changing masters. This means that at any given time only one device can be the master, and changing masters is done in software as explained later in this paper under "Commands". Physically the devices are connected in a half duplex format; transmission and reception are done on the same line. Messages are transferred in frames or groups of bytes.

Frame Formats

All words transferred using this protocol are nine bits long. The ninth bit is the address/data bit. This bit is set to a one for all address words and a zero otherwise. The data is transmitted in the following format: a start bit, eight data bits (least significant bit first), the address/data bit and the stop bit.

Each frame sent by the master begins with the destination address and ends with the address of the next destination. There is a null address available if no next destination is

desired. This would be used if the transmission was the last in a series.

Slave to master transmission frames consist of the data followed by the slave's address. Since μ LAN is a single master network, there is no need to transmit the master's address in a slave's transmission. The slave needs to transmit its own address at the end of its message so that the master can verify that it was talking to the desired slave.

There are two slightly different data formats which may be used. These are shown in Figure 1. Format I uses either an eight bit data word or a seven bit word with parity. All slaves default to this mode on power up. This mode has the advantage of simplicity and therefore low cost.

Format II uses a word that is always eight bits long. Error protection is provided by an eight bit checksum which is sent as the last word in the message. The checksum is calculated by summing the bytes of the message in modulo 256 arithmetic and then taking the 2's complement negation of the sum. Since it is not part of the data, the address/data specification bit is not included in the checksum. To verify the message, the bytes of the message, including the checksum, are added together in modulo 256 arithmetic. If the result is zero then the message was received correctly.

The minimum amount of time between the last data word or the checksum word and the address word for the next message is specified as four bit times. This time is also the maximum specification on time between words of a message. The four bit times are

Master to Slave Formats:

Format I : ADDRESS, DATA, DATA, . . . DATA, (IMG), ADDRESS

Format II : ADDRESS, DATA, DATA, . . . DATA, CHKSUM, (IMG), ADDRESS

Slave to Master Formats:

Format I : DATA, DATA, . . . DATA, (IMG), ADDRESS

Format II : DATA, DATA, . . . DATA, CHKSUM, (IMG), ADDRESS

Figure 1. Data Transfer Formats

measured from the end of the stop bit to the beginning of the next start bit. It is required to have this delay, referred to as the inter-message gap (IMG), so that the processors will have time to properly recover from parity errors.

Address Formats

All address words are nine bits long. There is a six-bit address field, a write/read bit, a parity bit, and the address/data specification bit. The parity bit is set to even parity including the address/data bit. The address/data bit is always set to a one in address words. The format is shown below:

6-bit address		W/R		P		A/D
---------------	--	-----	--	---	--	-----

Where: W/R is the read/write bit
P is the parity bit
A/D is the address/data bit

The address structure allows one-word addressing for up to 56 slaves. The remaining 8 addresses are used for special functions such as extended addressing. By making use of the extension mode up to 504 slaves can be addressed using two words for each address.

To maximize the capacity of the eight remaining addresses they are separated into "read" addresses and "write" addresses. If the read bit is set, the address is a base address for an extension address. This allows eight different bases, each of which is capable of supplying 56 addresses for a total of 448 addresses. Adding these addresses to the 56 one-word addresses yields a total of 504 possible addresses. All of the slaves must be able to detect a single level of address extension, even if the extension is not part of the slave's address.

The eight "write" addresses are used as either commands or the two special addresses: Broadcast and Escape (the null address). The reserved addresses are shown in the next column:

<u>ADDRESS</u>	<u>FUNCTION</u>
xxx 111 0 P 1	- Extension Address
111 111 1 P 1	- Broadcast Address
011 111 1 P 1	- Escape Code - 'ESC'

<u>ADDRESS</u>	<u>COMMAND</u>
101 111 1 P 1	- Reset
001 111 1 P 1	- Poll
110 111 1 P 1	- Query
010 111 1 P 1	- Configure
100 111 1 P 1	- Request Acknowledge
000 111 1 P 1	- Transfer Mastership

COMMANDS

Commands are used by the master to determine the status of the link and slaves or to change the operational mode of individual slaves or the whole link. Any of the commands can be used at any address except "Broadcast". Since the Broadcast address is really addressing all of the slaves simultaneously, only commands which require no return information make sense with this address. There are only two commands which fall into this category; Reset and Configure. Any data returned in response to other commands must contain a parity bit.

The six μ LAN commands are as follows:

RESET - This command causes the addressed slave or slaves to reset themselves to their initial default conditions.

POLL - After a slave receives a POLL command it returns seven bits of status information. It then resets its status register to the default conditions, except for the Parity/Checksum and Framing Error bits. These bits are not reset until a new specific (not Broadcast) address is received.

The poll response bits, LSB to MSB are:

0 Parity/Checksum Error - A bit error occurred in the previous frame

1 Framing Error - A Word was received with an invalid stop bit (i.e. bit 11 was not a one).

2 Slave Not Previously Addressed - The slave being polled was not addressed by the previous message. For the purpose of this bit a command is not considered a message. Therefore, the bit is cleared only when a

message is sent to another slave; commands (except RESET) will not reset the bit. This bit can be used by the master to determine if the slave received the previous message.

3 Device Not Ready - The peripheral is not ready to accept data.

4 Data Available - The peripheral has data to transmit.

5 Request Mastership - The slave requests mastership of the link.

6 reserved - to be defined

QUERY - This command allows the master to determine the types of slaves on the link and their capabilities. Upon receiving the query the slave returns two or three bytes of information. The first word indicates the capabilities of the slave, with a one indicating that the slave has that capability. The second word indicates the slave's current configuration and if a third word will be sent. The contents of the optional third word is left to the manufacturer and may contain additional information on the device type or features.

The bits of the first word are:

BIT	FUNCTION

0-2	Manufacturer defined bits
3	Data parity capability
4	Mastership capability
5	Acknowledge capability
6	Frame Format II capability

The bits of the second word are:

BIT	FUNCTION

0-3	Manufacturer defined bits
4	Third word follows
5	Acknowledge enabled
6	Frame Format II enabled

CONFIGURE - A master can use this command to change the mode of a slave's serial port if it has more than one mode.

The configure bits are:

BIT	FUNCTION

0-4	Manufacturer defined bits
5	Enable Acknowledge
6	Enable Frame Format II

REQUEST ACKNOWLEDGE (RACK) - The RACK command requests an acknowledge from the addressed slave.

TRANSFER MASTERSHIP - The master sends this command to a slave when it wants to give up mastership. If the slave accepts mastership it returns an acknowledge and the old master becomes a slave and selected slave becomes the master. If the slave does not want to become the master due to parity errors, framing errors, or other reasons it sends the master a negative acknowledge. The master will then continue to be the master. Mastership can be transferred to a slave without first polling the slave, but the slave need not accept mastership.

Error Handling

As there is a wide range of complexity in the members of the link, the error handling techniques will also vary from slave to slave. Unfortunately, it is possible for errors to occur in the address/data specification bit as well as any other bit. For this reason it is suggested that if an error occurs that the slave ignore the command and wait for the next valid address.

If the master determines that the network is having a large number of errors it can cause a break to occur by holding the serial data line low for a few words times. When a slave detects a break on the serial data line it clears the address match status flag and waits for the next address to be sent. This allows all of the slaves to properly re-synchronize themselves to the network.

Acknowledge Procedure

A method of slave acknowledging the master is provided to increase the reliability of the network. If a slave has the capability to acknowledge the master, and the function has been enabled, it is possible for the master to request an acknowledgment from the slave. To do this the master sends a

RACK (Request to ACKnowledge) to the slave. The Slave returns either an positive acknowledge (ACK) word or negative acknowledge (NACK) word depending on its status. The return of an ACK indicates that neither a parity/ checksum or a framing error had occurred.

The acknowledge word is the slave's address with the W/R bit set to a 1/0 to indicate ACK/NACK. If the slave is at an extended address then it must send its complete two word address back to the master.

ADDRESS INITIALIZATION

In some situations several of the same device may be connected to a network. This could present a problem since each of the devices on a network must have a unique address. If these devices are programmable, that is they have fuses, EPROMs, or EEPROMs, this is not a problem as the addresses could be programmed into each part individually. Using programmable parts is a good solution for high end peripherals, but the cost of the programmability might be too high for some low cost peripherals. For these devices another address selection scheme must be used.

There are two basic ways in which a peripheral can read information about its environment: using digital signals or using analog signals. If the peripheral does not have an Analog to Digital Converter (ADC) the information it senses must be digital.

The digital method for address

selection is simply to tie several lines from the chip either high or low. On reset the chip would read these lines and determine its address and default characteristics. If it is desired to save pins, an output line from the chip could be used to multiplex the information coming into the chip. Typically this information would be multiplexed as address/default characteristics.

Using analog signals to determine the initialization information for a chip would involve using one or more of the ADC input lines to sense a resistor divider. The voltage level of the divider could provide as many bits of information to the device as the ADC is accurate. This method requires less I/O lines than the digital method, but changing the divider is more difficult than setting switches and the cost of the ADC is prohibitive unless it is needed for other functions of the chip.

A TYPICAL SYSTEM

A typical μ LAN system could be used either free standing or connected to another system by a high speed data link or a phone line connection. The system would be used to connect circuits, boards, and subsystems to each other over a distance limited only by the types of divers and receivers used. It could be used in applications ranging from multiple-chip consumer products, to desktop computer networks, to industrial control devices and robots. Figure 2 shows a single μ LAN network; several of these systems could be tied together if desired.

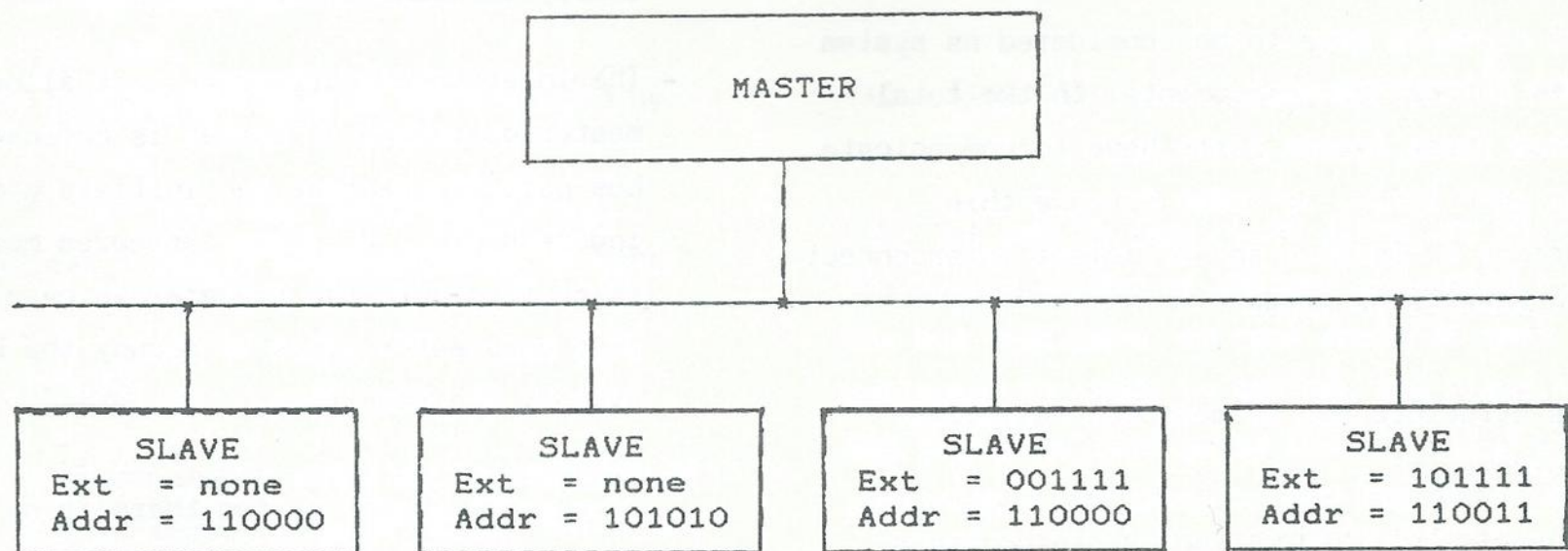


Figure 2. A Typical μ LAN System

THE I²C BUS - AN INTERCONNECT STRUCTURE FOR INTEGRATED CIRCUITS

Ir. A.P.M.M. Moelands
Philips International B.V.
Elcoma division
Eindhoven - The Netherlands

ABSTRACT

The I²C bus is a two-wire serial interface designed for communication between Ics and/or modules. The I²C bus is a true multimaster bus, which means that more than one bus participant can take the initiative to start a transfer. An arbitration procedure is provided to avoid corruption of the data in case more than one master tries to occupy the bus.

The bus clock frequency is not fixed and may be chosen between 0 and 100 Kbits/sec. Application areas in which the I²C bus can be used include consumer -, telephony, low-end industrial and domestic applications.

INTRODUCTION

VLSI circuits have to be considered as system parts rather than components. In the total system these system parts have to communicate with each other. In order to make this possible a flexible and efficient interconnect structure is required.

This structure has to meet the following requirements:

- The digital information has to be transferred in a serial way to reduce interconnection costs and to simplify printed

circuit board design.

- The serial bus must allow bidirectional transfers. This requirement follows directly from functional requirements in the system. For example a serial memory must be read and written via the same bus.
- Ics should be addressed by a binary address rather than enable lines. This again reduces interconnection costs and leaves more I/O pins on the controller for other purposes.
- A receiving device should acknowledge the reception of a message. This allows checks on the transfer, and the configuration of the system can be determined by the master.
- The interconnect structure must allow multi-master operation. A master is defined as a bus participant that can initiate a message independently. Of course measures must be taken to secure the transfer in case more than one master tries to occupy the bus at exactly the same time.

In a multi-master system there is no central controller to control all the functions in the system. This means that a multi-master system can be expanded in a modular way,

not affecting the existing system. The controls and software required for the addition are simply incorporated in an additional controller, which also acts as a master. In a single-master system the central controller continually has to send information throughout the system. The controllers in a multi-master system can restrict themselves to the exchange of information that is of common interest. This reduces the message rate on the bus and makes the use of the bus more efficient. In a multi-master system, real-time interrupts can be handled locally because the intelligence required to do this is distributed.

- The interconnect structure must be realised economically. A low-cost implementation in hardware is of paramount importance because it affects every device connected to the bus.

The I^2C bus meets all the requirements listed above:

- It is a two-wire serial bus using a clock-line and a data line.
- It allows bidirectional data transfers.
- The first byte in a transfer contains the 7-bit slave address. The least significant bit in the first byte is defined as a direction bit.
- Every transferred byte is acknowledged by the receiver.
- The I^2C bus is a real multi-master bus.
- It allows a simple implementation in hardware.

THE I^2C BUS

Before going into a detailed description of the bus the definitions of master and slave are given.

A master is the device which initiates a transfer. It generates start and stop conditions and is responsible for the generation of the clock during the transfer. The master always sends the address of the slave. A master can be either transmitter or receiver.

A slave is the device addressed by the current master. It is controlled by the clock generated by the master.

A slave can be either transmitter or receiver.

I^2C bus - general characteristics

Both the data line SDA and the clock-line SCL are bidirectional lines, connected to a positive supply voltage via a pull up resistor. (see fig. 1)

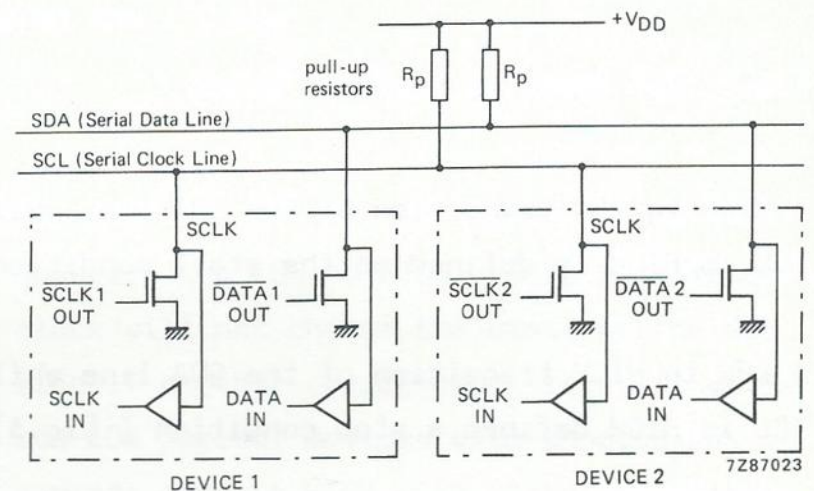


Fig.1 Connection of devices to the I^2C bus

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 in order to perform the wired-AND function. This means that a device which generates a LOW level on the bus overrides a device which generates a HIGH level.

Data on the I²C bus can be transferred at a rate up to 100 kbits/sec. The number of devices connected to the bus and the maximum length of the bus is solely dependent on the limiting bus capacitance of 400 pf.

I²C bus - bit transfer

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. (Fig.2)

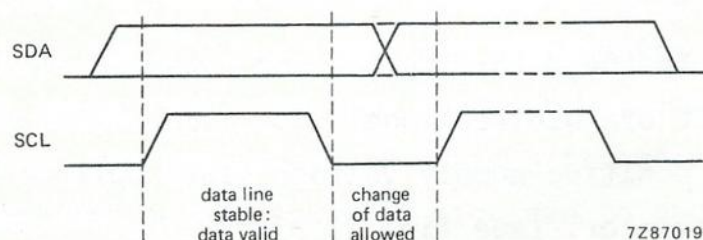


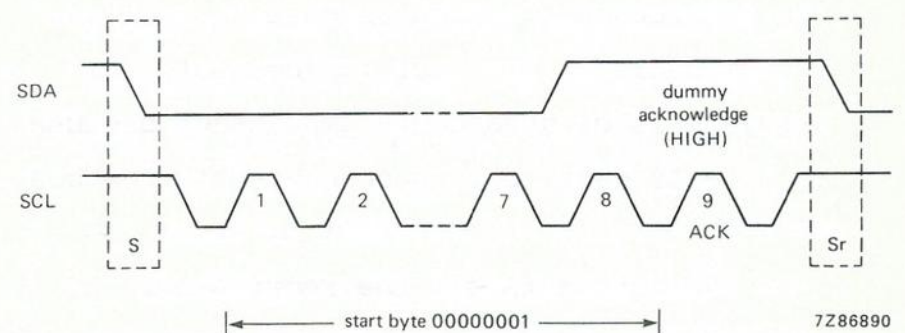
Fig.2 Bit transfer on the I²C bus

A HIGH to LOW transition of the SDA line while SCL is HIGH is defined as the start condition.

A LOW to HIGH transition of the SDA line while SCL is HIGH defines a stop condition (Fig.3)

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.

Detection of start and stop conditions by devices connected to the bus is easy if they possess the necessary interfacing hardware. However, microcomputers with no such interface have to sample the SDA line at least twice per clock period in order to sense the transition. In this case the data transfer can be preceded by a start procedure which is longer than normal.



The start procedure consists of:

- A start condition S
- A start byte 00000001
- An acknowledge clock pulse
- A repeated start condition Sr

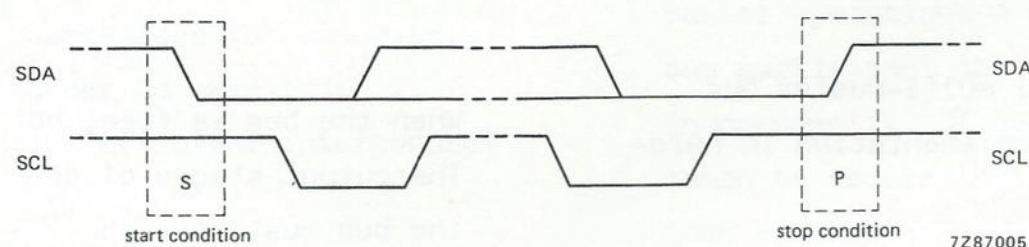


Fig.3 Start and stop conditions

I²C bus - byte format

Every byte 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. (Fig.4)

If a receiving device cannot receive another complete byte of data until it has performed some other function, for example, to service an internal interrupt, it can hold the clock line SCL LOW to force the transmitter into a wait state. Data transfer then continues when the receiver is ready for another byte of data and releases the clock line SCL.

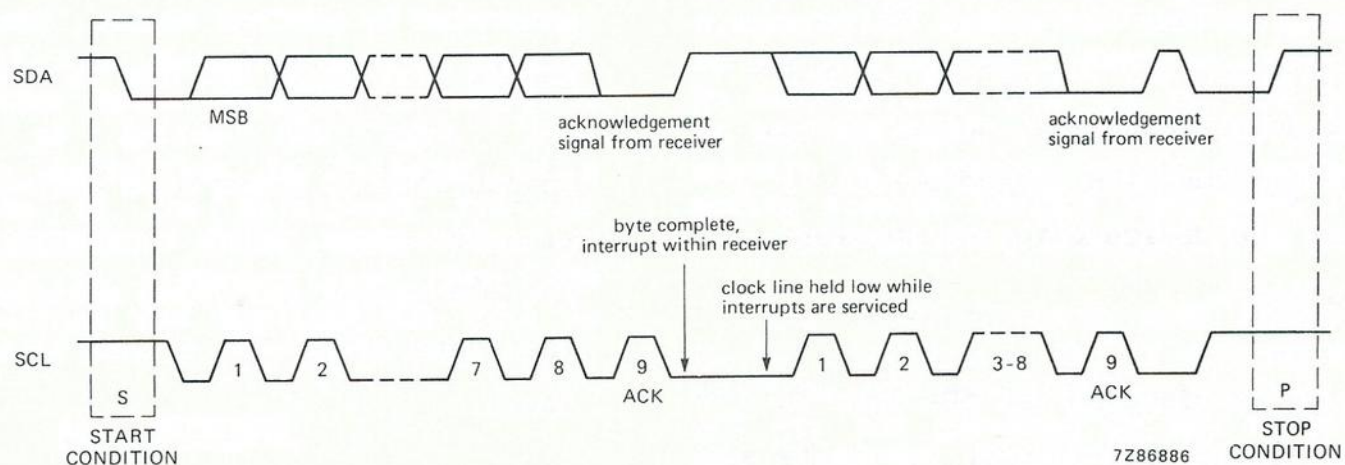


Fig.4 Data transfer on the I²C bus

I²C - Arbitration and clock generation

All masters generate their own clock to the SCL line to transfer messages on the I²C bus. Data is only valid during the clock HIGH period or the SCL line.

A defined clock is therefore needed to enable a bit-by-bit arbitration on the data line if more than one master starts generating clock pulses.

Clock synchronisation is performed using the wired - AND connection of devices to the SCL line. This means that a HIGH to LOW transition on the SCL line will force the devices concerned to generate their LOW period. Once a device clock has gone LOW it will hold the SCL line in that state until the end of the LOW period is reached.

However, the LOW to HIGH change in this device clock will not change the state of the SCL line if another device clock is still within its LOW period. Devices with shorter LOW periods enter a HIGH wait state during this time. (Fig.6)

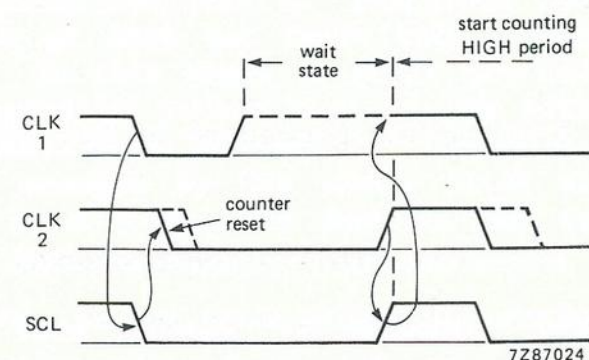


Fig.6 Clock synchronisation during the arbitration procedure

When all devices concerned have finished 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 devices will start generating their HIGH periods. The first device to complete its HIGH period will again pull the SCL line LOW.

In this way, a synchronised clock SCL is generated for which the LOW period is determined by the device with the longest clock LOW period while the HIGH period on SCL is determined by the device with the shortest clock HIGH period.

Arbitration takes place on the SDA line in such a way that the master which transmits a HIGH level, while another master transmits a LOW level, will switch off its DATA output stage since the level on the bus does not correspond to its own level. Arbitration can carry on through many bits.

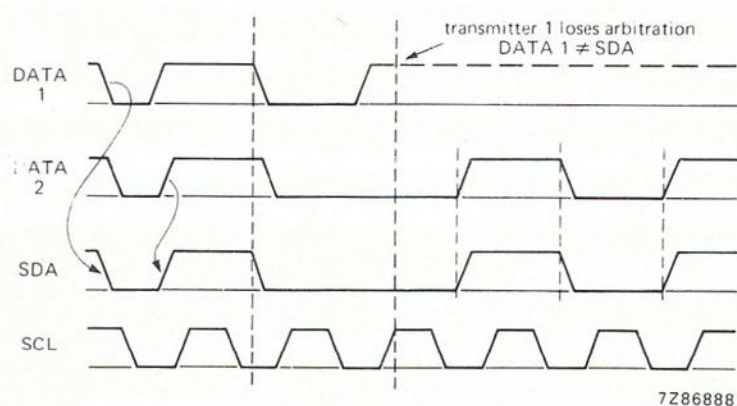


Fig.7 Arbitration procedure of two masters

Fig.7 shows the arbitration procedure for two masters. Of course more may be involved depending on how many masters simultaneously try to occupy the bus.

As control of the I²C bus is decided solely

on the address and data sent by competing masters, there is no central master, no any order of priority on the bus.

It must again be emphasized that arbitration only takes place between masters that initiate messages at approximately the same time.

Normally when a master which wants to make a transfer has detected a start condition generated by another master, the first master has to wait until the bus is free again.

In addition to being used during the arbitration procedure, clock synchronisation can be used to enable receiving devices to handle fast data transfers, either on a byte or bit level.

I²C bus - formats

The first byte after a start condition always contains the slave address and the direction bit, which is located at the least significant position.

If it is necessary to change direction within a transfer, the master repeats the start condition (without generating a stop condition) and re-transmits the same slave address but now with the direction bit inverted.

The number of bytes in one transfer is not resrestricted, and is left to the responsibility of the system designer.

A detailed description of the electrical specifications and bus timing can be found in the Philips and Signetics I²C bus specification.

