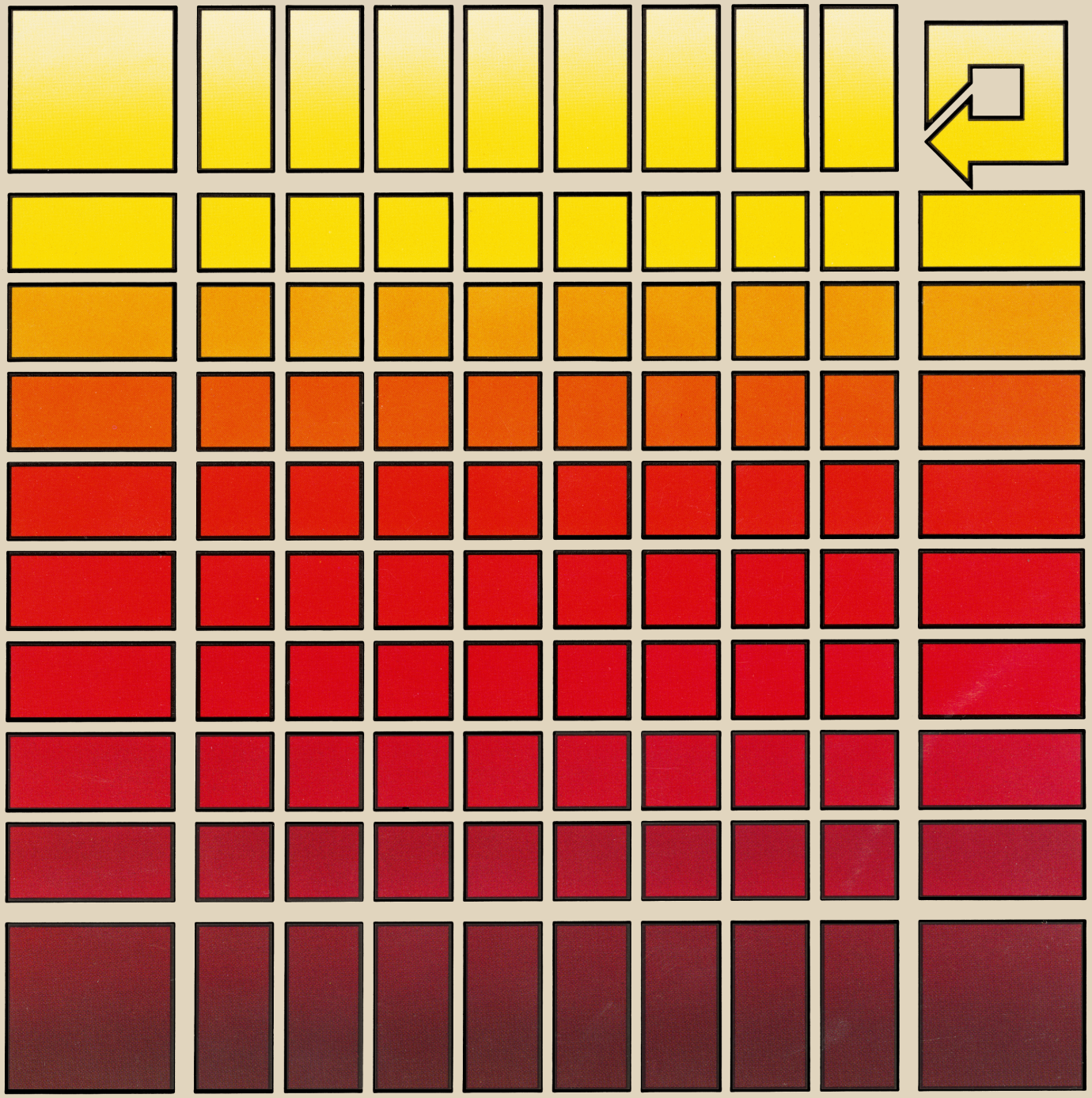
HEWLETT-PACKARD

The HP-IL Interface Specification



HEWLETT PACKARD

The HP-IL Interface Specification

November 1982

82166-90017

Printed in U.S.A.

(C) Hewlett-Packard Company 1982

Preface

Chapter one provides an overview of the HP-IL interface system with some example implementations. The complete functional, electrical, and mechanical specifications of HP-IL are contained in chapters two, three, and four respectively. Chapter five discusses system considerations such as recommended sequences of interface messages, asynchronous loop operations, and device compatibility. In the appendices information such as message coding and glossary of terms may be found. This manual is not intended to be a tutorial on HP-IL or I/O techniques, therefore, familiarity with other interfaces is very helpful. Because of the extensive similarity between HP-IL and HP-IB, an understanding of the latter is recommended, though not required.

CONTENTS

Chapter 1: Introduction

1.1 General Information 1-1
1.1.1 Application Examples 1-1
1.2 Functional Overview 1-3
1.3 Electrical Overview 1-7
1.4 Mechanical Overview 1-10

Chapter 2: Functional Specifications

2.1 Introduction 2-1
2.1.1 Logical Partitions 2-1
2.2 State Diagram Notation 2-5
2.3 R (Receiver) Function 2-7
2.4 D (Driver) Function 2-10
2.5 AH (Acceptor Handshake) Function 2-13
2.6 SH (Source Handshake) Function 2-16
2.7 C (Controller) Function 2-19
2.8 T (Talker) Function 2-25
2.9 L (Listener) Function 2-31
2.10 SR (Service Request) Function 2-36
2.11 RL (Remote Local) Function 2-40
2.12 AA (Automatic Address) Function 2-43
2.13 PD (Power Down) Function 2-48
2.14 PP (Parallel Poll) Function 2-51
2.15 DC (Device Clear) Function 2-53
2.16 DT (Device Trigger) Function 2-55
2.17 DD (Device Dependent Command) Function 2-57
2.18 Remote Message Coding 2-59

Chapter 3: Electrical Specifications

3.1 General 3-1
3.2 Electrical and Logical Relationships 3-2
3.3 Output Specifications 3-3
3.3.1 Open Circuit Voltage 3-3
3.3.2 Output Impedance 3-5
3.3.3 Common Mode Output Voltage 3-10

3.4 Input Specifications 3-10
3.4.1 Input Impedance 3-11
3.4.2 Input Hysteresis 3-11
3.4.3 Input Test Circuits 3-11
3.4.4 Input Filtering 3-13
3.5 Interface Cable Specifications 3-14
3.5.1 Cable Type 3-14
3.5.2 Characteristic Impedance 3-15
3.5.3 Cable Rise Time 3-16
3.5.4 Cable Loss 3-16
3.6 Isolation Requirements 3-18
3.7 Electromagnetic Compatibility (EMC) 3-19
3.8 Electrostatic Discharge (ESD) 3-22

Chapter 4: Mechanical Specifications

Chapter 5: System Guidelines

5.1 System compatibility 5-1
5.2 System Configuration 5-2
5.3 Address Assignment 5-2
5.4 Asynchronous Operations 5-4
5.4.1 Power Up and Error Recovery 5-5
5.4.2 Loop Integrity Check 5-5
5.4.3 Asynchronous Service Requests 5-6
5.5 Operational Sequences 5-7

Appendix A: Capability Subsets

Appendix B: Message Glossary

B.1 Local Messages and Pseudomessages B-1
B.2 Remote Messages B-6

Appendix C: Message Coding

C.1 Command Coding C-1
C.2 Ready Coding C-2
C.3 Frame Hierarchy C-3
C.4 Accessory Identification C-4
C.5 System Status Messages C-7

1. INTRODUCTION

1.1 General Information

HP-IL (Hewlett-Packard Interface Loop) is a system which permits communication between devices. In comparison with other interface systems HP-IL is small, low power, low cost, and medium distance. HP-IL has the capability to transfer data at a faster rate than commonly available serial interfaces. As the name implies, devices are connected in a circular loop structure. Digital messages travel from one device to the next around the loop in one direction only. All devices must obey certain functional, electrical, and mechanical rules in order to communicate by means of HP-IL.

This chapter provides an introduction to the HP-IL interface specifications which are presented in chapters 2, 3, and 4. Because this manual is not tutorial in nature, the novice may wish to refer to "THE HP-IL SYSTEM"* for a complete tutorial before proceeding.

1.1.1 Application Examples

The following three applications help show the wide range of capabilities available with HP-IL.

For his dissertation, a forestry graduate student is studying the effect of various types of herbicides on underbrush, tree growth, etc. Observations are made on several different plots many miles from the school. Because of its low cost, small size, and continuous memory feature he uses the HP-41CV handheld computer as a data collector. A short program he has written prompts him for the correct data items and stores them in the appropriate registers. The university forestry lab has a small HP-IL system including the HP 82160A HP-IL Module for the

* Gerry Kane, Steve Harper, and David Ushijima, "THE HP-IL SYSTEM: An Introductory Guide to the Hewlett-Packard Interface Loop" (Osborne/McGraw-Hill 1982).

HP-41CV, the HP 82161A Digital Cassette Drive, the HP 82162A Thermal Printer, and the HP 7470A Plotter. When the student returns the lab he connects his HP-41CV to the HP-IL system and inserts a minicassette into the tape drive. The cassette contains his programs and the data he has collected on his project over the last several months. He loads and runs a program from the minicassette which reads the data taken today from the HP-41CV and stores it on the tape with the rest of the data. He then loads another program which uses all the collected data on the tape to make finished graphic plots and histograms of various combinations of the data, such as toxicity levels over time, amount of undesirable foliage with respect to the type of herbicide, etc. Prior to the HP-41CV and HP-IL, the cost of such an application would have been prohibitive. Furthermore, the low power requirements and the power down feature of HP-IL permit unattended operation for extended time periods. If the student had needed to measure toxicity level, for example, every hour for a week, a small system in a weatherproof box could be left on site under battery power. Naturally, the measuring apparatus would need to be connected to the HP-IL system, but this could be accomplished with the HP 82166A HP-IL Converter.

Each jewelery store in a small chain uses from two to five point-of-sale terminals consisting of an HP-75C computer and an HP 82162A Thermal Printer. These devices are all connected on HP-IL to an HP 82161A Digital Cassette Drive and an HP 82905B 80 column printer located in the back room of the store. Each HP-75C contains a program which prompts the salesclerk for all necessary information about each transaction. The data, which might include item description, inventory number, price, and charge card number is stored on the cassette drive. A customer receipt is printed on the small printer on the counter and a transaction summary is printed simultaneously on the larger printer in the back room. At any time, a special program can be used to generate sales reports, lists of items to be re-ordered, etc. With the HP 82164A HP-IL/RS-232-C Interface and a modem the transaction or inventory data could be transmitted by telephone to the main store. Prior to HP-IL, such a system would have been much too expensive for very small businesses.

A small electronics firm is about to begin volume production of a new audio amplifier printed circuit board. The production engineer has designed an automatic test system consisting of the HP-85A Personal Computer with the HP 82938A HP-IL Interface, and the HP 3468A HP-IL Digital Multimeter. The engineer has also built a special device containing a test fixture for the PC boards, a programmable power supply, a programmable waveform generator, and relay switches to connect

these devices to the various PC board test points. This special device uses the HP 82166A HP-IL Converter so that it too is controlled with HP-IL. After initial debugging, the engineer writes the test program so that up to three identical test stations can be controlled on the same loop since he has found that HP-IL and the HP-85A are fast enough to support this throughput. Test programs and results are stored on the internal cartridge tape drive for future failure analysis. Later when a second version of the amplifier board goes into production some simple program modifications permit testing of both versions at the same time on different test stations using the same system. The low cost and flexibility of HP-IL were the deciding factors in the use of automated testing as opposed to a traditional manual approach.

1.2 Functional Overview

HP-IL is a master-slave interface system. One of the devices on the loop is designated the loop controller and this device has the responsibility to transmit all commands to other devices on the loop. The HP-41C and the HP-85A are examples of devices that can be HP-IL controllers when they are equipped with the proper plug-in module. A device with the ability to send device dependent data to other devices on the loop is called a talker. Note that even though a device has the talker capability it must not actually send its data until commanded to do so by the loop controller. The HP 82161A Digital Cassette Drive is a device which can be a talker. Listeners are devices with the capability to receive data from the loop. Once again, listeners must remain inactive until they receive a command from the controller which enables them to receive data. The HP 82162A Thermal Printer is a device which can be a listener.

Controller, talker, and listener functions are the three basic capabilities of HP-IL devices. A device may have only one of the three or may include some combination of capabilities as is more often the case. Talkers often have listener capability and controllers almost always have both talker and listener capabilities as well. In general, every HP-IL system has all three capabilities somewhere within its devices; however, under special circumstances a system can be constructed with only a talker and one or more listeners without the controller. In this case, all devices must be able to operate in the talk-only or listen-only mode. An example might be a voltmeter logging

readings on a printer. In larger loop systems there are usually several talkers and listeners. The controller will permit only one talker to be active at a given time, but may enable multiple listeners to receive the talker's data, if desired. There may even be several controllers on one loop. In this case, one of the controllers (the system controller) is in charge when the system is first turned on. Protocol exists within HP-IL to allow the various controllers to take turns controlling the other devices on the loop. Only one of the controllers is active at one time.

In addition to the talker, listener, and controller functions the following capabilities are available on HP-IL:

- Devices may indicate a need for service from the controller using two different methods.

- Power down mode may be used to conserve power.

- The controller may assign addresses in three different ways for a maximum of 31 or 961 devices.

- Devices may be identified by model number or a device capability code.

- Devices can be triggered, cleared and commanded to ignore local controls and only respond to loop messages.

- The talker may be temporarily interrupted during long data transmissions so that other tasks may be performed.

This wide range of capabilities permits the controller to be in charge of both the configuration and the operation of the loop. It is possible to implement high-level functions in the controller which make the details of loop protocol entirely transparent to the user. In such a friendly system, the user would simply execute a function and the computer would automatically configure the loop, find the desired device, and send the appropriate commands and data.

There are two major classes of messages which are sent over the loop. Interface messages are usually sent by the active controller and are used to direct the operation of the system. The controller's commands are an example of this type of message. In general, these messages are considered overhead. The primary purpose of HP-IL, of course, is the transmission of data. Data messages are used as a communication medium and do not directly affect the operation of the interface system. Data messages are sent by the active talker and received by one or more active listeners.

Normally only one message is in transit around the loop at any given time. In general, when a device sources a message it waits for the message to go completely around the loop and return before transmitting the next message. Most messages are held at each loop device until it is ready to accept the next message. This is the normal loop handshake and it guarantees that talkers and controllers do not send messages faster than devices can accept them. This handshake also provides an excellent error-checking capability. If the message returns the same as it was sent, the sending device knows that all devices must have received it correctly.

Clearly, if there are a number of slow devices on the loop that delay each message until they are ready to pass it on to the next device, the loop speed will be unacceptably low. Fortunately, data messages are usually intended for only one or two devices at a time. Other devices on the loop are in their inactive state. Devices are built so that messages not meant for them are passed on very rapidly. In this manner the loop throughput can be maintained at an acceptable level.

Since commands are often meant for all devices on the loop, the handshake process for commands is modified slightly in order to maintain reasonable throughput. When a device receives a command message, it passes it on immmediately but also retains a copy of the message for interpretation. In this manner all devices can be executing the same command in parallel. The message returns to the controller fairly rapidly, but its return does not indicate that the devices are ready. The controller must now send a special message called Ready For Command (RFC). This message is held by each device until it is ready for another message. If all devices execute the command in about the same amount of time, there will be a delay until the first device is finished and then the RFC message will move fairly rapidly around the loop to the controller. Now the controller knows that all devices are ready and the next message can be sent.

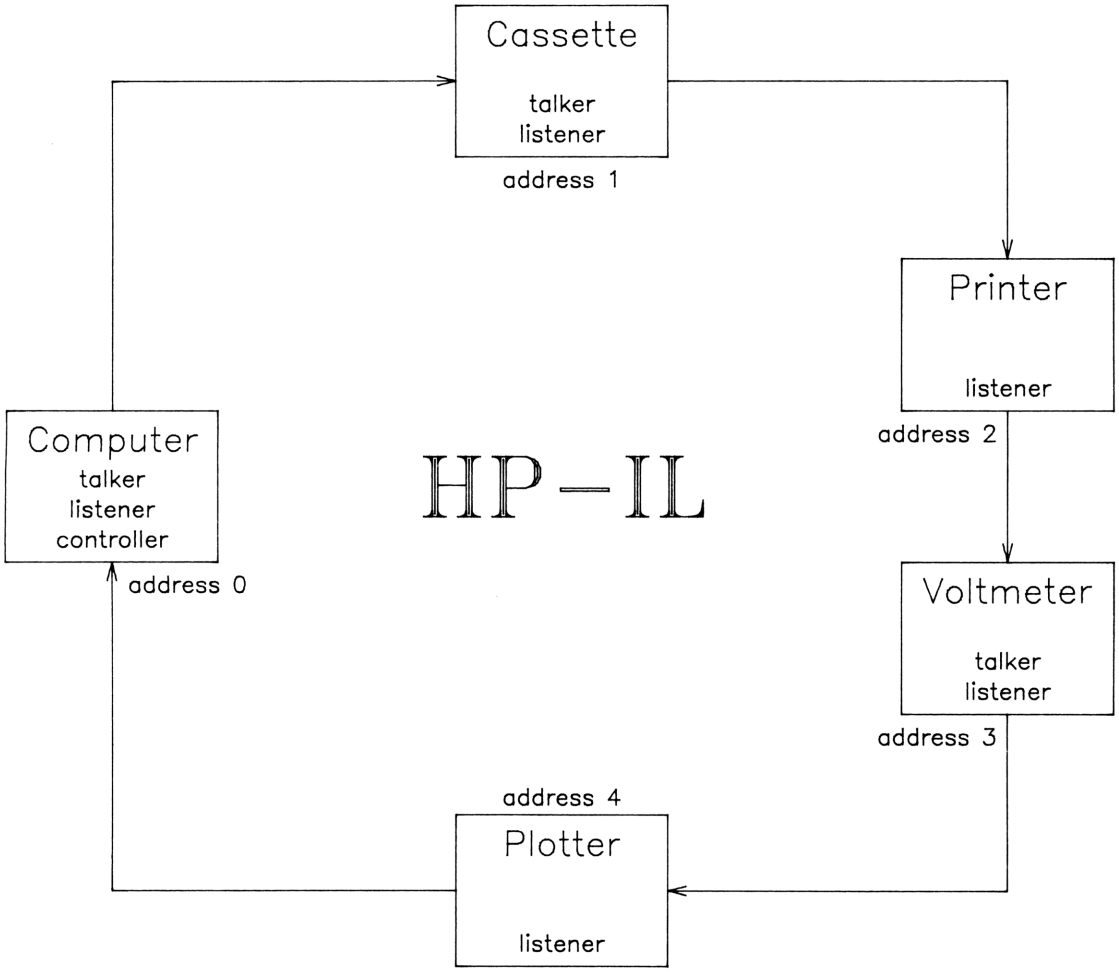


Figure 1-1. Sample HP-IL System

Suppose that we have an HP-IL system consisting of devices with basic capabilities and loop addresses as shown in figure 1-1. Assume that the voltmeter is ready to transmit a reading on the loop, and the user wishes to print it on the printer. The computer first sends the Unlisten command which causes any previous active listener devices to become inactive. When the command returns the controller sends the Ready For Command (RFC) message as is required after every command. The calculator now sends the Talk Address 3 command (and the RFC) to direct the voltmeter to be the talker and to cause any previously addressed talker to become inactive. At this point, even though the voltmeter is addressed to talk, it may not send its data until specifically instructed to do so. The computer then sends the Listen Address 2 command and RFC. The printer alone recognizes this command and becomes ready to receive data messages. This command does not cause other listeners to go inactive, hence the need for the Unlisten command at the beginning. At this point the computer sends a special message

called Send Data to cause the voltmeter to begin sourcing data messages. The Send Data is in the ready class of messages which is used for handshake purposes. When the voltmeter receives the Send Data it does not retransmit it, but instead sends its first data message. The voltmeter sends a sequence of data bytes, for example, +2.658VDC followed by carriage return and line feed characters. Each data byte goes around the loop to the printer, which holds the byte momentarily until it is loaded into the buffer, and back to the voltmeter. After each byte is received and error-checked the voltmeter sends the next character. When the line feed returns and error-checks properly the voltmeter sources a special End Of Transmission ready message. This message signals the computer that the transmission is now complete and that the computer should resume active control of the loop. Similar to the Send Data message, the computer does not retransmit the End Of Transmission but replaces it with the next command message.

It must be noted that HP-IL protocol is based very strongly on HP-IB (Hewlett-Packard's implementation of IEEE-488). While it appears that HP-IL is functionally a bit-serial version of HP-IB, the user is warned that there are some significant differences in protocol. While familiarity with HP-IB is very helpful, it is not necessary. All information needed to learn and use HP-IL is contained in this document.

1.3 Electrical Overview

Every message on the loop is sent as a sequence of eleven bits called a message frame. The first bit, called the sync bit, is coded in a special way so that each device can easily recognize the beginning of a frame. The sync bit and the next two bits are called control bits and they determine the major classification of a message. Command messages, ready messages, and data messages are examples of these major classes. The remaining eight bits, sometimes called data bits (not to be confused with the data message) specify the particular message within the classification. The Unlisten command frame, for example, would be 100 00111111, while the Interface Clear command is 100 10010000. The Send Data ready message is 101 01100000. The space between the control and data bits is for clarity only and does not represent a time delay or any other delimiter.

The electrical connection from one device to the next is a differential, voltage-mode, two-wire, balanced line. Both wires are floating with respect to both devices' ground connections. One of the wires is chosen as a reference and the voltage of the other wire is measured only with respect to the reference. This has several advantages. Device grounds need not all be at the same potential. In fact there can be rather large differences with no effect. This is especially handy for voltmeters which need to measure values not referenced to ground. It totally avoids the problem of ground loops in an interface system. Since noise pulses tend to affect both wires equally, the balanced nature of HP-IL strongly rejects these pulses. The same holds true for noise radiation from the system itself. As one wire's voltage rises, the other's falls, tending to cancel any radiated signal.

Bits are encoded using a three level, or bipolar code. The voltage difference between the two wires may be nominally -1.5 Volts, Ø Volts, or +1.5 Volts. A logical one is encoded as a high pulse (+1.5 Volts) followed by a low pulse (-1.5 Volts). A logical zero is a low followed by a high. A logical one sync is a high, low, high, low sequence. A zero sync is low, high, low, high. The nominal pulse width is one microsecond and each bit sequence is always followed by a minimum delay time (Ø Volts) of about two microseconds. Refer to figure 1-2. This type of code provides good noise immunity, is relatively insensitive to speed variations, is self-clocking, and has no DC component in the signal.

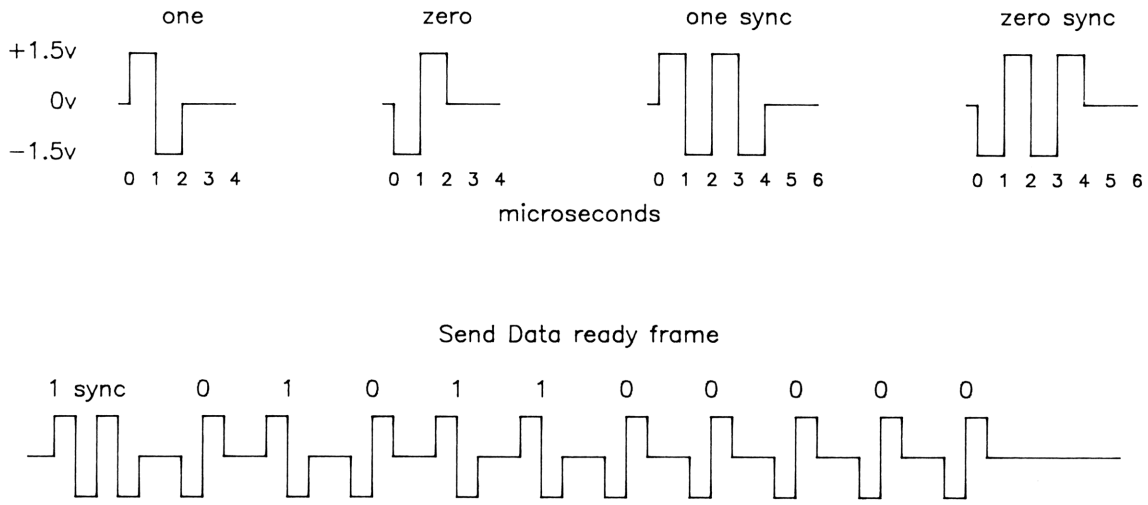


Figure 1-2. HP-IL Bit Encoding

Frames are asynchronous with respect to each other. That means no common system clock is necessary in HP-IL. Furthermore, even bits within frames are asynchronous. The sync bit requires six microseconds to transmit while other bits take four microseconds. A complete frame, which may contain one byte of data, takes 46 microseconds assuming no extra delay between bits. If we further assume no extra delay between frames, a maximum loop rate of approximately 20 kilobytes per second could be achieved. In a typical implementation with present electronics, other hardware and software delays lower this rate to somewhere between three and five kilobytes per second. This is enough to transmit more than a full single-spaced typewritten page of data every second.

Figure 1-3 is a simple block diagram of the interface electronics used in a particular implementation of HP-IL in a device. The HP-IL input and output lines are isolated through simple pulse transformers. The discrete components provide the proper load value for input and filtering for EMI reduction with impedance matching for the output side.

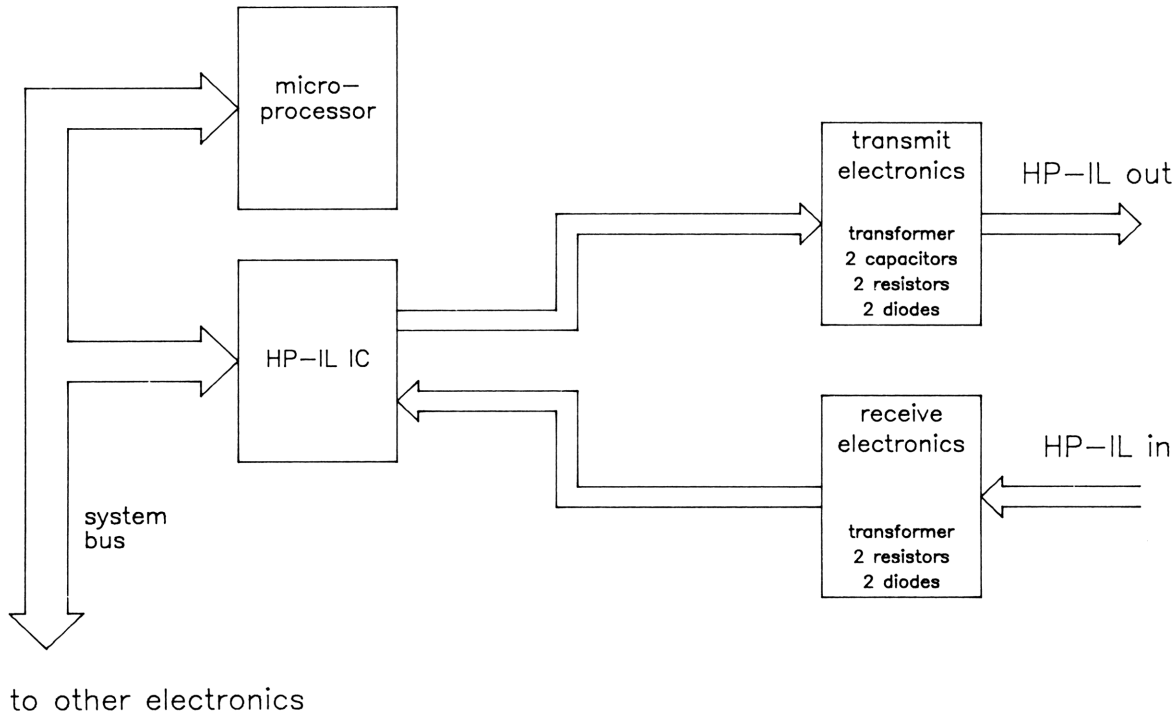


Figure 1-3. Example Device Electronics

Many of the HP-IL interface functions can be implemented in a single LSI integrated circuit with the remaining aspects of the protocol implemented in firmware using a standard microprocessor. The microprocessor may also be used to control the device functions and the interaction between the device and the interface. Since most devices will require the microprocessor anyway, its cost is only partly attributable to the interface and partly to the device itself.

1.4 Mechanical Overview

Since HP-IL utilizes a loop structure, each device only needs two connectors, an input and an output, regardless of how many devices are on the loop. In the interest of small size, HP-IL uses a special connector. Both connectors panel mounted together only require about 0.4 inches by 1.5 inches of panel space and about 0.8 inches of depth. If less space than this is available the lines themselves can go directly into the panel through strain reliefs with cable connectors at the loose ends. connectors are constructed so that the reference line cannot be reversed and also so that an output cannot be plugged into another output. Cable connectors plug together properly to form running cable "splices". See figure 1-4.

For relatively short distances up to a maximum of 10 meters from one device to the next, very low cost "zip cord" can be used as shown in the illustration. Note that the "rib" on the "zip cord" always corresponds to the reference line. For longer distances up to 100 meters between devices, shielded twisted pair cable must be used. No change is necessary in the device electronics for the longer distance. Only the cable between those particular devices that are farther than 10 meters apart must be changed.

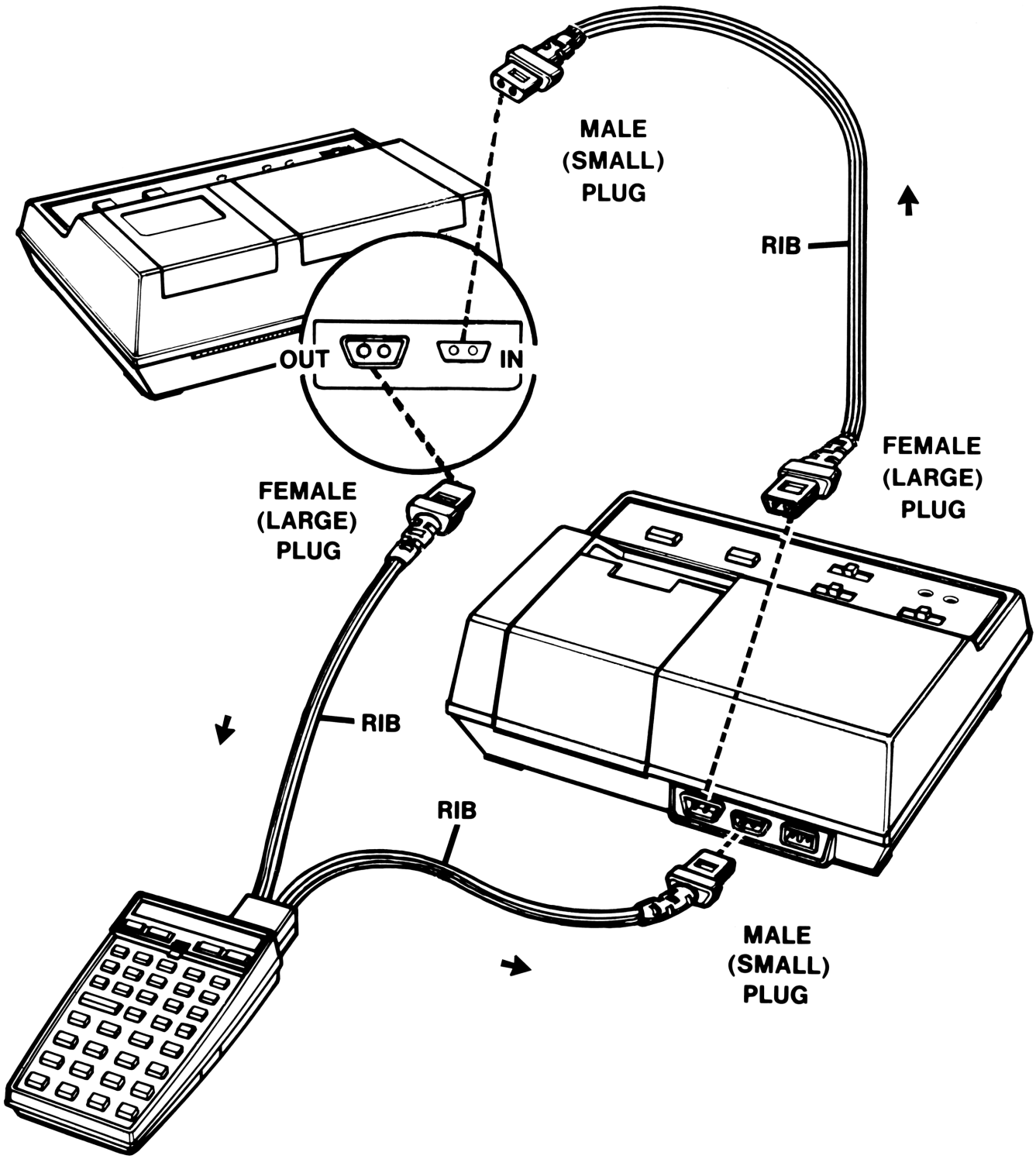


Figure 1-4. HP-IL Connectors

2. FUNCTIONAL SPECIFICATIONS

2.1 Introduction

The complete functional specification of the HP-IL system is contained throughout this chapter in the state diagrams, associated text, tables, and examples. In cases of apparent disagreement, the state diagrams are to be used as the primary reference with all other material of secondary importance. All cases not specifically accounted for in the diagrams or text are to be ignored.

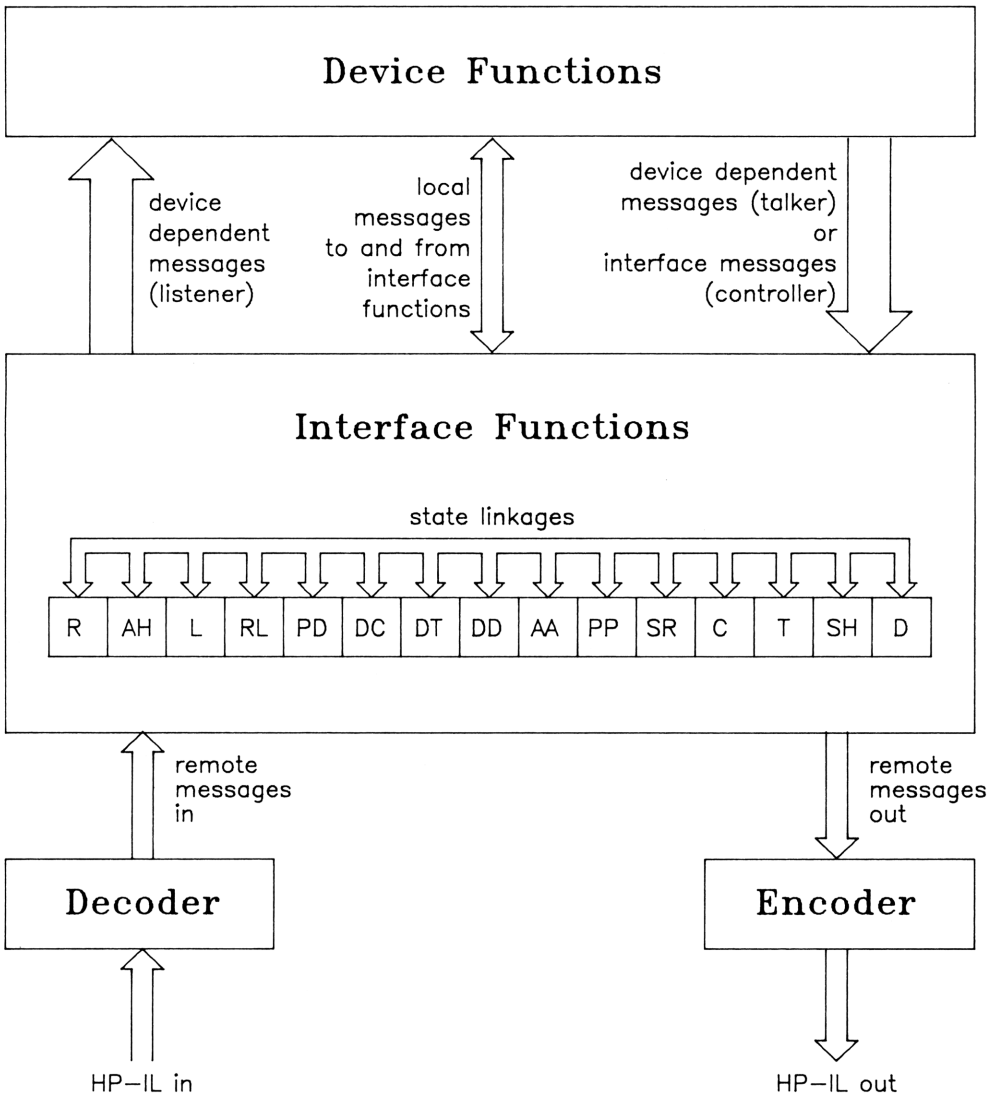
An HP-IL device may be partitioned conceptually into three functional areas, each with unique capabilities: device functions, interface functions, and message coding logic. Figure 2-1 shows how these functional areas interact within each HP-IL device.

2.1.1 Logical Partitions

Device functions are those unique features and capabilities that each device possesses. They represent the entire motivation for building any device and are specified by the device designer. The device functions can affect the operation of interface functions with local messages.

Interface functions are the basic operational units of the HP-IL system. Through the interface functions, the device may receive, process, and send messages. Interface functions are not specified by the designer, but are defined in this chapter. All interface functions within a physical device must conform to the specifications exactly in order for the device to properly communicate with other devices in the HP-IL system.

The message coding logic performs transformations between logical interface messages and signal line values.



Section	Interface Function		Section	Interface Function	
2.3	R	receiver	2.11	RL	remote local
2.4	D	driver	2.12	AA	automatic address
2.5	AH	acceptor handshake	2.13	PD	power down
2.6	SH	source handshake	2.14	PP	parallel poll
2.7	C	controller	2.15	DC	device clear
2.8	T	talker	2.16	DT	device trigger
2.9	L	listener	2.17	DD	device dependent command
2.10	SR	service request			

Figure 2-1. HP-IL Functional Partitions

The interface functions are defined in terms of one or more groups of interconnected states called state diagrams. The state diagrams define only the external behavior of the interface functions and are not intended to limit the design. Interface functions may be implemented with any combination of hardware, firmware, or software as long as the external behavior is the same. Refer to figure 2-1 for a complete list of interface functions and their mnemonics.

Within each connected group of states, one and only one state may be active at any given time. The state variable associated with a state is true if and only if that state is active. Definitions given for each state describe the messages that must be sent while that state is active and the conditions which cause a transition to another state within the group.

All communications between an interface function and its environment is through messages or combinations of messages. A complete understanding of messages and of figure 2-1 is essential to understanding the state diagrams and the entire interface system. Each message has a logical value associated with it which is either true or false at any specific time. The inputs to an interface function are remote messages, pseudomessages, state linkages, and local messages. The outputs are remote messages and local messages (state variables).

Local messages are those messages sent between the device functions and the interface functions. The set of local messages sent to the interface functions is defined and may not be changed by the device designer. Local messages sent from interface functions to device functions may include any state variable or combination of state variables and are completely designer specified. Local messages sent by device functions must exist for enough time to cause the required state transitions.

Remote messages are translated to or from the interface signal line values through the process of encoding and decoding. Interface functions must ignore any message coding not specifically defined.

Pseudomessages (not shown in figure 2-1) are a small set of messages similar to local messages which are sent by the message coding logic to the interface functions.

A state linkage is a logical interconnection between an active state of one state diagram and a transition between states in another. This is the vehicle for the extensive interaction between the interface functions.

Prior to decoding, remote messages consist of eleven-bit message frames. The three control bits place the message in one of four major classes, and the eight data bits specify the particular message within the category. These bits are sent in the following order and designated as shown:

C2 C1 CØ D7 D6 D5 D4 D3 D2 D1 DØ

Note that C2 is also the sync bit indicating the start of a frame. The major classes of messages are:

1. Data or End messages (DOE). These are device dependent messages which are sourced by the active talker and received by the active listener(s). End messages (control bit C1 set) are data messages that indicate an end-of-record condition (not end of transmission). The service request message (control bit CØ set) indicates to the controller that one or more devices on the loop need attention.

2. Command messages (CMD). These interface messages are always sourced by the active controller. Several groups of commands are defined by the coding of the data bits. These groups are UCG (universal command group) messages, ACG (addressed command group) messages, TAG (talk address group) messages, LAG (listen address group) messages, and SAG (secondary address group) messages. All devices respond to UCG, TAG, and LAG commands, but only devices enabled by a talk or listen address respond to ACG or SAG commands.

3. Ready messages (RDY). These interface messages are sometimes sourced by the active controller and sometimes by the active talker or active listener. They are used for handshake purposes such as initiating or terminating a data transmission. Groups of ready messages are RFC (ready for command group) messages, ARG (addressed ready group) messages which include SOT (start of transmission) and EOT (end of transmission) messages, and AAG (auto address group) messages.

4. Identify messages (IDY). These messages are normally sent by the controller to determine if devices have a need for service. Note that bit CØ of both DOE and IDY messages may be set by devices to indicate a need for service.

Detailed definitions of all local and remote messages are given in appendix B. Information regarding coding of remote messages can be found in section 2-18 as well as appendix C.

2.2 State Diagram Notation

Figure 2-2 shows the notation used throughout this chapter within the state diagrams. Each state that an interface function can assume is represented by a four letter upper-case mnemonic always ending in S within a circle. All permissible transitions between states are represented by arrows qualified by expressions that evaluate either true or false. The interface function shall remain in its current state if all expressions which qualify transitions leading to other states are false. The interface function shall enter the state pointed to if and only if the associated expression becomes true.

Notation	Meaning	Notation	Meaning
(TACS)	interface function state	expression →	transition between states
ACDS	linkage from other interface functions	+	OR logical operator
PPU	remote messages from the interface	•	AND logical operator
rdy	local messages from the device	$\overline{\textbf{SPAS}}$	NOT logical operator
sync	pseudomessages from the encoding/ decoding logic	[+ SDC • **LACS**]	optional expression term (at the designer's choice)

Figure 2-2. State Diagram Notation

An expression consists of one or more local messages, pseudomessages, remote messages, or state linkages used in conjunction with the operators AND, OR, or NOT. Local messages to an interface function are represented by lower-case three-letter mnemonics. Pseudomessages are represented by lower-case four-letter mnemonics. Remote messages are represented by upper-case three-letter mnemonics. A linkage from another state diagram is represented by the state mnemonic in boldface type. A state linkage is true if the state is currently active; otherwise, it is false. The AND operator is represented by a dot. The OR operator is represented by a plus sign. The NOT operator is represented by a horizontal bar placed over the portion of the expression to be negated. The resulting negated expression has a true value if and only if the value of the expression under the bar is false. The AND operator takes precedence over the OR operator in an expression unless indicated otherwise by parentheses.

If a portion of an expression is optional in that its true value is not required for the complete expression to be true (at the designer's choice), then it is enclosed within square brackets. If a specific expression causes a transition to a state from all other possible states of the diagram, an arrow without a state at its origin is used, and is assumed to originate in all states. Note also that the power-off state is a valid state of most interface functions and should normally be shown with a transition leading to the state to be entered when power is first turned on. This state is omitted for clarity in the diagram and only the transition to the first state is shown.

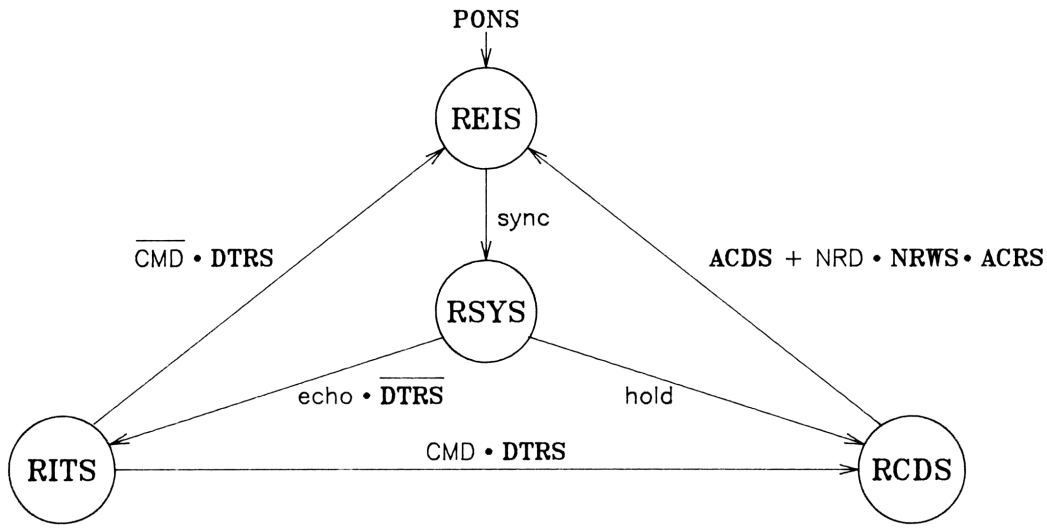
2.3 R (Receiver) Function

The R interface function provides the device with the capability to receive messages over the interface. It categorizes the received message into major classifications which are used by the AH (Acceptor Handshake) and D (Driver) interface functions to effect proper communication.

The R function receives the remote messages one bit at a time and decides as quickly as possible if immediate retransmission is required. If so, the message is passed directly to the D interface function for retransmission. The R function also decides whether the message is to be interpreted by the device or other interface functions or is to be ignored. If the device is to interpret the message, it is passed to the AH function. If no device or interface interpretation is required, the frame is passed to the D function. When the D or AH function signals that it has accepted the message, the R function returns to its idle state ready to receive the next remote message.

While it is possible to simplify the implementation of the R function by waiting for all message bits to be available before performing any decoding, the resulting throughput would be unacceptable. It is strongly recommended that the decoding begin as soon as enough bits have been received to permit correct decoding. This occurs on the first bit for DOE messages, the second bit for IDY, the third bit for CMD, etc. In general, only a few bits need be received to decide if retransmission is necessary.

Please note that all devices must implement the full R interface function capability. No subsets are permitted.



$$
\begin{aligned}
\text{echo} = {} & \text{DOE} \bullet \overline{\text{TACS}} \bullet \overline{\text{SPAS}} \bullet \overline{\text{DIAS}} \bullet \overline{\text{AIAS}} \bullet \overline{\text{TAHS}} \bullet \overline{\text{TERS}} \bullet \overline{\text{LACS}} \bullet \overline{\text{NRWS}} \bullet \overline{\text{CACS}} \\
& + (\text{CMD}+\text{IDY}) \bullet \overline{\text{CACS}} \bullet \overline{\text{CSBS}} \\
& + \text{ARG} \bullet \overline{\text{TADS}} \bullet \overline{\text{TACS}} \bullet \overline{\text{SPAS}} \bullet \overline{\text{DIAS}} \bullet \overline{\text{AIAS}} \bullet \overline{\text{TAHS}} \bullet \overline{\text{TERS}} \bullet \overline{\text{TPAS}} \bullet \overline{\text{LACS}} \bullet \overline{\text{CACS}} \bullet \overline{\text{CSBS}} \\
& + (\text{AAD}+\text{AMP}) \bullet \overline{\text{AAUS}} + \text{AEP} \bullet \overline{\text{AWPS}} + \text{AES} \bullet \overline{\text{ZES}} \bullet \overline{\text{AWSS}} + \text{AES} \bullet \overline{\text{AAUS}} \bullet \overline{\text{AWSS}} \\
& + \text{IAA} + \text{IEP} + \text{IMP} + \text{IES}
\end{aligned}
$$

$$
\begin{aligned}
\text{hold} = {} & \text{DOE} \bullet (\ \text{TACS}+\text{SPAS}+\text{DIAS}+\text{AIAS}+\text{TAHS}+\text{TERS}+\text{LACS}+\text{NRWS}+\text{CACS}\ ) \\
& + \text{RDY} \bullet \overline{\text{ARG}} \bullet \overline{\text{AAG}} + (\text{CMD}+\text{IDY}) \bullet (\ \text{CACS}+\text{CSBS}\ ) \\
& + \text{ARG} \bullet (\ \text{TADS}+\text{TACS}+\text{SPAS}+\text{DIAS}+\text{AIAS}+\text{TAHS}+\text{TERS}+\text{TPAS}+\text{LACS}+\text{CACS}+\text{CSBS}\ ) \\
& + (\text{AAD}+\text{AES}+\text{AMP}) \bullet \text{AAUS} + \text{AEP} \bullet \text{AWPS} + \text{ZES} \bullet \text{AWSS}
\end{aligned}
$$

Messages

sync	valid sync bit received
AAD	auto address
AAG	auto address group
AEP	auto extended primary
AES	auto extended secondary
AMP	auto multiple primary
ARG	addressed ready group
CMD	command
DOE	data or end
IAA	illegal auto address
IDY	identify
IEP	illegal extended primary
IES	illegal extended secondary
IMP	illegal multiple primary
NRD	not ready for data
RDY	ready
ZES	zero extended secondary

Interface States

RCDS	receiver data state (links to AH,C,L)
REIS	receiver idle state
RITS	receiver immediate transfer state (links to D,C,SR,PP)
RSYS	receiver sync state
AAUS	auto address unconfigured state (from AA)
ACDS	acceptor data state (from AH)
ACRS	acceptor ready state (from AH)
AIAS	accessory ID active state (from T)
AWPS	auto wait for primary state (from AA)
AWSS	auto wait for secondary state (from AA)
CACS	controller active state (from C)
CSBS	controller standby state (from C)
DIAS	device ID active state (from T)
DTRS	driver transfer state (from D)
LACS	listener active state (from L)
NRWS	not ready wait state (from L)
PONS	power on state (from PD)
SPAS	serial poll active state (from T)
TACS	talker active state (from T)
TADS	talker addressed state (from T)
TAHS	talker hold state (from T)
TERS	talker error state (from T)
TPAS	talker primary active state (from T)

Figure 2-3. R State Diagram

REIS (Receiver Idle State)

In REIS the R interface function is not engaged in any message transfer process. The function powers on in this state. Transition out of this state does not occur until a valid sync bit has been received from the loop.

RSYS (Receiver Sync State)

In this state the R interface function has begun to receive a message from the interface and is in the process of determining whether to immediately retransmit the message (echo) or not (hold).

RITS (Receiver Immediate Transfer State)

DOE messages not sourced by or intended for this device, CMD or IDY messages not sourced by this device, and ARG messages not for this device should be immediately retransmitted on the loop. In RITS the R function is signaling the D (Driver) function that retransmission of the incoming frame should commence. RITS is not entered until the D function is finished transmitting any previous message and is not exited until the D function has accepted the current message and commenced retransmission. Most messages which are echoed do not affect the device or interface functions (except R and D), however CMD messages are both echoed and held for device and interface interpretation. This decoding is also done in RITS. If the message is not a CMD the function returns to REIS, otherwise the transition is to RCDS.

RCDS (Receiver Data State)

In this state the R function is signaling the AH (Acceptor Handshake) function that the message being received is for this device. When the AH function signals in return that it has accepted the message, the R function returns to REIS.

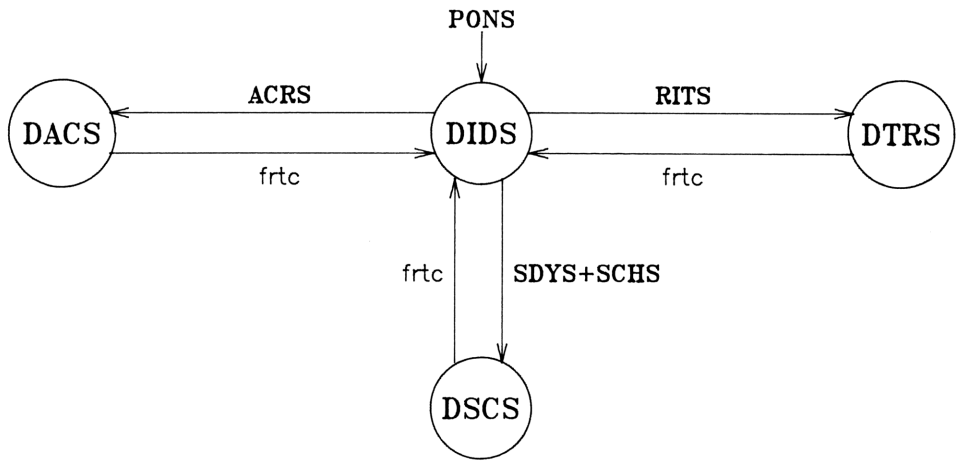
2.4 D (Driver) Function

The D interface function provides the device with the ability to transmit messages on the loop. These messages come from three different sources: messages generated by this device, messages received on the loop which must be immediately retransmitted, and messages received on the loop which must be retransmitted after acceptance by the device.

When the R (Receiver), SH (Source Handshake), or AH (Acceptor Handshake) functions indicate that a message is available for transmission, the D function begins the transmission. When the frame transmission is complete, the D function again becomes idle.

The D function transmits all remote messages. These messages are transferred to it by the R, AH, or SH functions. In this transfer certain modifications of the message may be performed by the parallel poll and service request interface functions. The SR function allows the device to set the SRQ bit on DOE or IDY frames before the Driver transmits them on the loop. The PP function allows the device to modify one bit of IDY frames before retransmission for parallel poll response. Refer to these two functions for a full description of this modification process.

All devices must have the full D function implemented. No subsets are permitted.



Messages

frtc frame transmission complete

Interface States

DACS driver transmit from acceptor state (links to AH,L)
DIDS driver idle state (links to SR,PD)
DSCS driver transmit from source state (links to SH)
DTRS driver transfer state (links to R,PP)
ACRS acceptor ready state (from AH)
PONS power on state (from PD)
RITS receiver immediate transfer state (from R)
SCHS source command handshake state (from SH)
SDYS source delay state (from SH)

Figure 2-4. D State Diagram

DIDS (Driver Idle State)

In DIDS the D function is not transmitting any message. The function powers on in this state. When the SH (source handshake), AH (acceptor handshake), or R (receiver) functions indicate that a frame is ready for transmission, the D function enters DSCS, DACS, or DTRS respectively.

DSCS (Driver Source State)

In this state the function is transmitting a message frame originated by this device. DSCS is entered when the SH (Source Handshake) function indicates that the message is ready to be transmitted. When transmission is complete the function returns to DIDS.

DTRS (Driver Transfer State)

In DTRS the D function has commenced retransmitting a message transferred from the R (Receiver) function which must not be held up by the device's ability to interpret it. When finished with the retransmission the D function returns to DIDS.

DACS (Driver Acceptor State)

This state is entered when retransmission of a message which has been received and interpreted by the device has begun. The message is transferred from the AH (Acceptor Handshake) function. When transmission is complete the function transitions to DIDS.

2.5 AH (Acceptor Handshake) Function

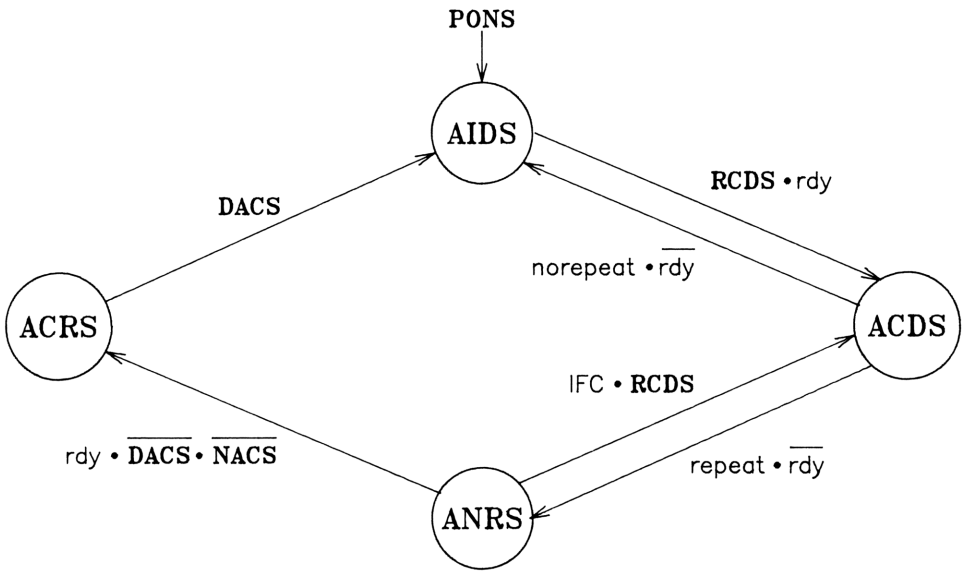
The AH interface function provides the device with the capability to interpret remote messages. It also determines if the frame needs to be retransmitted after interpretation (repeat) or not (norepeat), and indicates to the D (Driver) interface function when this retransmission should begin.

If the Receiver interface function signals that a frame has been received and is intended for this device, the Acceptor Handshake function indicates to the device and to the other interface functions when that frame is ready for interpretation. If the message is to be retransmitted, the transmission begins after the device indicates that it has finished interpreting the message (with the local message rdy). The AH function also controls the NRD (not ready for data) sequence by holding the data while the device sources the NRD message and retransmitting the data when the NRD returns.

The AH interface function is not capable of originating remote messages. It only accepts received messages from the R (Receiver) interface function and transfers them to the device and other interface functions for appropriate action. After the device indicates that interpretation is complete, the AH function can transfer the received message to the Driver interface function for retransmission.

The AH function may be implemented such that it receives all eleven bits of the message frame prior to any decoding (repeat, norepeat expressions). It is permitted, however, to perform this decoding earlier in the frame so that loop throughput may be considerably enhanced at the expense of greater complexity.

Please note that all devices must implement the full AH capability. No subsets are permitted.



$$
\begin{aligned}
\text{repeat} = {} & \text{DOE} \bullet (\ \mathbf{LACS} + \mathbf{NRWS} + \mathbf{CACS}\ ) \\
& + \text{RDY} \bullet \overline{\text{SOT}} \bullet \overline{\text{AAG}} \bullet \overline{\mathbf{CACS}} \bullet \overline{\mathbf{CSBS}} \\
& + \text{SOT} \bullet \overline{\mathbf{TADS}} \bullet \overline{\mathbf{TACS}} \bullet \overline{\mathbf{SPAS}} \bullet \overline{\mathbf{DIAS}} \bullet \overline{\mathbf{AIAS}} \bullet \overline{\mathbf{TAHS}} \bullet \overline{\mathbf{TERS}} \\
& + \text{NRD} \bullet \overline{\mathbf{CACS}}
\end{aligned}
$$

$$
\begin{aligned}
\text{norepeat} = {} & \text{DOE} \bullet (\ \mathbf{TACS} + \mathbf{SPAS} + \mathbf{DIAS} + \mathbf{AIAS} + \mathbf{TAHS} + \mathbf{TERS}\ ) \\
& + \text{CMD} + \text{IDY} + \text{AAG} \\
& + \text{RDY} \bullet \overline{\text{NRD}} \bullet (\ \mathbf{CACS} + \mathbf{CSBS}\ ) \\
& + \text{SOT} \bullet (\ \mathbf{TADS} + \mathbf{TACS} + \mathbf{SPAS} + \mathbf{DIAS} + \mathbf{AIAS} + \mathbf{TAHS} + \mathbf{TERS}\ ) \\
& + \text{NRD} \bullet \mathbf{CACS}
\end{aligned}
$$

Messages

rdy	ready for next frame
AAG	auto address group
CMD	command
DOE	data or end
IDY	identify
IFC	interface clear
NRD	not ready for data
RDY	ready
SOT	start of transmission

Interface States

ACDS	acceptor data state (links to R,SH,C,T,L, SR,RL,AA,PD,PP,DC,DT,DD)
ACRS	acceptor ready state (links to R,D)
AIDS	acceptor idle state (links to PD)
ANRS	acceptor not ready state
AIAS	accessory ID active state (from T)
CACS	controller active state (from C)
CSBS	controller standby state (from C)
DACS	driver transmit from acceptor state (from D)
DIAS	device ID active state (from T)
LACS	listener active state (from L)
NACS	not ready active state (from L)
PONS	power on state (from PD)
RCDS	receiver data state (from R)
SPAS	serial poll active state (from T)
TACS	talker active state (from T)
TADS	talker addressed state (from T)
TAHS	talker hold state (from T)
TERS	talker error state (from T)

Figure 2-5. AH State Diagram

AIDS (Acceptor Idle State)

In AIDS the AH function is not engaged in the handshake cycle and does not have a message available. This is the power-on state for this function. When the R (Receiver) function indicates that a message for this device is available, the AH function enters ACDS.

ACDS (Acceptor Data State)

In this state the interface function is indicating to all other interface and device functions that a message for this device has been received and is now valid for interpretation. This state also decodes the message to determine whether it should be retransmitted or not. DOE, CMD, RDY or IDY messages sourced by this device, and SOT messages intended for this device are not retransmitted on the loop. After the device has indicated that it has accepted the message for interpretation by sending the rdy local message false (not ready), the AH function returns to AIDS if retransmission is not necessary, or transitions to ANRS if the message must be retransmitted.

ANRS (Acceptor Not Ready State)

In ANRS the AH function is waiting for the device to indicate (via rdy) that it is ready for the next message frame. This typically occurs after the device has completed any actions necessitated by the current message. When the device is ready the function transitions to ACRS. If the R function indicates that the IFC message has been received while the AH function is waiting in ANRS, the function transitions back to ACDS.

ACRS (Acceptor Ready State)

This state is indicating to the D (Driver) function that retransmission of the current message frame should begin. When the D function indicates in return that it has accepted the frame and has commenced transmission, the AH function returns to AIDS.

2.6 SH (Source Handshake) Function

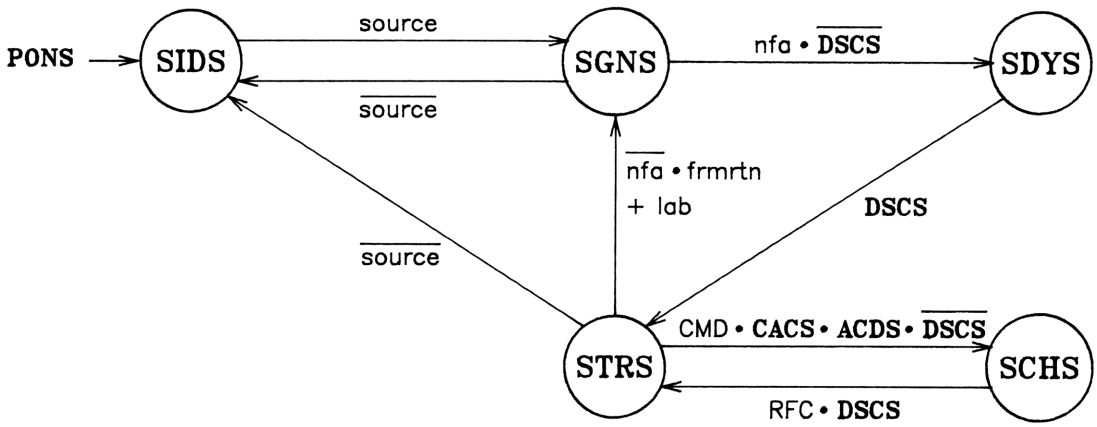
The SH interface function gives the device the capability to originate remote messages primarily while the device is a talker or controller. It coordinates the message transfer from the device to the interface.

For normal loop handshake, the SH interface function accepts a message from the device and transfers it to the D interface function. When the R function indicates that the transmitted frame has returned, the SH function completes the handshake with the device and becomes ready to accept another message. For sourcing commands, the SH function causes the RFC (ready for command) message to be sent after every command is sent and received. The SH function also controls asynchronous handshaking with the local message lab (local abort).

The SH function becomes ready to accept messages from the device when the device is first enabled to source remote messages (source expression). If the SH function is ready and the device has a message available, message transmission begins and the function waits for the message to return. When the R function indicates that the frame has returned, the SH function again becomes ready to source another frame. As soon as the device is no longer enabled to source messages, the source handshake function returns to idle.

The SH function does not originate remote messages on the loop but merely controls the orderly transfer of messages from the device to the D (Driver) function for transmission. The device may change the message frame while SIDS or SGNS are active, but must hold the frame valid while in SDYS and STRS.

All devices must have the full SH function. No subsets are permitted.



source = **TACS** + **SPAS** + **DIAS** + **AIAS** + **TAHS** + **TERS** + **CACS**
+ **NACS** + **ARSS** + **AAIS** + **ASIS** + **APIS** + **AMIS**

frmrtn = DOE • (**TACS** + **SPAS** + **DIAS** + **AIAS** + **TAHS** + **TERS**) • **ACDS**
+ (RDY+IDY) • **CACS** • **ACDS**

Messages

lab	local abort
nfa	new frame available
CMD	command
DOE	data or end
IDY	identify
RDY	ready
RFC	ready for command

Interface States

SCHS	source command handshake state (links to D)
SDYS	source delay state (links to D)
SGNS	source generate state
SIDS	source idle state
STRS	source transfer state (links to C,T,SR,AA)
AAIS	auto address increment state (from AA)
AIAS	accessory ID active state (from T)
ACDS	acceptor data state (from AH)
AMIS	auto multiple increment state (from AA)
APIS	auto primary increment state (from AA)
ARSS	asynchronous request source state (from SR)
ASIS	auto secondary increment state (from AA)
CACS	controller active state (from C)
DIAS	device ID active state (from T)
DSCS	driver transmit from source state (from D)
NACS	not ready active state (from L)
PONS	power on state (from PD)
SPAS	serial poll active state (from T)
TACS	talker active state (from T)
TAHS	talker hold state (from T)
TERS	talker error state (from T)

Figure 2-6. SH State Diagram

SIDS (Source Idle State)

The SH function powers on in SIDS and in this state no handshake is taking place. The device is not sourcing frames. When the appropriate interface functions indicate that the device should begin to source messages (source expression) the function transitions to SGNS.

SGNS (Source Generate State)

In SGNS the device is in the process of generating the next message frame to be sourced. When the device indicates that the new message is available (nfa local message) and the D (Driver) function indicates that it is not still transmitting a previously sourced frame, the function enters SDYS. When the sourcing device is no longer enabled to source messages, the function will return to SIDS.

SDYS (Source Delay State)

In this state the SH function is waiting for the D (Driver) interface function to acknowledge receipt of the sourced message and begin transmission. When this occurs the function enters STRS.

STRS (Source Transfer State)

In STRS the SH function is waiting for the sourced message to go around the loop and return (normal loop handshake, frmrtn expression). When this occurs the function returns to SGNS if the sourced message was not a command and transitions to SCHS if the message was a command. Asynchronous loop operations are possible using the lab local message to force the return to SGNS before the handshake is complete so a new message frame may be sent. Normally, the last frame sourced by a device will be a Ready frame (such as EOT if talker) and will cause a transition directly from STRS to SIDS since the frame might not return to the device.

SCHS (Source Command Handshake State)

The function enters SCHS when the receiver function indicates that the sourced command has returned. In this state, the function is signaling to the driver function that the RFC (ready for command) message is ready for transmission. When the driver function indicates that the RFC transmission has begun, the function returns to STRS to wait for the RFC to return.

2.7 C (Controller) Function

The C function provides a device with the ability to send interface messages (including CMD, RDY, and IDY messages) to devices on the loop. It also provides the capability for conducting parallel polls, detecting the SRQ message from devices requesting service, and detecting transmission errors in device dependent data transmissions. These capabilities exist only while the controller function is in its active state.

If more than one device that possesses the C function is on the loop, then all but one must be in the idle state at any given time. The device with the active C function is called the active controller. Protocol is provided to allow controllers to take turns as active controller of the loop.

One and only one controller device may send the local message scl true indicating that it is the system controller. This condition should remain true throughout the operation of the interface. The system controller takes charge at power on and is the only device which can source the IFC message. It can do this at any time regardless whether it is the active controller.

When this function is in its active state, the device is enabled to originate interface messages including CMD, RDY, and IDY message frames under control of the SH (Source Handshake) function. These interface messages may be used to control the interface functions within other devices on the loop. When the C function is not in its active state, the device may not source interface messages. The C function does not give the device the capability to source or accept DOE (data or end) messages. Those capabilities are offered only by the T and L functions.

The normal loop handshake consists of sourcing one frame at a time and waiting for its return while each receiving device holds the frame until it is ready for the next. When a frame returns, all devices on the loop are ready to receive the next message. For frames intended for only a few devices, the penalty of such serial interpretation is not severe. However, since CMD frames are often intended for all devices, they are retransmitted immediately by each receiving device to allow for parallel interpretation. When a CMD frame returns, the active controller sources the RFC ready frame which is held by each device until it is ready for the next message. Refer to the SH function for more information about both types of loop handshake.

It is strongly recommended that the controller error check its messages by comparing the received frame with the previously transmitted frame to guarantee proper reception of messages and to enhance reliability.

The active controller may source the IDY message asynchronously to the loop handshake at any time but will not normally do so unless there has been a long delay in loop transmission. All devices should be implemented so that they will respond properly to the IDY and yet still continue correct response to other messages without disturbing normal loop operation. The controller may use the local message gta to enable the asynchronous IDY if it is not already in the active state. Refer to chapter 5 for complete discussion of asynchronous operations.

Devices on the loop indicate a need for service from the controller by setting the SRQ bit in DOE or IDY frames. When rapid response is important, the controller may execute a parallel poll by sending the IDY message. When configured to do so, devices respond by modifying one of the assigned eight data bits in the IDY frame. Devices may also have the capability of asynchronously sourcing their own IDY frames with the SRQ bit set. The controller may choose to enable this mode when no loop operations are in progress.

For various reasons the controller may choose to interrupt a device dependent message transmission between a talker and listener(s). This may be done through the L (listener) interface function. The controller may also enable active listeners to perform this NRD sequence so that the transfer may be halted before extensive delays occur. Refer to section 2.9 for further information about this capability.

The system controller is the only device on the loop which is permitted to source the IFC message. It may do this at any time (even if it is not the currently active controller) and take control of the loop asynchronously. Under this circumstance it is possible to have an extraneous frame on the loop since the source of the frame may be cleared before the frame returns. To prevent this frame from recirculating, the system controller, after sourcing the IFC, must destroy any frames which it receives until the IFC returns. Further, no device may source any frame after retransmitting the IFC (and RFC) until specifically commanded to do so by succeeding interface messages. The sic or gta local messages may initiate this sequence depending on the state of the system controller.

When power is first applied the system controller should observe special protocol to insure that all devices on the loop are present and properly connected. The recommended sequence is to send the IFC message at regular intervals while stopping all frames received to guarantee that no extra frames are circulating. When one IFC returns, loop continuity is verified. At that time, the controller should send one and only one RFC while stopping all frames being received. When the RFC returns, the loop is ready for normal operations. Refer to chapter 5 for more information on the power up sequence.

No controller capability is designated CØ. If the device has the controller capability, it must include subset C1. The allowable subsets of the controller function are:

C1: Basic controller capability. Includes the ability to send interface messages and detect transmission errors by interpreting the ETO or ETE messages. Requires CIDS, CACS, CSBS, CEIS, and CEMS.

C2: System controller capability. Provides the capability to send the Interface Clear message. If this capability is not present the local messages sic and scl must always be false. It is strongly recommended that the scl message not always be true so that the device may be used on a loop with multiple controllers.

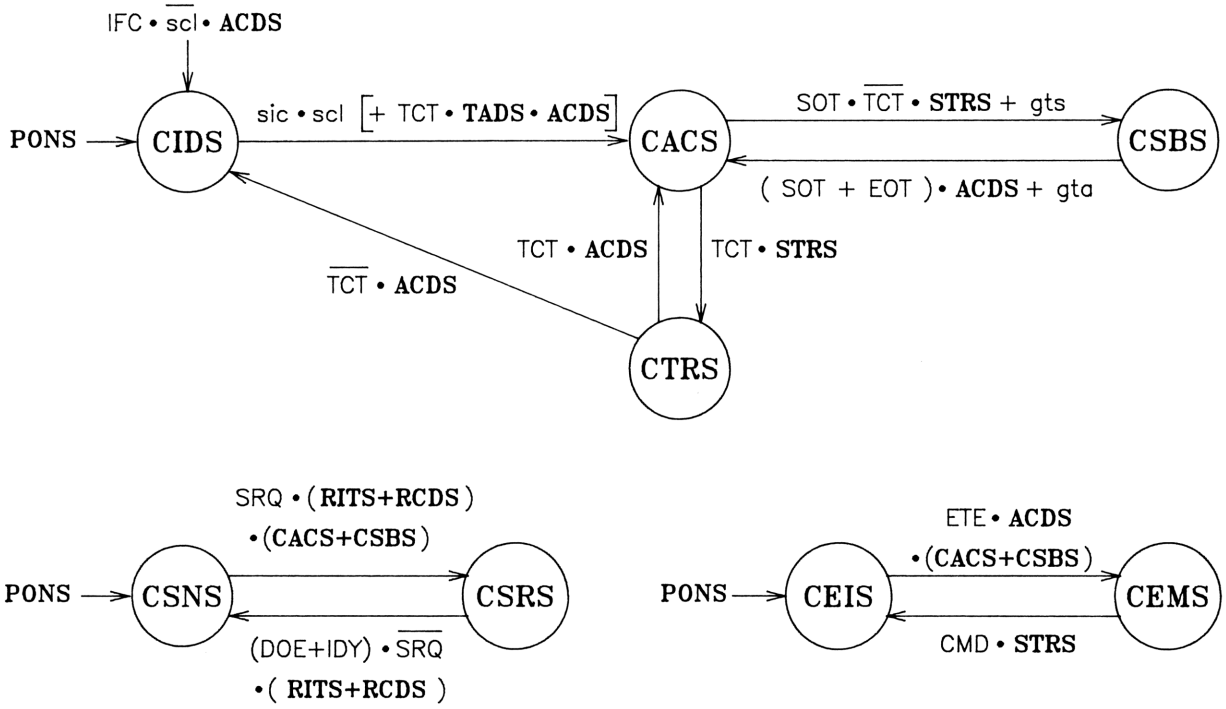
C3: Detect SRQ capability. Note that the L function is necessary if the device is to receive and interpret status bytes. Requires CSNS and CSRS.

C4: Pass and receive control capability. This allows controllers to take turns controlling the loop. Requires CTRS and the optional term in the transition between CIDS and CACS. The T function is also required.

C5: Parallel poll capability. Includes the ability to conduct parallel polls and to enable device response.

C6: Asynchronous service request capability. Includes the ability to enable devices to source asynchronous IDY messages and to interpret those messages.

All controllers will include C1 with, optionally, some combination of the other capabilities. A simple system controller with only SRQ capability would be C1,2,3. A basic controller with parallel poll and control passing capability would be designated C1,4,5.



Messages

gta	go to active
gts	go to standby
scl	system control
sic	send interface clear
CMD	command
EOT	end of transmission
ETE	end of transmission with error
IFC	interface clear
SOT	start of transmission
SRQ	service request
TCT	take control

Interface States

CACS	controller active state (links to R,AH, SH,T,L,PD)
CEIS	controller error idle state
CEMS	controller error mode state
CIDS	controller idle state
CSBS	controller standby state (links to R,AH,L)
CSNS	controller service not requested state
CSRS	controller service requested state
CTRS	controller transfer state
ACDS	acceptor data state (from AH)
PONS	power on state (from PD)
RCDS	receiver data state (from R)
RITS	receiver immediate transfer state (from R)
STRS	source transfer state (from SH)
TADS	talker addressed state (from T)

Figure 2-7. C State Diagram

CIDS (Controller Idle State)

In CIDS the device has no loop control capability. This is the power-on state. If the function receives the TCT message while addressed to talk or the device is system controller and has need of sending the IFC (interface clear) message (local messages scl and sic), then the function enters CACS.

CACS (Controller Active State)

In CACS the device is enabled to send interface messages through the SH (source handshake) function. If the function enables a talker to begin sourcing DOE (data or end) messages or the local message gts (go to standby) is true, the function enters CSBS. If the function sends the TCT (take control) message in an attempt to pass active control to another device on the loop, the function enters CTRS.

CSBS (Controller Standby State)

In CSBS the function has enabled a data transfer between a talker and listener(s) with an SOT (start of transmission) message. If the SOT returns or an EOT (end of transmission) is received, the function returns to CACS. While in CSBS the device may not source any remote messages unless it is the active talker. The function may return active with the gta (go to active) local message if asynchronous operations are desired.

CTRS (Controller Transfer State)

In CTRS the device has sent the TCT in an attempt to pass active control to another loop device. If the TCT returns, the attempt is unsuccessful and the device must resume active control of the loop. If any other frame is received, the attempt was successful and this function must enter CIDS.

CSNS (Controller Service Not Requested State)

In this state the function is indicating that no loop devices are currently requesting service. This is the power-on state. If a DOE (data or end) or IDY (identify) message is received with the SRQ bit true, the function enters CSRS.

CSRS (Controller Service Request State)

In CSRS the function is indicating that at least one loop device is requesting service. If a DOE or IDY frame is received with the SRQ bit false, then the function returns to CSNS.

CEIS (Controller Error Idle State)

In this state the function is indicating to the device that no error has been detected by the talker in a data transfer. If the ETE message is received the function moves to CEMS.

CEMS (Controller Error Mode State)

In CEMS the function is indicating to the device that an talker has discovered an error in a data transfer. When the controller sends its next command the function returns to CEIS.

2.8 T (Talker) Function

The T interface function provides the device with the capability to send device dependent data, status, device ID, and accessory ID on the loop. This capability exists only when the device has been addressed to talk.

This function enables the device to send device dependent DOE (data or end) messages on the loop under the control of the SH (Source Handshake) interface function. The device may send data messages only if one of TACS, SPAS, DIAS, or AIAS is active.

To enhance device compatibility it is strongly recommended that wherever possible device dependent data messages be sent using the ASCII code.

The talker may send the END message within the transmitted data string to indicate an end-of-record condition to the listener(s) without terminating the transmission. The EOT (end of transmission) message should be sent to the controller when the talker has no more data ready to send. The EOT message does not necessarily indicate an end of message condition, it is used only as a handshake to the controller. Therefore, if a delay may occur before the talker is able to source another data byte, ETO should be sent.

The controller or an active listener (if enabled by the controller) may elect to interrupt the data transfer using an NRD sequence as controlled by the L (Listener) and AH (Acceptor Handshake) interface functions. The talker's current data byte is held by the device interrupting the transfer and the NRD message is inserted in its place. When the talker receives and retransmits the NRD frame, it waits for the held data byte to be returned. When the interrupting device receives its NRD, then the held data byte is retransmitted to the talker. When the talker receives the data byte, it will source the appropriate EOT (end of transmission) message. Refer to the L interface function in section 2.9 for a more complete description.

Each time a talker becomes enabled to source status or ID information with the SST, SAI, or SDI ready messages, it must begin with the first byte regardless of any previous operations. However, if a talker is enabled to source data with an SDA ready message, the transmission must begin with the current data byte.

A talker must have the capability to be interrupted one or more times during a data transfer without adverse effects. During this interruption, the controller may ask devices on the loop for status or ID information one or more times, or may simply start a transfer between other devices. The talker must be able to be interrupted in this manner without affecting the data to be sourced. When the talker is re-enabled to source data, it must continue at the point of interruption unless directed otherwise by the controller in a device dependent manner.

A device's status response may contain one or more bytes that indicate various device dependent conditions. The two most significant bits of the first byte are defined and may not be used for any other purpose. Bit 7 (msb) is set if the byte is a system status byte and clear otherwise. Bit 6 is always equal to the value of the local message rsv. Refer to the SR (Service Request) interface function on page 2-36 and section 2.18 for information on status message coding.

The device ID is an ASCII string representing the the device's model identification followed by carriage return and linefeed. It is typically of the following form: two alpha characters representing the company code, a one to five digit model number, and a single alpha character model revision. Devices may optionally send additional information (before the carriage return) to indicate such things as options installed in the device or other information to identify or classify the device capabilities.

The accessory ID consists of one or more bytes that provide information about device functions and capabilities. Refer to section 2.18 for a complete description of the accessory ID byte(s).

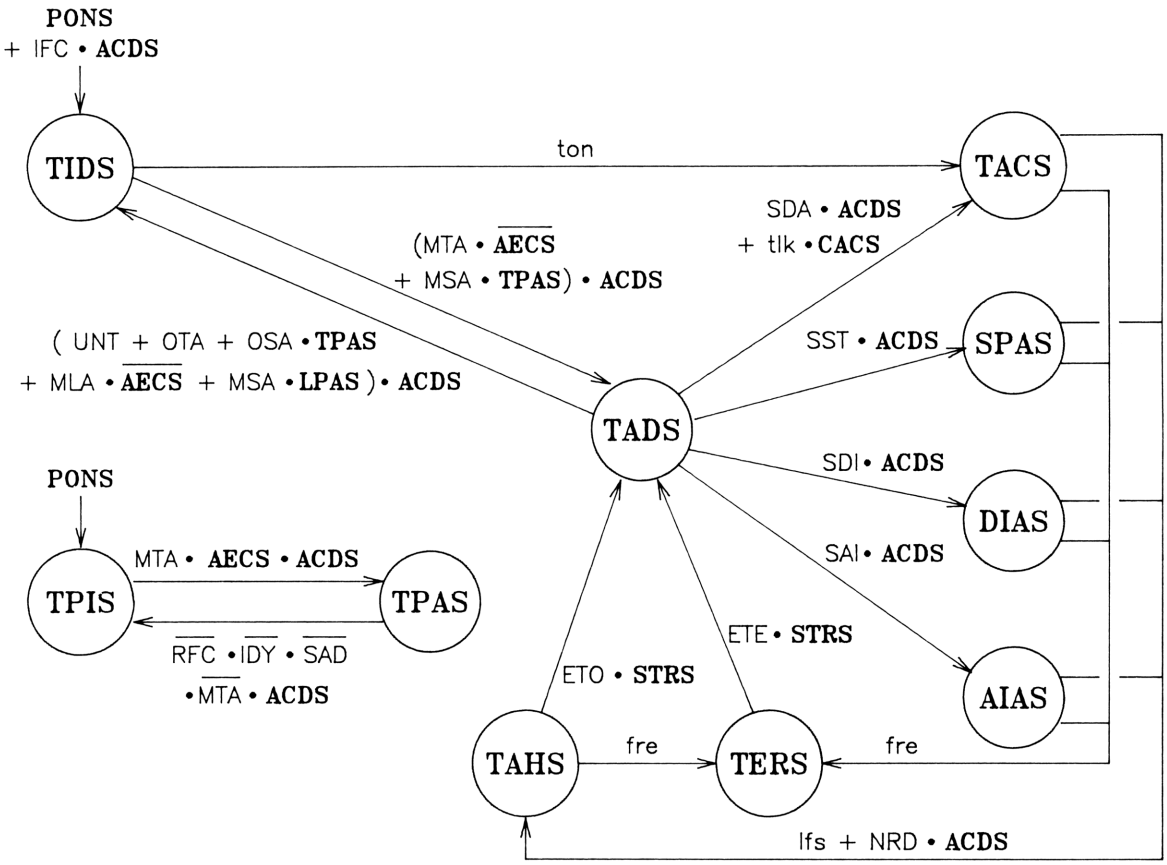
When a DOE message returns to the talker the SRQ bit may be set. If the talker is not also the controller, the SRQ message must be sent to the active controller. The talker may do this by sourcing the following DOE message (if there is one) with the SRQ bit set. The talker must not allow more than one DOE frame to be sourced without the SRQ bit set after receiving a DOE with the SRQ bit set. If the SRQ bit on any received DOE message is received false, the talker must discontinue sourcing the SRQ message with its device dependent data.

The loop structure is ideal for error checking purposes since the originator of a message can compare the returned frame with the original transmitted frame. While not specifically required it is strongly recommended that devices perform this error checking function to enhance loop reliability. If a device detects an error it should set the local message fre true. The T function will then complete the current handshake and source the ETE (end of transmission with error) message to signal the controller that a transmission error has occured. If the device does not perform error checking, it must always send the fre local message false.

As with other interface functions, a complete understanding of the four frame handling functions (Receiver, Acceptor Handshake, Source Handshake, and Driver) is essential for correct implementation of the Talker function.

A talk-only listen-only system has no controller and involves some special protocol. The talker in such a system must not send the EOT message (since it would circulate endlessly) and therefore must always send the local messages lfs and fre false. Furthermore, after power-on, if the talker has not received back its first message frame after a certain amount of time (say, one second) it should continue to send this message at this slow, device dependent interval until the frame returns completing the loop handshake. Then the talker may begin normal transmission. Without this provision the first frame might be destroyed at an as yet unpowered listener and the talker would never send the next frame. This permits devices in this type of system to be powered on in any order.

If the talker function is not implemented, the device is designated TØ. The designer may choose among several possible subsets of the T function, all of which (except TØ) must include TIDS, TADS, TAHS, TERS. The data, status, accessory ID, and device ID capabilities are designated by the numbers 1, 2, 3, and 4 respectively. It is very strongly recommended that devices respond to both accessory ID and device ID. Talk-only mode (designated T5) requires subset T1 and allows the ton local message to be true. Some means of disabling the ton local message must be provided. Extended addressing may be added with the TPIS and TPAS states (requires the AA2 or AA3 subsets of the auto address function). For example, a device which can send data and status only would be given the T1,2 designation. The same device with the inclusion of talk-only mode would be listed as T1,2,5. A device that can send only status and its accessory ID and which has extended addressing capability would be T2,4,6.



Messages

tlk	local talk	MTA	my talk address
ton	talk only	NRD	not ready for data
lfs	last frame sent	OSA	other secondary address
fre	frame error detected	OTA	other talk address
		RFC	ready for command
ETE	end of transmission with error	SAD	secondary address
ETO	end of transmission, OK	SAI	send accessory ID
IDY	identify	SDA	send data
IFC	interface clear	SDI	send device ID
MLA	my listen address	SST	send status
MSA	my secondary address	UNT	untalk

Interface States

AIAS	accessory ID active state (links to R,AH,SH)	**ACDS**	acceptor data state (from AH)
DIAS	device ID active state (links to R,AH,SH)	**AECS**	auto extended configured state (from AA)
SPAS	serial poll active state (links to R,AH,SH,SR)	**CACS**	controller active state (from C)
TACS	talker active state (links to R,AH,SH)	**LPAS**	listener primary active state (from L)
TADS	talker addressed state (links to R,AH,C,DD)	**PONS**	power on state (from PD)
TAHS	talker hold state (links to R,AH,SH)	**STRS**	source transfer state (from SH)
TERS	talker error state (links to R,AH,SH)		
TIDS	talker idle state		
TPAS	talker primary addressed state (links to R,L)		
TPIS	talker primary idle state		

Figure 2-8. T State Diagram

TIDS (Talker Idle State)

In this state the device is not able to send data messages. The function powers on in this state. A talk-only device may use the local message ton to enter TACS directly from TIDS without first being addressed. If the MTA (my talk address) message is received or the MTA and MSA (my secondary address) are received, and the device is configured for extended addressing, then the function enters TADS.

TADS (Talker Addressed State)

In TADS the T function has received its talk address (and secondary address if extended addressing is used). The function is prepared for, but not yet engaged in the sending of device dependent messages over the loop. When the appropriate start of transmission (SOT) message is received the function enters one of the active states to begin sending the device dependent messages. Controllers may use the local message tlk to enter TACS without receiving the SDA (send data) message. If a CMD which causes the function to become unaddressed is received while in TADS the function returns to TIDS.

TACS (Talker Active State)

In this state the SDA (send data) message has been received and the device is enabled to begin sourcing device dependent data messages (DOE). These transmissions are controlled by the SH (Source Handshake) function. When an NRD (not ready for data) ready message is received or the device detects that no more data is ready to be sent, the function enters TAHS. If the device detects that a DOE frame has been received in error, then the device enters TERS. If an IFC is received the function immediately aborts back to TIDS.

SPAS (Serial Poll Active State)

In this state the SST message has been received and the device is enabled to replace this message with one or more bytes of device status information. Transition to TAHS occurs when the status is sent or when an NRD is received. Transition to TERS occurs if an error is detected. IFC causes an immediate abort to TIDS.

DIAS (Device Identify Active State)

In this state the function has received the SDI message and the device is enabled to replace this message with its device ID string followed by a transition to either TAHS or TERS.

AIAS (Accessory Identify Active State)

In AIAS the SAI frame has been received and the device is enabled to replace it with one or more bytes of accessory ID information followed by a transition to either TAHS or TERS.

TAHS (Talker Hold State)

In talker hold state the device has completed sourcing its device dependent data and is waiting for the SH (source handshake) function to indicate that the current handshake cycle is complete. If an error is detected then the function enters TERS. If the device does not indicate that an error has occurred, then ETO (end of transmission, OK) is sourced and the function enters TADS.

TERS (Talker Error State)

In TERS the T function has just received the fre local message indicating that a transmission error has occurred. When the SH function signals that the current handshake cycle is complete, the ETE (end of transmission with error) is sourced and the function enters TADS.

TPIS (Talker Primary Idle State)

The function is able to respond to its primary talk address in TPIS but must not recognize its secondary address. This is the power-on state.

TPAS (Talker Primary Active State)

In TPAS the function has received its primary talk address and is now able to recognize and respond to its secondary address. Receipt of any frame other than IDY, RFC, MSA, or MTA (again) will cause return to TPIS.

2.9 L (Listener) Function

This interface function provides the device with the capability to receive device dependent data including status, device ID, and accessory ID over the loop from other devices. The L function also allows devices the capability to halt data transfers if enabled to do so by the controller.

When the device is addressed to listen, DOE (data or end) messages received from the loop may be interpreted in a device dependent manner. When not addressed to listen, received data may not be interpreted. Message receiving is controlled by the AH (Acceptor Handshake) interface function. If necessary and enabled by the controller, a listener may halt the data transfer by sending an NRD (not ready for data) sequence to the active talker as controlled by the AH (acceptor handshake) and SH (source handshake) functions.

To enhance the compatibility between the various devices on the loop it is strongly recommended that wherever possible device dependent data messages be sent using the ASCII code.

The talker may send the END message as part of the sent data to indicate an end-of-record condition. The listener(s) may use this information in a device dependent manner. The EOT (end of transmission) message, however, is intended only for the controller and is not interpreted by the listener(s).

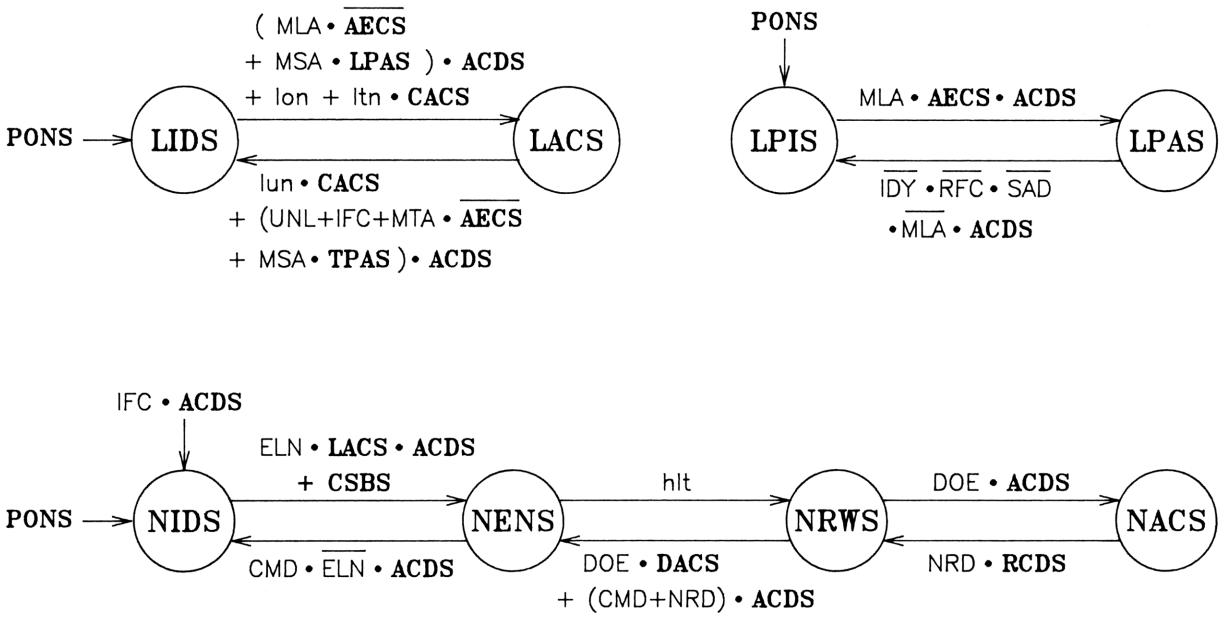
The L interface function controls only when data may be accepted from the loop but does not describe how the data is to be interpreted. Since the data transfer may be interrupted so that other operations may be performed, the listener must not allow state changes within the L function to affect the interpretation of the received data.

Listeners may stop active data transfers if enabled to do so by the controller with the ELN (enable listener not ready) command (controllers have this capability intrinsically). If necessary, the device may halt a data transfer with an NRD sequence by sending the local message hlt (halt data transfer) true. If the function is enabled, then it becomes ready to halt the transfer as soon as a DOE (data or end) message is received. When the AH (acceptor handshake) function signals that a DOE

frame has been received and is being held, the NRD message is sent out on the loop. As soon as the R function indicates that the NRD has returned, the AH function retransmits the held DOE frame to the talker. If an IFC (Interface Clear) frame is received at any time, the function immediately returns to its inactive state.

The most important usage of the transfer interrupt capability is for listeners to halt the input data before a buffer overrun condition occurs. When a listener detects that it no longer has room to hold the incoming data, it may send the NRD message (if enabled) to the talker before it is forced to stop all loop operations by holding a DOE frame.

Devices which do not implement the L function are designated LØ. The basic L capability is designated L1 and includes LIDS and LACS. If the L function includes the ability to operate in the listen-only mode, the number 2 is added to the designation. Devices must allow some means of disabling the lon message so that it is not always true. Devices without the listen only capability must always send the lon local message false. If the device has the capability to use a two byte address then LIDS, LACS, LPIS, and LPAS are required and the number 3 is added to the designation. Devices with the capability to halt data transfers add the number 4 to the designation and include NIDS, NENS, NRWS, and NACS. For example, a basic listener with listen-only mode would be written L1,2. A basic listener with extended addressing capability would be L1,3. A listener with listen-only capability, extended addressing, and the capability to halt the data transfer when necessary would be designated L1,3,4.



Messages

hlt	halt data transfer
lon	listen only
ltn	local listen
lun	local unlisten
CMD	command
DOE	data or end
ELN	enable listener not ready
IDY	identify
IFC	interface clear
MLA	my listen address
MSA	my secondary address
MTA	my talk address
NRD	not ready for data
RFC	ready for command
SAD	secondary address
UNL	unlisten

Interface States

LACS	listener active state (links to R,AH,RL,PP, DC,DT,DD)
LIDS	listener idle state
NACS	not ready active state (links to AH,SH)
NENS	not ready enabled state
NIDS	not ready idle state
NRWS	not ready wait state (links to R)
LPAS	listener primary addressed state (links to T)
LPIS	listener primary idle state
ACDS	acceptor data state (from AH)
AECS	auto extended configured state (from AA)
CACS	controller active state (from C)
CSBS	controller standby state (from C)
DACS	driver transmit from acceptor state (from D)
PONS	power on state (from PD)
RCDS	receiver data state (from R)
TPAS	talker primary addressed state (from T)

Figure 2-9. L State Diagram

LIDS (Listener Idle State)

In this state the device is not able to receive device dependent messages. The function powers on in this state. A listen-only device or a controller may use the appropriate local message (lon, ltn respectively) to enter LACS without being addressed to listen. If the MLA message is received and the device is not configured for extended addressing, then the function enters LACS. If the device is configured for extended addressing, both the MLA and MSA (my secondary address) messages must be received before LACS is entered.

LACS (Listener Active State)

In LACS the device has received its listen address (and its secondary address if extended addressing is used) and is ready to receive device dependent data messages over the loop. This message transfer is controlled by the AH (Acceptor Handshake) interface function. If a command is received that causes the device to become unaddressed, the function returns to LIDS. A controller device may use the lun local message to cause the function to return to LIDS.

LPIS (Listener Primary Idle State)

In LPIS the device is able to recognize its primary listen address but is not able to respond to its secondary address. The L function powers on in this state. If the device is configured for extended addressing, then the MLA (my listen address) message will cause a transition to LPAS.

LPAS (Listener Primary Active State)

The function has received its primary listen address and is now able to respond to and recognize its secondary address in this state. The receipt of any frame other than MSA, MLA (again), RFC, or IDY will cause a return to LPIS.

NIDS (Not Ready Idle State)

This is the power-on state. In NIDS the function is not enabled to interrupt a data transfer. If the device is active listener and the ELN (enable listener not ready) command is received or the device is controller, the function enters NENS.

NENS (Not Ready Enabled State)

In NENS the function has been enabled to perform the NRD sequence but is not actively doing so. If the device decides that the data transfer should be halted, it sends the local message hlt true and the function enters NRWS. If any command is received, the function returns to NIDS.

NRWS (Not Ready Wait State)

In this state, the function is waiting for a data byte to be received so that it can perform the NRD sequence. When a DOE frame is received, the function enters NACS. If a CMD or NRD frame is received, the function returns to NENS.

NACS (Not Ready Active State)

In NACS the function is in the process of sourcing the NRD frame. When the NRD frame returns, the function returns to NRWS.

2.1Ø SR (Service Request) Function

The SR function provides the device the ability to request service from the controller. Devices may request service by setting the SRQ (CØ) bit in DOE and IDY messages. The set SRQ bit notifies the controller that one or more devices on the loop have requested service. CMD and RDY frames do not have an SRQ bit and may not transmit the service request message.

The device enables the SR function to send the service request message by setting the local message rsv true. When the controller asks the device to send its status, the SR function stops sending the SRQ message on the loop unless the local message rsv is set false and then true again.

A device may send the service request message to the controller by setting the SRQ bit on any DOE or IDY frame that the device sources or retransmits. This requires a commitment on the part of the controller to source IDY frames periodically when no data is being transferred to check for service requests. If the loop is to be idle and periodic polling is not desirable, the controller may enable devices to source asynchronous IDY frames with the SRQ bit set by sending the EAR (enable asynchronous requests) command. When the controller receives an asynchronous IDY or decides on its own that it needs to perform further operations, it will source a UCG (universal command) to disable further asynchronous requests and proceed with normal operations. While devices are enabled to source asynchronous IDYs, the controller must not send any frames except UCG or IDY. Since asynchronous IDYs may not travel the entire loop, the parallel poll bits will not necessarily be correct.

Two bits in the first byte of a device's status are reserved and must not be used for any other purpose. Bit 7 (msb) is set to indicate that the first byte is a system status message and is clear otherwise. System status messages are a device independent method of communicating status information (Refer to appendix C for a complete list of system status messages and their usage). Bit 6 is always equal to the device's local message rsv.

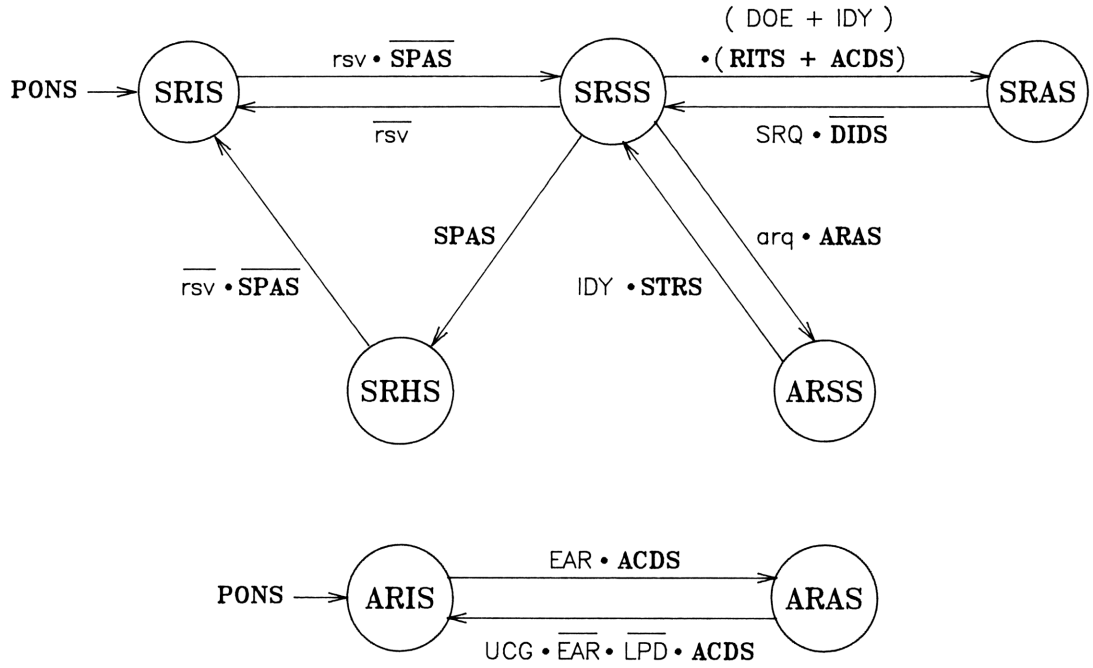
To enhance the compatibility between many different devices, system status messages should be used whenever possible. This allows generalized controllers to be implemented that can understand status messages from any device on the loop. Most devices can describe all possible status with the system messages defined. If more specific information is necessary or desirable, the device may send additional bytes of status.

Devices that do not implement system status should not request service for reasons other than data integrity unless enabled by the controller. Events which involve data integrity are defined to be those events that risk either data loss or an infinite halt of the loop data transfer. The controller may enable devices to send the SRQ message for causes other than data integrity (if the device has the capability) using either device dependent commands or ASCII commands defined for that purpose.

Devices that implement system status should request service for data integrity reasons and may optionally send the SRQ message whenever the current status changes. System status messages are required for change of status SRQs because the controller needs no device specific information to correctly interpret the request.

When a controller receives a service request from a device, it should stop the current operation immediately and find the reason for the SRQ. If data loss is a possibility, there may be a limited amount of time before data is actually lost.

SRØ indicates no service request capability. Basic service request capability is indicated by SR1 with the states ARSS, ARIS, and ARAS omitted. SR2 indicates full SR capability including asynchronous service requests with no states omitted. SR1 and SR2 both require the T2 subset of the talker function to send the device's status byte(s). Note that the arq message may be sent true only when SRSS is active. It must be sent false immmediately on entry to ARSS.



Messages

rsv	request service
arq	asynchronous request
DOE	data or end
EAR	enable asynchronous requests
IDY	identify
LPD	loop power down
SRQ	service request
UCG	universal command

Interface States

ARAS	asynchronous request active state
ARIS	asynchronous request idle state
ARSS	asynchronous request source state (links to SH)
SRIS	service request idle state
SRSS	service request standby state
SRAS	service request active state
SRHS	service request hold state
ACDS	acceptor data state (from AH)
DIDS	driver idle state (from D)
PONS	power on state (from PD)
RITS	receiver immediate transfer state (from R)
SPAS	serial poll active state (from T)
STRS	source transfer state (from SH)

Figure 2-10. SR State Diagram

SRIS (Service Request Idle State)

In SRIS the SR function is not requesting service. This is the power-on state. When the rsv local message goes true the function enters SRSS provided the controller is not currently requesting the device's status byte(s) via a serial poll.

SRSS (Service Request Standby State)

In this state the device is waiting for an opportunity to send the SRQ message. When a DOE or IDY frame arrives the function enters SRAS. If ARAS is active and the arq local message is true the function enters ARSS. If the device no longer requires service the function returns to SRIS. When the controller conducts a serial poll to retrieve the device's status the function enters SRHS.

SRAS (Service Request Active State)

In SRAS a DOE or IDY message has been received and the device is setting the SRQ bit in this message prior to its transmission by the D (Driver) interface function. When the transmission begins, the function returns to SRSS.

ARSS (Asynchronous Request Source State)

In this state the device sources its own IDY message with the SRQ bit set. When the D function indicates that transmission has begun the function returns to SRSS.

SRHS (Service Request Hold State)

In SRHS the device requires service but may not request it over the loop. When the device sends the local message rsv false, the function returns to SRIS.

ARIS (Asynchronous Request Idle State)

In ARIS the device may not generate asynchronous service requests. This is the power on state. When the EAR message is received the function moves to ARAS.

ARAS (Asynchronous Request Active State)

In this state the device is enabled to generate asynchronous service requests. If any universal command other than EAR or LPD is received the function returns to ARIS.

2.11 RL (Remote Local) Function

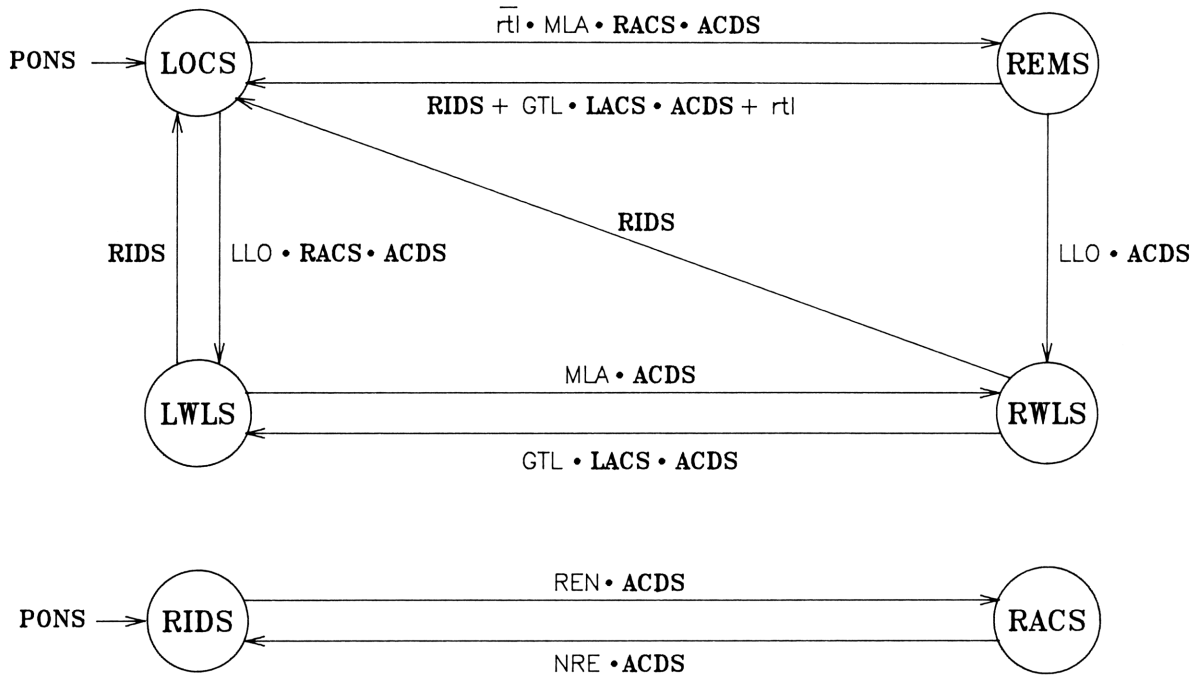
The RL interface function provides a device with the capability to select between two sources of input information. The input may either be from a local source (such as a front panel keyboard) or from the interface.

The RL function allows the device to have three basic levels of local control. Full local control allows all local controls such as switches or keyboards to be fully operative. Device commands received in this mode may be ignored or treated as device dependent data. Remote control requires the device to ignore all local controls that have corresponding remote controls. Local controls that generate local messages to the interface functions will continue to be operative including the rtl message which can be used to return the function to local control. Remote control with local lockout requires the device to ignore all local controls including those that generate local messages.

Note that this function does not affect a device's ability to send or receive messages on the loop, although device interpretation of data messages may be altered. Selected devices may be placed in remote control by sending the device's listen address, while others are left under local control. Note also that the IFC (Interface Clear) message does not affect the RL function.

When the function transitions from one of the local states to one of the remote states the device functions may retain their local settings until receiving specific remote settings, or they may immmediately revert to the remote settings previously received. Conversely, when returning to local control, the device functions may retain their remote settings until local controls override them, or they may immediately revert to the present settings of the local controls.

Devices without RL capability are designated RLØ. RL1 indicates basic RL capability without local lockout and includes all states except LWLS and RWLS. Note that devices designated RL1 must always send the rtl local message false. RL2 indicates full RL capability including local lockout and includes all states. Both RL1 and RL2 designations require the L function.



Messages

rtl	return to local
GTL	go to local
LLO	local lockout
MLA	my listen address
NRE	not remote enable
REN	remote enable

Interface States

LOCS	local state
LWLS	local with lockout state
RACS	remote active state
REMS	remote state
RIDS	remote idle state
RWLS	remote with lockout state
ACDS	acceptor data state (from AH)
LACS	listener active state (from L)
PONS	power on state (from PD)

Figure 2-11. RL State Diagram

RIDS (Remote Idle State)

In RIDS the function is not enabled for remote operation. The function powers on in this state. If the REN (remote enable) message is received the function enters RACS.

RACS (Remote Active State)

In this state the function is indicating that the active controller has enabled the interface for remote operation, however, the device functions remain under local control if LOCS is active. If the NRE (not remote enable) message is received the function immediately returns to LOCS and RIDS.

LOCS (Local State)

In LOCS all local controls of the associated device functions (both front and rear panel) are operative. The device may store but not respond to corresponding device dependent messages from the interface. The RL function powers on in this state. If RACS is active and MLA is received the function will enter REMS, provided the rtl local message is not true. If LLO (local lockout) is received the function enters LWLS.

LWLS (Local With Lockout State)

In this state, as in LOCS, the local controls are operative. The device may store but not respond to corresponding device dependent messages from the loop. If RACS is active and MLA is received the function enters RWLS. NRE causes the function to return to LOCS.

REMS (Remote State)

In REMS the local controls which have corresponding remote controls are inoperative and those device functions are under control of messages received from the interface. This does not include those controls which generate local messages for the interface functions. If the NRE or GTL messages are received or if the local message rtl goes true the function returns to LOCS. If the LLO message is received the function enters RWLS.

RWLS (Remote With Lockout State)

In this state all local controls are inoperative and the function will not respond to the rtl local message. If the GTL message is received the function goes to LWLS. If the NRE command is received, the function will return to LOCS.

2.12 AA (Automatic Address) Function

The AA function gives the device the ability to have its address assigned by the active controller. Either a one byte or a two byte address may be assigned to each device. If all devices have the capability to accept a two byte address, a maximum of 961 devices may be on the loop.

At power-on the device has no address assigned to it and may elect to respond to a default address or respond to no address at all. When an implemented addressing sequence is received, the device becomes configured for the assigned address. While the device is configured, it may not respond to any other address or accept another address assignment. If at any time the AAU (auto address unconfigure) message is received, the function immediately returns to its unconfigured state and the device must assign itself a default or switch selectable address. Note that since devices may not have any address at all at power-on, controllers intending to use the default or switch selectable addresses must send the AAU command at least once.

The least significant five bits of the AAD (auto address) AEP (auto extended primary), AES (auto extended secondary), and AMP (auto multiple primary) messages contain the binary address assignment, addresses Ø through 3Ø. Address 31 is not a legal address (11111). This message is called IAA, IEP, IES, or IMP (illegal AAD, AEP, AES, AMP respectively) and the AA function does not respond in any way to these message though it might generate them in response to an incoming auto address message.

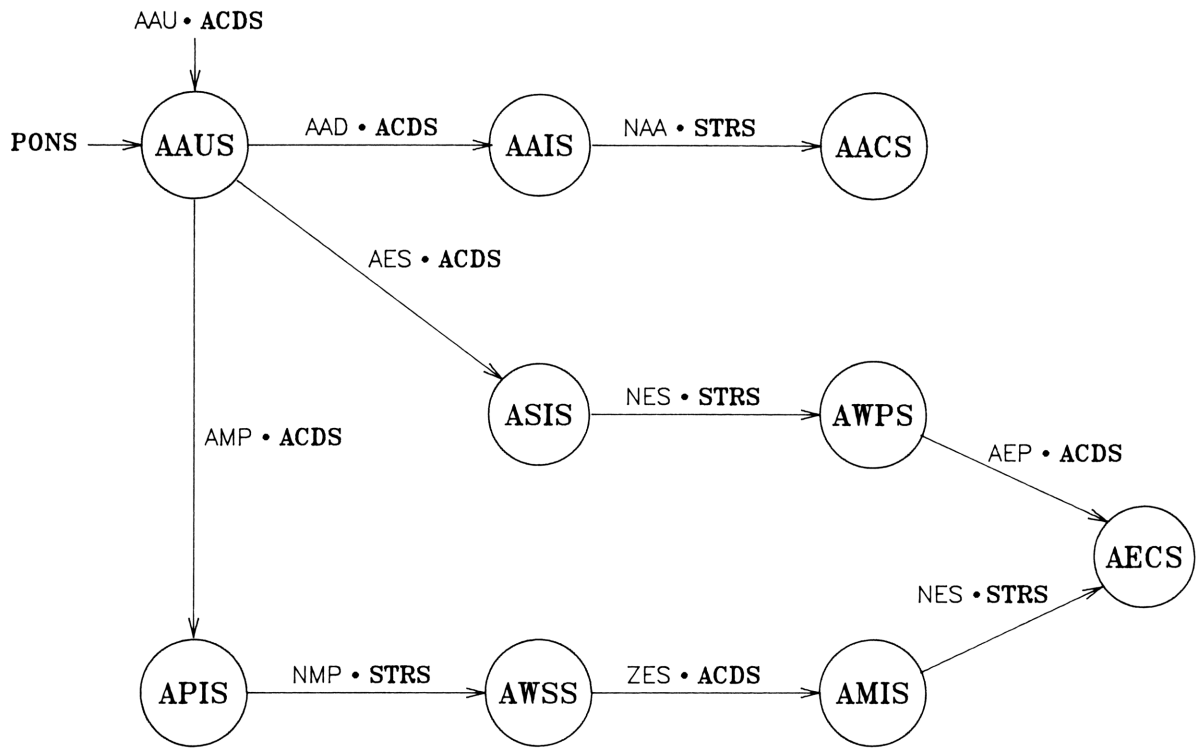
To configure the loop for one byte addressing, the controller would send the AAD1 (auto address 1) message (reserving address Ø for itself). If a valid AAD message returns, then the address it contains is one more than the number of devices on the loop. If the returned message is IAA then the loop may contain more than 3Ø devices and the controller should send AAD3Ø to check. If the AAD3Ø returns unchanged then there are exactly 3Ø devices on the loop and the controller may proceed as usual. If IAA is returned again, more than 3Ø devices are on the loop and improper operations may result.

Auto extended addressing allows each device on the loop to be asigned a two byte address so that a maximum of 961 devices

may be connected at once. For configuration, the controller would send the AES1 (auto extended secondary 1) message (reserving address Ø,Ø for itself). The first group of devices would receive their secondary addresses and increment the AES until IES is generated. When the controller receives the IES message, it will send the AEPØ (auto extended primary Ø) message. Only those devices which have received legal secondary addresses will respond to and accept this primary address. This group is now configured and will not respond to any auto address messages until AAU is received. Note that the primary address is not incremented by the loop devices and must be incremented by the controller. The controller should now send AESØ followed by AEP1 to configure the next group of devices. The controller should continue this AESØ-AEPn sequence until the AES message returns with a legal address (not IES). The value of the returned AES message and the coresponding AEP message sent indicate how many devices are on the loop. This addressing sequence must be sent before the auto multiple sequence if both are to be used on one loop, or it will not work correctly.

Auto multiple addressing allows devices with multiple functions to be assigned one primary address and have a separate secondary address for each function. This form of addressing allows as many as 31 devices (including the controller) with up to 31 functions in each. The configuration sequence begins when the controller sends the AMP1 (auto multiple primary) message (reserving address Ø for itself). The address returned in the AMP message indicates the number of devices connected (IMP, illegal multiple primary, would indicate that there may be more than 3Ø devices on the loop as in simple auto addressing). Then the controller would send ZES (zero extended secondary) once for each device. The return values indicate the number of separate functions in each device. If both auto extended and auto multiple addressing are to be implemented in the same loop, the auto extended addressing sequence must be sent first so that the ZES message of the auto multiple sequence is not incorrectly used by extended addressing devices

Simple auto addressing is designated AA1. Extended addressing is designated AA2. Auto multiple addressing is designated AA3. All devices must implement one or more of these three subsets. For example, a device that can be assigned either one or two byte addresses is designated AA1,2.



Messages

AAD	auto address
AAU	auto address unconfigure
AEP	auto extended primary
AES	auto extended secondary
AMP	auto multiple primary
NAA	next auto address
NES	next extended secondary
NMP	next multiple primary
ZES	zero extended secondary

Interface States

AACS	auto address configured state
AAIS	auto address increment state (links to SH)
AAUS	auto address unconfigured state (links to R)
AECS	auto extended configured state (links to L,T)
AMIS	auto multiple increment state (links to SH)
APIS	auto primary increment state (links to SH)
ASIS	auto secondary increment state (links to SH)
AWPS	auto wait for primary state (links to R)
AWSS	auto wait for secondary state (links to R)
ACDS	acceptor data state (from AH)
PONS	power on state (from PD)
STRS	source transfer state (from SH)

Figure 2-12. AA State Diagram

AAUS (Auto Address Unconfigured State)

In AAUS the function has no automatic address assignment. At power-on the device may elect to respond either to a preset or switch selectable address or to no address at all (in order to reduce the chances of two devices having the same address). After the AAU message is received, the device must respond to either a default or switch selectable address. If the AAD (auto address), AES (auto extended secondary), or AMP (auto multiple primary) messages are received, the function enters AAIS, ASIS, or APIS respectively. AAUS is the power-on state.

AAIS (Auto Address Increment State)

In AAIS the AAD message has been received with the device's primary address contained in the lower five bits. In this state the device accepts the address assignment and then increments it to generate the NAA (next auto address) message for the next device. When the NAA transmission begins the function enters AACS.

AACS (Auto Address Configured State)

In this state the device is configured for one byte addressing and must only respond to the address assigned by the active controller in the AAD message. The device may not respond to AAG (auto address group) messages while AACS is active. If the AAU command is received the function returns to AAUS.

ASIS (Auto Secondary Increment State)

In ASIS the AES message has been received with the device's secondary address assignment contained in the lower five bits. The device accepts this assignment and increments the AES message to generate the NES (next extended secondary) message for the next device. When transmission of NES begins the function enters AWPS.

AWPS (Auto Wait for Primary State)

In AWPS the function has received its secondary address assignment and has sent the NES message to the next device. The function now waits for the AEP message containing its primary address. The function enters AECS when AEP is received. The AAU command causes a transition to AAUS.

APIS (Auto Primary Increment State)

In APIS the function has received the AMP message containing its primary address assignment in the lower five bits. The function accepts the assigned primary address and increments it to generate the NMP (next multiple primary) message for the next multiple function device. When transmission of NMP begins the function enters AWSS.

AWSS (Auto Wait for Secondary State)

The function has its primary address and is waiting for the ZES (zero extended secondary) message. When the ZES message is received, the function enters AMIS. If AAU is received the function returns to AAUS.

AMIS (Auto Multiple Increment State)

In AMIS the device has received the ZES message and is assigning each function of the device a secondary address starting with zero. The last address assigned within the device becomes the NES message. Note that all devices to be configured for extended addressing (and not auto multiple addressing) must already be configured or they will respond to the NES message. When transmission begins, the function enters AECS.

AECS (Auto Extended Configured State)

In this state the device is configured for two byte addressing and must only respond to the primary and secondary addresses assigned by the active controller. The function will no longer respond to any remote messages except AAU. If AAU is received the function returns to AAUS.

2.13 PD (Power Down) Function

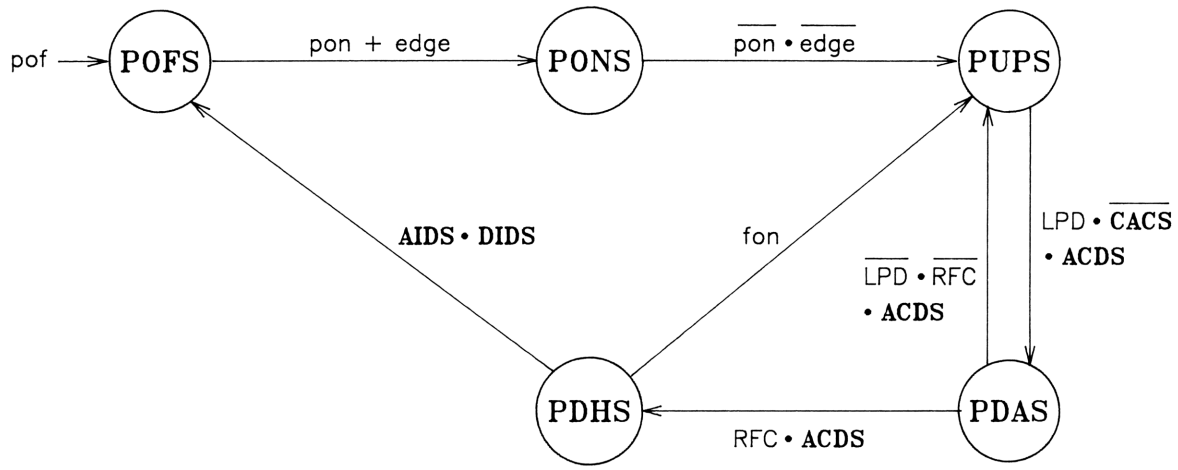
This function gives a device the capability to be placed in a power-down or low power mode by command received over the loop and likewise to be reactivated for normal operation. When in the power-down mode the device only needs to monitor the loop for any pulse. When any frame arrives the edge pseudomessage is generated which reactivates the device and places the interface functions in the power-on state.

If the PD function receives the LPD (loop power down) command it will proceed to turn the device off after the other interface functions have signaled that they are ready. If the local message fon (force on) is true, the device does not power down, but instead returns to the normal power-up state. If an edge is detected while the device is powered down, the function initializes the interface and device functions and returns to the normal power-up state. The local message pof allows a device to be powered down without receiving a command from the loop. If an edge is detected while the pof message is true, the device will remain in the powered-off state.

The active controller should wake up a powered-down loop by repeatedly sending a universal command (NOP is recommended) until it returns followed by one and only one RFC to complete the handshake. This will permit devices whatever time is necessary to complete their individual power-on sequences and begin to recognize and retransmit frames. The controller would do this either on its own initiative or in response to an asynchronous IDY if this mode was enabled prior to power-down.

Note that devices must not retransmit any frames after the LPD and RFC until the power down operation is complete. Devices may use the local message fon to disable the power down function, if desired. Because it must be able to wake up a powered down loop, the controller may not allow its interface functions to be powered down.

All devices must implement at least POFS, PONS, and PUPS. This capability is designated PDØ and requires that the edge message is always false. PDØ devices simply ignore the LPD command. PD1 indicates full capability. All states must be implemented as shown. No other subsets are permitted.



Messages

edge	pulse edge detected
fon	force on
pof	power off
pon	power on
LPD	loop power down
RFC	ready for command

Interface States

PDAS	power down active state
PDHS	power down hold state
PONS	power on state (links to all interface functions)
POFS	power off state
PUPS	power up state
ACDS	acceptor data state (from AH)
AIDS	acceptor idle state (from AH)
CACS	controller active state (from C)
DIDS	driver idle state (from D)

Figure 2-13. PD State Diagram

POFS (Power Off State)

This state is entered whenever the local message pof is true. This message is normally generated from the off position of the device power switch. All of the other interface functions have the equivalent of this state although it is only shown here for clarity. When POFS becomes active in the PD function, it causes an immediate transition to this equivalent state in all other interface functions. When the device power switch generates the pon local message or a frame arrives generating the edge message, the function enters PONS.

PONS (Power On State)

PONS causes the entry to the first (power-on) state in all other interface functions. The pon or edge message must remain true for sufficient time to allow interface and device functions to properly respond. When these messages both are false again, the function enters PUPS, the normal operating state.

PUPS (Power Up State)

PUPS is the normal operating state for the device. If the LPD command is received and the device is not the active controller, the function enters PDAS. The active controller must remain on in order to power up the loop when necessary. If the controller is also the system controller, it may allow itself to power down.

PDAS (Power Down Active State)

In PDAS the device is waiting for the RFC following the LPD command. When the RFC is received the function enters PDHS. If any other remote message is received the function aborts and returns to PUPS.

PDHS (Power Down Hold State)

In this state the function is waiting for the completion of the retransmission of the RFC message before actually going to the power down mode, POFS. If the fon local message is true, the function will return to PUPS.

2.14 PP (Parallel Poll) Function

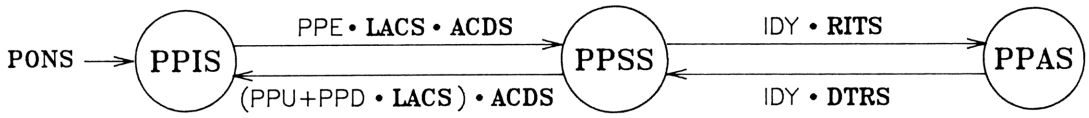
The PP interface function gives a device the ability to present to the controller one bit of status information without being addressed to talk. The eight data bits in the IDY message carry this status information from the devices to the controller. Any number of devices can be accomodated by sharing the available bits. Devices can be configured to present the logical OR or the logical AND of their respective status on a particular bit.

Whenever a device receives the PPE (parallel poll enable) command while addressed to listen, it becomes enabled to respond to parallel polls on the assigned bit with the assigned polarity on any IDY message received. This may result in setting one bit of an IDY frame before retransmission. The controller may assign a new response bit or polarity to a device at any time by sending a new PPE command while the device active listener. The PPD (parallel poll disable) addressed command and the PPU (parallel poll unconfigure) universal commands disable the parallel poll function.

An important use of parallel poll is reducing the controller's service request response time. When the SRQ message is received by the controller, a parallel poll may be conducted first to narrow the search. If the devices of interest are appropriately configured for parallel poll, the device requiring service may be located after sourcing only a few frames.

Three bits (D2, D1, and DØ) in the PPE command represent the binary bit number on which to respond to parallel polls (ØØØ assigns bit DØ (lsb), ØØ1 indicates bit D1, etc.). Another bit in the PPE command (D3) indicates the logical sense of a true response. If bit D3 is a Ø, the device's positive response is to not modify the incoming bit of the IDY; the negative response would be to OR in a 1 in the assigned bit. If bit D3 is a 1, the positive response is to OR in a 1, the negative response is no modification. This permits devices to be configured to present the logical OR or the logical AND of their responses. Note that regardless of the logical sense assigned, the device may only OR its response to the assigned bit and therefore may not affect the bit if it is received as a 1.

No capability is designated PPØ. Full capability is designated PP1. No subsets are permitted.



Messages

IDY	identify	PPE	parallel poll enable
PPD	parallel poll disable	PPU	parallel poll unconfigure

Interface States

PPAS	parallel poll active state	**ACDS**	acceptor data state (from AH)
PPIS	parallel poll idle state	**DTRS**	driver transfer state (from D)
PPSS	parallel poll standby state	**LACS**	listener active state (from L)
		PONS	power on state (from PD)
		RITS	receiver immediate transfer state (from R)

Figure 2-14. PP State Diagram

PPIS (Parallel Poll Idle State)

PPIS is the power-on state. In PPIS, the PP function may not respond to parallel polls and must retransmit all IDY messages exactly as received. When the PPE command is received, the function enters PPSS.

PPSS (Parallel Poll Standby State)

In PPSS the device is configured to respond to parallel polls and is waiting for the active controller to source the IDY message. When an IDY frame is received, the function enters PPAS. If a new PPE command is received, the function accepts a new bit number and polarity for PP response.

PPAS (Parallel Poll Active State)

In PPAS the function is actively responding to a parallel poll. The device may modify the IDY frame before retransmission by setting the assigned bit according to the assigned sense and the device's individual status. When the D function signals that retransmission has begun, the function returns to PPSS.

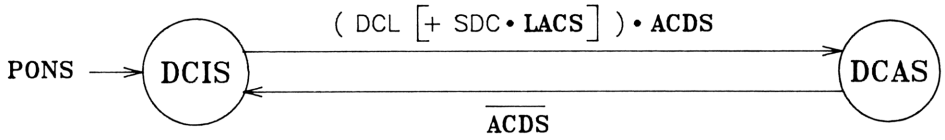
2.15 DC (Device Clear) Function

The DC interface function provides the device with the ability to be cleared or initialized. The Controller may choose to send the clear message to all devices on the loop or only to those devices addressed to listen.

Through the DC function, the device may receive and respond to the DCL (device clear) and SDC (selected device clear) messages. The DCL or SDC message allows the device to alter device conditions or states usually returning them to their initial or power-on states. The device may not alter any interface functions in response to the clear message. If both the DCL message and the SDC message are implemented they must affect exactly the same device functions in exactly the same way.

Note that this function affects only the device function and has no effect on the other interface functions (which are initialized by the IFC message or the power on state). The device may use the DC function for any purpose consistent with its operation. This would typically include placing the device (not the interface) in its power-on state, but may be used to place the device function in any predefined state as specified by the designer.

If the DC function is not implemented the designation is DCØ. DC1 designates the ability to respond to the DCL message. All devices with DC1 are cleared together by the DCL message. If the device also includes the ability to respond to the SDC message DC2 is the designation. This message clears only addressed devices. This capability requires the L function and the optional terms in the DC state diagram. If this capability is not implemented the optional terms must be omitted. Repeating, a device which can only respond to DCL is designated DC1 while a device that can respond to both DCL and SDC is designated DC2.



Messages

DCL device clear

SDC selected device clear

Interface States

DCAS device clear active state

DCIS device clear idle state

ACDS acceptor data state (from AH)

LACS listener active state (from L)

PONS power on state (from PD)

Figure 2-15. DC State Diagram

DCIS (Device Clear Idle State)

In this state the DC function is not actively clearing the device functions. When the DCL message is received or the SDC message is received and the device is in Listener Active State, the function enters DCAS.

DCAS (Device Clear Active State)

In DCAS the the function is signaling the device to clear or initialize its internal function. As soon as the device signals to the AH function that it has accepted the clear message, the function returns to DCIS.

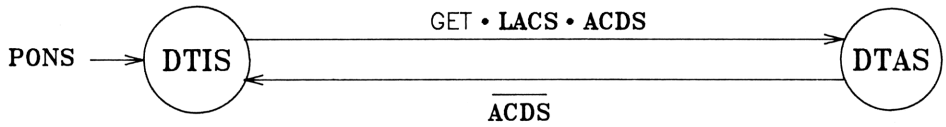
2.16 DT (Device Trigger) Function

The DT function gives the device the ability to have its basic operation started on command. Since the device must be addressed to listen in order to respond to the trigger command, either a single device or a group of devices may be triggered at the same time. The command does not affect other interface functions.

Through the DT function, the device may receive and respond to the GET (group execute trigger) message. To trigger a device to begin its basic operation, the active controller must source the GET command when the device is active listener. An important use of the DT function is to allow the controller to synchronize the basic operations of several devices on the loop. Therefore, it is recommended that the device commence its basic operation as soon as DTAS becomes active. Once the device operation is started the device should not respond to further transitions of the DT interface function until the operation is complete.

Note that the GET message affects only the device functions and has no effect on any other interface functions. When the trigger message is received, the device may perform any action consistent with the device's operation as defined by the device designer.

If the device does not include the device trigger capability it is designated DTØ. If the complete device trigger capability is present, it is designated DT1 and requires the L function. No subsets are permitted.



Messages

GET group execute trigger

Interface States

DTAS device trigger active state
DTIS device trigger idle state
ACDS acceptor data state (from AH)
LACS listener active state (from L)
PONS power on state (from PD)

Figure 2-16. DT State Diagram

DTIS (Device Trigger Idle State)

In this state the DT function is not active. The function powers on in this state. If the GET command is received and the device is listener active, the function enters DTAS.

DTAS (Device Trigger Active State)

In DTAS the DT function is signaling the device function to start performing its basic operation. When the device indicates to the AH (acceptor handshake) function that the trigger message has been received, the function returns to DTIS.

2.17 DD (Device Dependent Command) Function

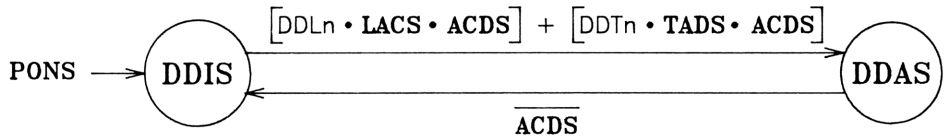
The DD function allows the device to accept device dependent commands from the interface to be used as the designer chooses. Sixty-four command codes are reserved for device dependent purposes. There are 32 DDL commands that a device may interpret only if it is active listener. There are also 32 DDT commands that a device may interpret only if addressed to talk.

The device dependent commands received by the DD interface function must affect only device functions. Interface functions such as talker and listener must not be affected by any device dependent command.

It is recommended that the device commence its response as soon as DDAS becomes active. Once this response is begun the device should not respond to further transitions of the DD interface function until the operation is complete.

A separate DD interface function should be implemented for each device dependent command to which the device responds. Only the appropriate one of the two optional terms in the transition will be implemented in any particular DD function.

No capability is designated DDØ. If the device responds to one or more DDL or DDT commands, it is designated DD1. DD1 devices must include appropriate documentation on each device dependent command response.



Messages

DDLn device dependent listener command n

DDTn device dependent talker command n

Interface States

DDAS device dependent active state

DDIS device dependent idle state

ACDS acceptor data state (from AH)

LACS listener active state (from L)

PONS power on state (from PD)

TADS talker addressed state (from T)

Figure 2-17. DD State Diagram

DDIS (Device Dependent Idle State)

In this state the device is not performing the operation the designer has specified for the command represented by this interface function. When the command is received and the device is appropriately addressed, the function transitions to DDAS.

DDAS (Device Dependent Active State)

In DDAS the function is indicating to the device that it should begin performing the operation specified for the command represented by this interface function.

2.18 Remote Message Coding

This section gives definitions and codes for all remote messages. Remote messages on the signal lines are sensed by the decoder, are decoded, and are passed on to the interface functions for appropriate action. Interface functions originate and control the transmission of remote messages to the encoder, which encodes and transmits them over the signal lines.

Throughout this section, Ø represents a logical zero signal, 1 represents a logical one signal, and X represents a receiver "don't care" or a transmitter "no change" for retransmission. The letters a, b, m, n, s, and t are used to describe bits with specific meanings. These are described in the notes at the end of the section.

The first three bits of each frame (C2, C1, CØ in that order) are called control bits and serve to classify frames into major categories for retransmission and response. The following table gives the codes for these bits of the frame and their meanings. Please note that the interface system responds in precisely the same way to both data and end messages. The end bit is provided solely as a mechanism for devices to use to indicate an end-of-record condition within device dependent message transmissions without necessarily terminating the transmission. The end bit is C1 and the service request bit is CØ in those frames which have provision for these bits.

CONTROL BIT CODES

C2	C1	CØ	Classification	Mnemonic
Ø	Ø	Ø	data byte	DAB
Ø	Ø	1	data byte with service request	DAB (SRQ)
Ø	1	Ø	end byte	END
Ø	1	1	end byte with service request	END (SRQ)
1	Ø	Ø	command	CMD
1	Ø	1	ready	RDY
1	1	Ø	identify	IDY
1	1	1	identify with service request	IDY (SRQ)

The remaining eight bits (D7 through DØ in order) are called data bits (not to be confused with a data message) and indicate the specific message within the general classifications described above. The specific codes for command, ready, and identify messages are given in this section. The codes for data or end messages may be any eight bit pattern which the involved devices are all able to interpret. It is recommended for maximum compatibility, however, that the ASCII code be used whenever possible. Bit Ø of the ASCII code representation corresponds to HP-IL bit DØ and bit 7 corresponds to D7. Regardless of the data coding used, it is recommended that the most significant bit be sent on D7 and the least significant bit correspond to DØ.

Messages may be defined as the logical combination (AND, OR, or NOT) of other messages. Messages are therefore divided into several classes and sub-classes. To facilitate the user's understanding of these message relationships a frame hierarchy table is included. This table graphically shows which messages are included within other messages. An alphabetic listing of all messages, their mnemonics, and their coding follows the frame hierarchy table. For convenience, a table of messages by coding is also included.

FRAME HIERARCHY TABLE

```
DOE .... DAB
       . DAB(SRQ)
       . END
       . END(SRQ)

CMD .... ACG .... NUL
       .        . GTL
       .        . SDC
       .        . PPD
       .        . GET
       .        . ELN
       .        . PPEØ(Ø-7)
       .        . PPE1(Ø-7)
       .        . DDL(Ø-31)
       .        . DDT(Ø-31)
       .
       . UCG .... NOP
       .        . LLO
       .        . DCL
       .        . PPU
       .        . EAR
       .        . IFC
       .        . REN
       .        . NRE
       .        . AAU
       .        . LPD
       .
       . LAG .... LAD(Ø-3Ø)
       .        . MLA(Ø-3Ø)
       .        . UNL
       .
       . TAG .... TAD(Ø-3Ø)
       .        . MTA(Ø-3Ø)
       .        . OTA(Ø-3Ø)
       .        . UNT
       .
       . SAG .... SAD(Ø-3Ø)
                . MSA(Ø-3Ø)
                . OSA(Ø-3Ø)
```

```
IDY .... IDY
       . IDY(SRQ)

RDY .... RFC
       .
       . ARG .... EOT .... ETO
       .        .        . ETE
       .        .
       .        . NRD
       .        .
       .        . SOT .... SDA
       .                  . SST
       .                  . SDI
       .                  . SAI
       .                  . TCT
       .
       .
       . AAG .... AAD(Ø-3Ø)
                . NAA(1-31)
                . IAA
                .
                . AEP(Ø-3Ø)
                . IEP
                .
                . ZES
                . AES(Ø-3Ø)
                . NES(1-31)
                . IES
                .
                . AMP(Ø-31)
                . NMP(1-31)
                . IMP
```

MESSAGE TABLE

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 210	DDDDDDDD 76543210	NOTES
AAD	auto address 0-30	RDY	AAG	101	100aaaaa	1
AAG	auto address group	RDY		101	1xxxxxxx	
AAU	auto address unconfigure	CMD	UCG	100	10011010	
ACG	addressed command group	CMD		100	x000xxxx	
			or	100	101xxxxx	
			or	100	110xxxxx	
AEP	auto extended primary 0-30	RDY	AAG	101	101aaaaa	1
AES	auto extended secondary 0-30	RDY	AAG	101	110aaaaa	1
AMP	auto multiple primary 0-30	RDY	AAG	101	111aaaaa	1
ARG	addressed ready group	RDY		101	01xxxxxx	
CMD	command			100	xxxxxxxx	
DAB	data byte	DOE		00x	xxxxxxxx	
DCL	device clear	CMD	UCG	100	00010100	
DDL	device dependent listener command 0-31	CMD	ACG	100	101xxxxx	
DDT	device dependent talker command 0-31	CMD	ACG	100	110xxxxx	
DOE	data or end			0xx	xxxxxxxx	
EAR	enable asynchronous requests	CMD	UCG	100	00011000	
ELN	enable listener not ready	CMD	ACG	100	00001111	
END	end	DOE		01x	xxxxxxxx	
EOT	end of transmission	RDY	ARG	101	01000000x	
ETE	end of transmission, error	RDY	ARG	101	01000001	

MESSAGE TABLE (continued)

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 21Ø	DDDDDDDD 76543210	NOTES
ETO	end of transmission, OK	RDY	ARG	1Ø1	Ø1ØØØØØØ	
GET	group execute trigger	CMD	ACG	1ØØ	ØØØØ1ØØØ	
GTL	go to local	CMD	ACG	1ØØ	ØØØØØØØ1	
IAA	illegal auto address	RDY	AAG	1Ø1	1ØØ11111	
IDY	identify			11x	xxxxxxxx	
IEP	illegal extended primary	RDY	AAG	1Ø1	1Ø111111	
IES	illegal extended secondary	RDY	AAG	1Ø1	11Ø11111	
IFC	interface clear	CMD	UCG	1ØØ	1ØØ1ØØØØ	
IMP	illegal multiple primary	RDY	AAG	1Ø1	11111111	
LAD	listen address Ø-3Ø	CMD	LAG	1ØØ	ØØ1aaaaa	1
LAG	listen address group	CMD		1ØØ	ØØ1xxxxx	
LLO	local lockout	CMD	UCG	1ØØ	ØØØ1ØØØ1	
LPD	loop power down	CMD	UCG	1ØØ	1ØØ11Ø11	
MLA	my listen address Ø-3Ø	CMD	LAG	1ØØ	ØØ1mmmmm	2
MSA	my secondary address Ø-3Ø	CMD	SAG	1ØØ	Ø11mmmmm	2
MTA	my talk address Ø-3Ø	CMD	TAG	1ØØ	Ø1Ømmmmm	2
NAA	next auto address 1-31	RDY	AAG	1Ø1	1ØØnnnnn	3
NES	next extended secondary 1-31	RDY	AAG	1Ø1	11Ønnnnn	3
NMP	next multiple primary 1-31	RDY	AAG	1Ø1	111nnnnn	3
NOP	no operation	CMD	UCG	1ØØ	ØØØ1ØØØØ	
NRD	not ready for data	RDY	ARG	1Ø1	Ø1ØØØØ1Ø	

MESSAGE TABLE (continued)

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 321	DDDDDDDD 87654321	NOTES
NRE	not remote enable	CMD	UCG	100	10010011	
NUL	null command	CMD	ACG	100	00000000	
OSA	other secondary address	CMD	SAG	100	011ttttt	4
OTA	other talk address	CMD	TAG	100	010ttttt	4
PPD	parallel poll disable	CMD	ACG	100	00000101	
PPE	parallel poll enable 0-15	CMD	ACG	100	1000sbbb	5
PPU	parallel poll unconfigure	CMD	UCG	100	00010101	
RDY	ready			101	xxxxxxxx	
REN	remote enable	CMD	UCG	100	10010010	
RFC	ready for command	RDY		101	00000000	
SAD	secondary address 0-30	CMD	SAG	100	011aaaaa	1
SAG	secondary address group	CMD		100	011xxxxx	
SAI	send accessory ID	RDY	ARG	101	01100011	
SDA	send data	RDY	ARG	101	01100000	
SDC	selected device clear	CMD	ACG	100	00000100	
SDI	send device ID	RDY	ARG	101	01100010	
SOT	start of transmission	RDY	ARG	101	01100xxx	
SRQ	service request	DOE		0x1	xxxxxxxx	
	or	IDY	or	111	xxxxxxxx	
SST	send status	RDY	ARG	101	01100001	
TAD	talk address 0-30	CMD	TAG	100	010aaaaa	1
TAG	talk address group	CMD		100	010xxxxx	

MESSAGE TABLE (continued)

MNEMONIC	MESSAGE	CLASS	GROUP	CCC 321	DDDDDDDD 87654321	NOTES
TCT	take control	RDY	ARG	101	01100100	
UCG	universal command group	CMD		100	x001xxxx	
UNL	unlisten	CMD	LAG	100	00111111	
UNT	untalk	CMD	TAG	100	01011111	
ZES	zero extended secondary	RDY	AAG	101	11000000	

NOTES

1. 'aaaaa' represents a five bit binary device address in the range 0 to 30. The address may be either a primary listen or talk address or a secondary address, depending on the particular message. The least significant bit corresponds to bit D0.

2. 'mmmmm' represents the particular device address which matches the address assigned to this device.

3. 'nnnnn' represents the incremented device address which is transmitted by a device during auto address configuration. The range of this incremented value is 1 to 31. Note that 31 is not a valid address.

4. 'ttttt' represents a device address which does not match the address assigned to this device. The address may be either a primary talk address or a secondary address, depending on the particular message.

5. 's' represents the sense of a positive response to a parallel poll (IDY). If s is Ø the device should do nothing for a positive response and should change its assigned bit to a 1 for a negative response. If s is a 1 the device should change its assigned bit to a 1 for a positive response and should do nothing for a negative response. "Do nothing" means retransmit the frame unchanged.

6. 'bbb' is the binary bit number on which the device the device responds to parallel poll. ØØØ means bit DØ, ØØ1 means bit D1, and so on to 111, which means respond on bit D7.