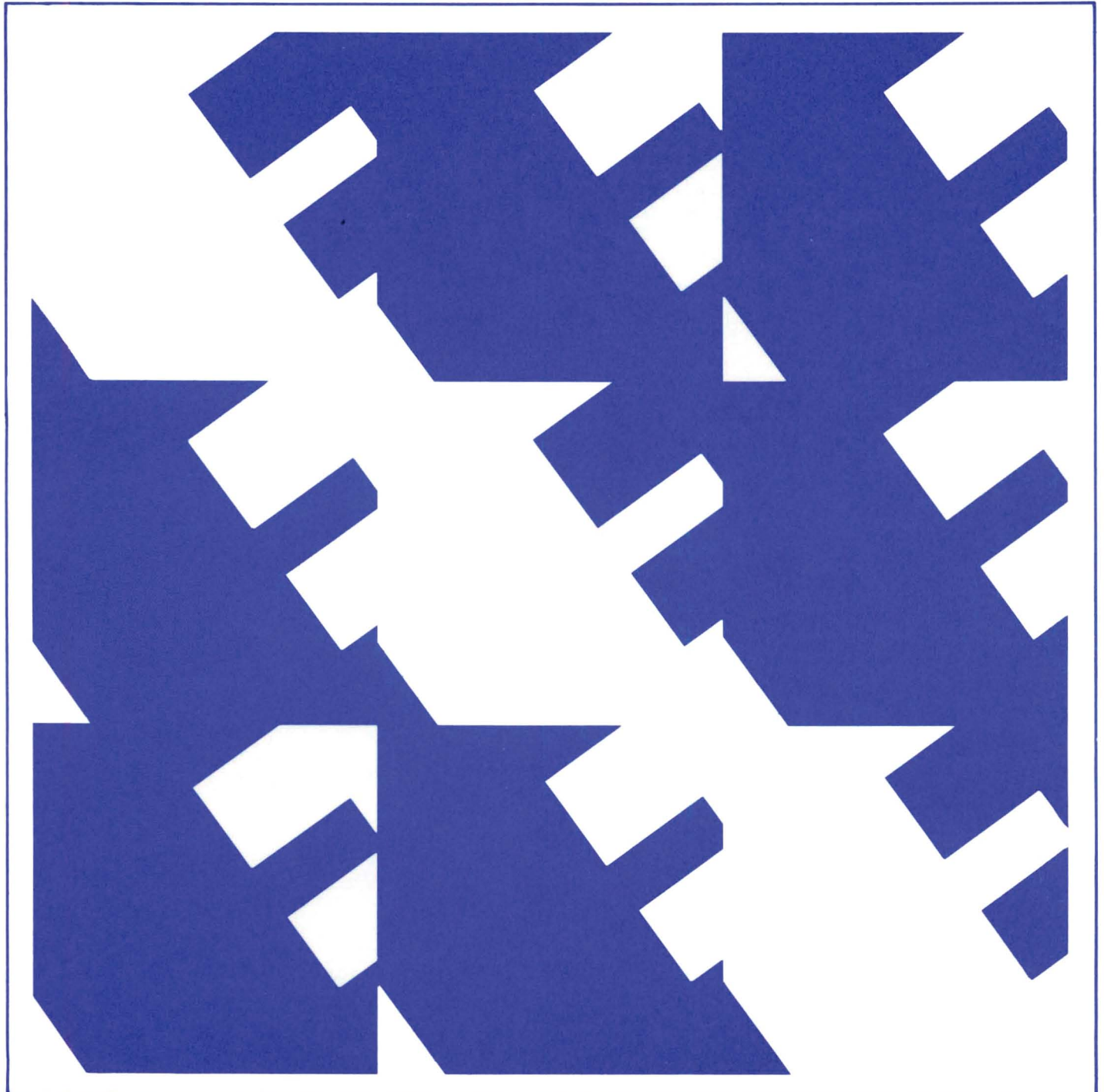


IEEE Standard Digital Interface for Programmable Instrumentation

ANSI/IEEE Std 488-1978 and Supplement



Published by The Institute of Electrical and Electronics Engineers, Inc. 345 East 47th Street, New York, NY 10017, USA

November 30, 1978

Authorized licensed use limited to: Tsinghua University. Downloaded on February 25, 2026 at 16:11:07 UTC from IEEE Xplore. Restrictions apply.

SH07260



Documents

Document Sharing Hub

ANSI/IEEE
Std 488-1978
(Revision of ANSI/IEEE
Std 488-1979)
(Includes supplement
IEEE Std 488A-1980)

An American National Standard

**IEEE Standard Digital Interface for
Programmable Instrumentation**

Sponsor
IEEE Instrumentation and Measurements Group

Approved June 9, 1978
IEEE Standards Board

Approved July 18, 1979
American National Standards Institute

© Copyright 1978 by

The Institute of Electrical and Electronics Engineers, Inc

*No part of this publication may be reproduced in any form,
in an electronic retrieval system or otherwise,
without the prior written permission of the publisher.*

NOTE RELATION TO IEC STANDARDS

(This note is not a part of IEEE Std 488-1978)

IEEE Standard 488-1978 is in concert with its international counterpart, International Electrotechnical Commission (IEC) Publication 625-1*. Major portions of the forthcoming IEC publication and this standard are identical, word for word. A number of paragraphs and clauses in the IEC publication have, for international reasons, received slightly different editorial treatment although the technical content and intended meaning are also identical with this standard. In some instances, there are apparent differences although, in fact, the technical content is identical (for example, the IEC publication assumes, via special notation, whereas this standard explicitly identifies state change exit conditions for IFC). The two standards are, in practice, identical in their functional and electrical specifications. The IEC document uses a mechanical connection system identical to the EIA RS 232-C data communication interface. Component damage, due to incompatible voltage levels, is possible if data communication and instrumentation interfaces are inadvertently interconnected. IEEE and US industry practice is to minimize component damage to programmable instrumentation systems by specifying the connector system of this standard. Any mechanical specification difference between the IEEE and IEC standards may be accommodated by a simple (physical) adapter assembly when products implemented from the two standards are interconnected.

Fourth Printing

Corrected edition July 25, 1980

Includes supplement IEEE Std 488A-1980

Approved September 27, 1979

IEEE Standards Board

*Publication dated 1979.

Foreword

(This Foreword is not a part of ANSI/IEEE Std 488-1978, IEEE Standard Digital Interface for Programmable Instrumentation.)

IEEE Std 488-1975 has enjoyed wide use since its initial publication. Experience with the standard indicates that some portions of the written text could be more clearly stated. A draft international standard IEC Publication 625* using the same interface concepts has been further elaborated since the original issuance of IEEE Std 488 and provides additional points of clarification. The principle improvements to the written text of this revised standard focus on editorial changes rather than substantive technical changes to the standard itself. Further, where changes of a technical nature have been made, they provide clarification of potentially ambiguous statements and broadening of certain specifications. However, the changes do not contradict the technical concepts as defined in the original published standard. Those changes that could have a bearing on the technical compatibility between independently designed products are identified as follows:

- Clauses 2.3.3, 2.4.2, (SH Function), END Message, Table 3, Table 6
- Clause 2.4.2, AH Function, Optional Transition Term, Fig 4
- Clause 2.5.2, TE and T Function, and L Alternate Transition, Fig 6 Note
- Clause 2.5.2, (TE Message Outputs), END Message, Table 9
- Clause 2.6.2, LE and T Function, Alternate Transition, Fig 8, Table 17 Notes
- Clause 2.8.1, RL and LE Function, Alternate Transition, Fig 10
- Clause 2.12.2, C Function, Transitions Clarified, Fig 14
- Clause 2.13.3, Valid Uniline Message Transfer Times
- Clause 2.13.5, Remote Message Coding, Perspectives, Table 38
- Clause 3.3.2, Driver Specifications, Schottky Levels
- Clause 3.4.2, Receiver Specification, Preferred
- Clause 3.7.2, 4.3, Cable Construction
- Clause 4.4, Connector Mounting, Dimensional Tolerances, Fig 18
- Clause 5.2.3 Data Rate Consideration, Higher Speed Operation
- Clause 6.2.4, Devices Powered Off
- Clause 6.3.1, UNT use
- Clause 6.5, Operational Sequences, Exit Condition Caution
- Clause 6.5.2, Serial Poll Completion
- Clause 6.5.4, Parallel Poll Configure, Recommended Sequence
- Appendix C, Interface Function Capability Identification

NOTE: Text changes associated with the above mentioned clauses are identified by a vertical line in the margin opposite the relevant subject matter.

The IEEE Std 488-1978, IEEE Standard Digital Interface for Programmable Instrumentation, deals with systems that use a byte-serial bit-parallel means to transfer digital data among a group of instruments and system components. The interface system described herein is optimized as an interdevice interface for system components in relatively close proximity able to communicate over a contiguous party-line bus system.

This document contains seven sections as follows.

Section 1 contains the scope, the object, basic definitions, and summary description of the interface.

Section 2 deals with functional concepts and specifications of the interface system described in this standard. One or more interface functions contained within a device are each able to process messages and change states to maintain an orderly flow of information among a set of interconnected devices.

Section 3 deals with the electrical realization of the interface in order to transfer messages among a set of interconnected devices.

Section 4 deals with the mechanical realization of the interface in order to implement the electrical aspects of the interface system.

*Will be published as IEC Publication 625-1.

Section 5 deals with system considerations that must be given to the design of an individual device in order to make it compatible with other devices of a measurement system.

Section 6 deals with system considerations that must be recognized by the user of devices designed in accordance with this standard.

Appendixes deal with explanatory matter and examples.

In order to design equipment in accordance with this standard, the designer must have read the entire document.

In order to interconnect and program equipment designed in accordance with this standard, the user should have knowledge of Sections 1 and 6. If the coding and transfer of messages is not done automatically by the apparatus to be programmed, it will be necessary that the user have knowledge of Section 2. General familiarity with the other sections is recommended. The user must also be familiar with device-dependent characteristics of apparatus that may be used in a system, but that are beyond the scope of this standard.

This standard defines an interface with the objective to assure that messages may be accurately communicated between two or more devices in a system, but does not guarantee that each device will interpret properly all possible messages sent to it or will properly generate all necessary messages. A wide latitude of interface capability is permitted within the scope of this standard which may permit operational incompatibility among interconnected devices.

A device designer must have sufficient awareness of the characteristics of systems which might include his device in order to select correctly among the options provided in this standard. Likewise, a system configurator must have sufficient awareness of the options included in each of the devices in his system in order to ensure that the correct communication techniques are used.

This standard does not specify the device-dependent or operational characteristics required for complete system compatibility. Therefore, following the rules and procedures of this standard alone will not guarantee unconditional compatibility.

The interface specified by this standard includes patented matter which pertains only to the three-wire handshake as defined throughout 2.3 and 2.4 and summarized in Appendix B.

The IEEE Standards Office calls attention to the fact that it is claimed that the three-wire handshake referenced in 2.3 and 2.4 is the subject of one US patent and corresponding patents of foreign countries owned by the Hewlett-Packard Company. Although these patents appear to cover the subject of 2.3 and 2.4 in this standard, the IEEE takes no position with respect to patent validity. The Hewlett-Packard Company has assured the IEEE that it is willing to grant a license under these patents on reasonable and nondiscriminatory terms and conditions to anyone wishing to obtain such a license. The Hewlett-Packard Company's undertakings in this respect are on file with the IEEE Standards Office and the license details may be obtained from the legal department of the Hewlett-Packard Company whose address is 1501 Page Mill Road, Palo Alto, Calif 94304.

This standard is based on work initiated by the International Electrotechnical Commission (IEC) within Technical Committee 66, Working Group 3, and follows the general concepts of a standard prepared by the IEC. This standard influenced, and was influenced by, working documents in the IEC.

The two IEEE Committees responsible for the preparation and evaluation of this standard within the US were the Instrumentation and Measurements Group Subcommittee on Instrument/Computer Interfaces (which also served as the US Advisory Committee to US representatives on IEC TC 66/WG3) and the Instrumentation and Measurements Group Technical Committee on Automated Instrumentation.

Personnel for the Instrument/Computer Interfaces Subcommittee were:

R. G. Fulks, *Chairman*

D. C. Loughry, *Secretary*

L. Costrell
H. Draye
G. Foley
M. Gipe
M. Graube

T. J. Harrison
D. Lehmer
J. Pepin
B. Perlman
D. Picker

L. C. Sollman

Personnel for the Automated Instrumentation Technical Committee were:

F. Liguori, *Chairman*

C. W. Osborn, *Secretary*

O. R. Batchelder
B. Bendel
R. D. Chew
D. M. Goodman
J. A. Houston

R. P. Keats
D. C. Loughry
R. H. Nesson
J. K. Skilling
M. G. Slade

P. Sylvan

When this standard was approved June 9, 1978, the IEEE Standards Board had the following membership:

Joseph L. Koepfinger, *Chairman*

Irvin N. Howell, Jr, *Vice Chairman*

Ivan G. Easton, *Secretary*

William E. Andrus
C. N. Berglund
Edward J. Cohen
Warren H. Cook
David B. Dobson
R. O. Duncan
Charles W. Flint

Jay Forster
Ralph I. Hauser
Loering M. Johnson
Irving Kolodny
William R. Kruesi
Thomas J. Martin
John E. May

Donald T. Michael
Voss A. Moore
William S. Morgan
Robert L. Pritchard
Blair A. Rowley
Ralph M. Showers
B. W. Whittington

Contents

SECTION	PAGE
A Helpful Note to Users of IEEE Std 488-1978.	10
1. General.	11
1.1 Scope	11
1.2 Object	11
1.3 Definitions	12
1.4 Interface System Overview	13
2. Functional Specifications.	15
2.1 Functional Partition	15
2.2 Notation Used to Specify Interface Functions	18
2.3 SH (Source Handshake) Interface Function	20
2.4 AH (Acceptor Handshake) Interface Function	23
2.5 T (Talker) Interface Function (Includes Serial Poll Capabilities).	25
2.6 L (Listener) Interface Function.	29
2.7 SR (Service Request) Interface Function	34
2.8 RL (Remote Local) Interface Function.	35
2.9 PP (Parallel Poll) Interface Function	37
2.10 DC (Device Clear) Interface Function	39
2.11 DT (Device Trigger) Interface Function	41
2.12 C (Controller) Interface Function	42
2.13 Remote Message Coding and Transfer	47
3. Electrical Specifications'.	52
3.1 Application	52
3.2 Logical and Electrical State Relationships.	52
3.3 Driver Requirements	52
3.4 Receiver Requirements	53
3.5 Composite Device Load Requirements	53
3.6 Ground Requirements.	54
3.7 Cable Characteristics	54
3.8 State Transition Timing Values	55
4. Mechanical Specifications.	56
4.1 Application	56
4.2 Connector Type	56
4.3 Connector Contact Assignments	56
4.4 Device Connector Mounting	56
4.5 Cable Assembly	57
5. System Applications and Guidelines for the Designer.	59
5.1 System Compatibility	59
5.2 Data Rate Consideration	59
5.3 Device Capabilities	59
5.4 AND and OR Logic Operations.	60
5.5 Address Assignment	61
5.6 Typical Combinations of Interface Functions	62
6. System Requirements and Guidelines for the User.	63
6.1 System Compatibility	63
6.2 System Installation Requirements.	63
6.3 Address Assignment	63
6.4 Cabling Restrictions	64
6.5 Operational Sequence Guidelines	64

FIGURES	PAGE
Fig 1 Interface Capabilities and Bus Structure.	14
Fig 2 Functional Partition Within a Device	16
Fig 3 SH State Diagram.	20
Fig 4 AH State Diagram	22
Fig 5 T State Diagram	25
Fig 6 TE State Diagram.	27
Fig 7 L State Diagram	31
Fig 8 LE State Diagram.	32
Fig 9 SR State Diagram.	34
Fig 10 RL State Diagram.	35
Fig 11 PP State Diagram	37
Fig 12 DC State Diagram.	39
Fig 13 DT State Diagram.	41
Fig 14 C State Diagram	43
Fig 15 DC Load Boundary Specification.	54
Fig 16 Typical Signal Line Input-Output Circuit	54
Fig 17 Connector Panel Cutout.	57
Fig 18 Mounting Dimensions.	57
Fig 19 Stacked Connector Assembly.	58
Fig 20 Lock Screw	58
Fig 21 Cable Connector Housing.	58
Fig 22 Remote — Local Message Paths	60
Fig 23 Bi-State Signal Line Logic (Open Collector Drivers).	61
Fig 24 Signal Line Logic and Timing Relationships.	62

TABLES

Table 1 Interface Function Repertoire	17
Table 2 SH Mnemonics	21
Table 3 SH Message Outputs	21
Table 4 Allowable Subsets to SH Function	22
Table 5 AH Mnemonics.	23
Table 6 AH Message Outputs.	23
Table 7 Allowable Subsets to AH Function.	24
Table 8 T Mnemonics	26
Table 9 T or TE Message Outputs	26
Table 10 TE Mnemonics	26
Table 11 Allowable Subsets to T Interface Function.	29
Table 12 Allowable Subsets to TE Interface Function	30
Table 13 L Mnemonics	30
Table 14 L or LE Message Outputs	30
Table 15 LE Mnemonics	31
Table 16 Allowable Subsets to L Interface Function.	33
Table 17 Allowable Subsets to LE Interface Function	33
Table 18 SR Mnemonics	34
Table 19 SR Message Outputs.	34
Table 20 Allowable Subsets to SR Interface Function	34
Table 21 RL Mnemonics	35
Table 22 RL Message Outputs.	36
Table 23 Allowable Subsets to RL Interface Function	36
Table 24 PP Mnemonics	38
Table 25 PP Message Outputs	38
Table 26 PPR Message Specified by Each of the Combination of Values P1 Through P3.	39
Table 27 Allowable Subsets to PP Interface Function.	39

TABLES	PAGE
Table 28 DC Mnemonics	40
Table 29 DC Message Outputs	40
Table 30 Allowable Subsets to DC Interface Function	40
Table 31 DT Mnemonics	41
Table 32 DT Message Outputs	41
Table 33 Allowable Subsets to DT Interface Function	41
Table 34 C Mnemonics	42
Table 35 C Message Outputs	44
Table 36 Multiline Messages	45
Table 37 Allowable Subsets to C Interface Function.	48
Table 38 Remote Message Coding	50
Table 39 Time Values	55

APPENDIXES

Appendix A Typical Instrument System	67
A1. Event Sequence 1 (Device Dependent Data Returned to Processor)	67
A2. Event Sequence 2 (Device Dependent Data Directed to Digital Printer)	68
Appendix B Handshake Process Timing Sequence	69
B1. General Comments	69
B2. List of Events for Handshake Process	69
Appendix C Interface Function Allowable Subsets	72
C1. General Comments	72
C2. Capability Identification Codes	72
C3. SH Function Allowable Subsets	73
C4. AH Function Allowable Subsets	73
C5. T Function Allowable Subsets.	73
C6. T Function (with Address Extension) Allowable Subsets	74
C7. L Function Allowable Subsets.	74
C8. L Function (with Address Extension) Allowable Subsets	75
C9. SR Function Allowable Subsets	75
C10. RL Function Allowable Subsets	75
C11. PP Function Allowable Subsets	76
C12. DC Function Allowable Subsets	76
C13. DT Function Allowable Subsets	76
C14. C Function Allowable Subsets.	77
Appendix D Interface Message Reference List	78
Appendix E Multiline Interface Messages: ISO Code Representation	80
Appendix F Logic Circuit Implementation.	81
F1. Implementing States Which Require No Memory	81
F2. Implementing States Which Require Memory	81
Appendix G Parallel Polling Sequence	83

APPENDIX FIGURES

Fig A1 Typical System Showing Capability of Interface System to Handle Variety of Instrumentation System Needs	67
Fig A2 Signal Line Timing Sequence for One Talker and Multiple Listeners Using Handshake Process	69
Fig B2 Logical Flow of Events for Source and Acceptor When Transferring Data Using Handshake Process.	70
Fig E1 Multiline Interface Messages: ISO-7 Bit Code Representation	80
Fig F1 Partial SR Function — (a) Composite Logic; (b) Gate Level Logic	82
Fig F2 Full SR Function — (a) Composite Logic; (b) Gate Level Logic	82
Fig G1 Parallel Poll Sequence: Signal Line Waveforms	83
Fig G2 Interface Function Active States During Parallel Poll	83

A HELPFUL NOTE TO USERS OF IEEE STD 488-1978

(This note is not a part of IEEE Std 488-1978)

IEEE Std 488-1978, Digital Interface for Programmable Instrumentation, specifies mechanical threaded fasteners per metric thread size ISO M3.5X0.6 as defined throughout Section 4, Mechanical Specifications.

The IEEE Standards Office calls your attention to the fact that a number of products designed for use with IEEE Std 488-1978 contain nonmetric threaded fasteners in contradiction to 4.4 (Fig 18) and 4.5 (Fig 20) of the standard. This situation results from the fact that some products otherwise in conformity with the standard were designed and manufactured before IEEE Std 488-1975 was approved and published. Most suppliers of instrumentation products now utilize metric threaded fasteners and fully satisfy IEEE Std 488-1978.

A number of manufacturers now use metric threaded fasteners with a black finish (for example, copper sulfate over nickel) to designate the use of metric threads whereas the older nonmetric parts have a shiny nickel finish. In addition, these parts are stamped with a small letter M on the top surface to further indicate the metric part.

In the interest of minimizing mechanical incompatibility problems for both manufacturers and users of this standard, it is recommended that the metric parts defined by 4.4 and 4.5 of IEEE Std 488-1978 be identified with a black finish to indicate and emphasize the use of metric threads.

Further details may be obtained from the IEEE Standards Office as to the technical specifications and typical sources of supply for metric threaded parts. Cooperation in this matter will facilitate the implementation and use of IEEE Std 488-1978.

An American National Standard

IEEE Standard Digital Interface for Programmable Instrumentation

1. General

1.1 Scope. This standard applies to interface systems used to interconnect both programmable and nonprogrammable electronic measuring apparatus with other apparatus and accessories necessary to assemble instrumentation systems.

It applies to the interface of instrumentation systems, or portions of them, in which the:

- (1) Data exchanged among the interconnected apparatus is digital (as distinct from analog)
- (2) Number of devices that may be interconnected by one contiguous bus does not exceed 15
- (3) Total transmission path lengths over the interconnecting cables does not exceed 20 m
- (4) Data rate across the interface on any signal line does not exceed 1 Mb/s.

The basic functional specifications of this standard may be used in digital interface applications which require longer distances, more devices, increased noise immunity, or combinations of these. Different electrical and mechanical specifications may be required (for example, symmetrical circuit configurations, high threshold logic, special connectors, or cable configurations) for these extended applications.

This standard may also be applicable to other instrumentation system elements such as processors, stimulus, display, or storage devices, and terminal units found useful in instrumentation systems.

It applies generally to laboratory and production test environments which are both electrically quiet and restricted as to physical dimensions (distances between the system components).

This standard deals only with the interface characteristics of instrumentation systems to the exclusion of design specifications, consideration of radio-interface regulations, performance requirements, and safety requirements of apparatus.

NOTE: For the latter two items, reference is made to IEC-Publ 348 (1971), Safety Requirements for Electronic Measuring Apparatus, and IEC-Publ 359 (1971), Expression of the Functional Performance of Electronic Measuring Equipment.¹

Throughout this standard, and insofar as further distinction is not necessary, the term “system” denotes the byte-serial bit-parallel interface system that, in general, includes all circuits, cables, connectors, message repertoire, and control protocol to effect unambiguous data transfer between devices; and the term “device” or “apparatus” denotes any programmable measurement device or other product connected to the interface system that communicates information via, and conforms to, the interface system definition.

A primary focus of this standard is to set forth an interface system to interconnect self-contained apparatus to other apparatus by external means. This same standard may be applied to interconnecting the internal subsections within a self-contained equipment.

1.2 Object. This standard is intended:

- (1) To define a general-purpose system for use in limited distance applications
- (2) To specify the device-independent mechanical, electrical, and functional interface requirements that the apparatus shall meet in order to be interconnected and communicate unambiguously via the system.

¹IEC documents are available from the American National Standards Institute, 1430 Broadway, New York, NY 10018.

(3) To specify the terminology and definitions related to the system

(4) To enable the interconnection of independently manufactured apparatus into a single functional system

(5) To permit apparatus with a wide range of capability — from the simple to the complex — to be interconnected to the system simultaneously

(6) To permit direct communication between the apparatus, without requiring all messages to be routed to a control or intermediate unit

(7) To define a system with a minimum of restrictions on the performance characteristics of the apparatus connected to the system

(8) To define a system that permits asynchronous communication over a wide range of data rates

(9) To define a system that, of itself, may be relatively low cost and permits the interconnection of low cost devices

(10) To define a system that is easy to use.

1.3 Definitions. The following definitions apply for the purpose of this standard.

This section contains only general definitions. Detailed definitions are given in further sections, as appropriate.

1.3.1 General System Terms

compatibility. The degree to which devices may be interconnected and used, without modification, when designed as defined throughout this standard (for example, mechanical, electrical, or functional).

handshake cycle. The process whereby digital signals effect the transfer of each data byte across the interface by means of an interlocked sequence of status and control signals. Interlocked denotes a fixed sequence of events in which one event in the sequence must occur before the next event may occur.

interface. A common boundary between a considered system and another system, or between parts of a system, through which information is conveyed.

interface system. The device-independent mechanical, electrical, and functional elements of an interface necessary to effect communication among a set of devices. Cables, connector, driver and receiver circuits, signal line descriptions, timing and control conventions,

and functional logic circuits are typical interface system elements.

local control. A method whereby a device is programmable by means of its local (front or rear panel) controls in order to enable the device to perform different tasks. (Also referred to as manual control.)

programmable. That characteristic of a device that makes it capable of accepting data to alter the state of its internal circuitry to perform a specific task(s).

remote control. A method whereby a device is programmable via its electrical interface connection in order to enable the device to perform different tasks.

system. A set of interconnected elements constituted to achieve a given objective by performing a specified function.

1.3.2 Units Connected Via the Interface System

programmable measuring apparatus. A measuring apparatus that performs specified operations on command from the system and, may transmit the results of the measurement(s) to the system.

terminal unit. An apparatus that terminates the considered interface system and by means of which a connection (and translation, if required) is made between the considered interface system and another external interface system.

1.3.3 Signals and Paths

bidirectional bus. A bus used by any individual device for two-way transmission of messages, that is, both input and output.

bit-parallel. Refers to a set of concurrent data bits present on a like number of signal lines used to carry information. Bit-parallel data bits may be acted upon concurrently as a group (byte) or independently as individual data bits.

bus. A signal line or a set of signal lines used by an interface system to which a number of devices are connected and over which messages are carried.

byte. A group of adjacent binary digits operated on as a unit and usually shorter than a computer word (frequently connotes a group of eight bits).

byte-serial. A sequence of bit-parallel data bytes used to carry information over a common bus.

high state. The relatively more positive signal level used to assert a specific message content associated with one of two binary logic states.

low state. The relatively less positive signal level used to assert a specific message content associated with one of two binary logic states.

signal. The physical representation of information.

NOTE: For the purpose of this standard, this is a restricted definition of what is often called "signal" in more general terms, and is hereinafter referred to digital electrical signals only.

signal level. The magnitude of signal compared to an arbitrary reference magnitude (voltage in the case of this standard).

signal line. One of a set of signal conductors in an interface system used to transfer messages among interconnected devices.

signal parameter. That parameter of an electrical quantity whose values or sequence of values convey information.

unidirectional bus. A bus used by any individual device for one-way transmission of messages only, that is, either input only or output only.

1.4 Interface System Overview

1.4.1 Interface System Objective. The overall purpose of an interface system is to provide an effective communication link over which messages are carried in an unambiguous way among a group of interconnected devices.

Messages (quantities of information) carried by an interface system belong to either of two broad categories:

(1) Messages used to manage the interface system itself, hereinafter called interface messages

(2) Messages used by the devices interconnected via the interface system that are carried by, but not used or processed by the interface system directly, hereinafter called device dependent messages

NOTE: The detailed specification of device dependent messages is beyond the scope of this standard.

1.4.2 Fundamental Communication Capabilities. An effective communication link requires three basic functional elements to orga-

nize and manage the flow of information to be exchanged among devices:

- (1) A device acting as a listener
- (2) A device acting as a talker
- (3) A device acting as a controller

In the context of the interface system described by this standard:

(1) A device with the capability to listen can be addressed by an interface message to receive device dependent messages from another device connected to the interface system.

(2) A device with the capability to talk can be addressed by an interface message to send device dependent messages to another device connected to the interface system.

(3) A device with the capability to control can address other devices to listen or to talk. In addition, this device can send interface messages to command specified actions within other devices. A device with only this capability neither sends nor receives device dependent messages.

NOTE: The use of the word controller throughout this standard applies strictly to the management (control) of the interface system and does not imply the broad capabilities typically associated with the word in the data processing context. Further classification of the controller will be made in Section 2 to distinguish between different types of controller capabilities related to the interface system.

Listener, talker, and controller capabilities occur individually or in any combination in devices interconnected via the interface system as shown in Fig 1.

In addition to the basic listener, talker, and controller functions, the system provides interface messages to accomplish the following operations:

(1) A serial poll sequence may be initiated when a device (with talker function) requires some action by the controller, by transmitting the service request message. The controller will then obtain the status byte of all possible devices in sequence to ascertain which required service.

(2) The Parallel Poll function provides a device with the ability to transmit on the controller's demand one bit of status information (request service) simultaneously with several other devices. The assignment of a data line to a particular device for the response to a parallel poll may be accomplished through interface messages.

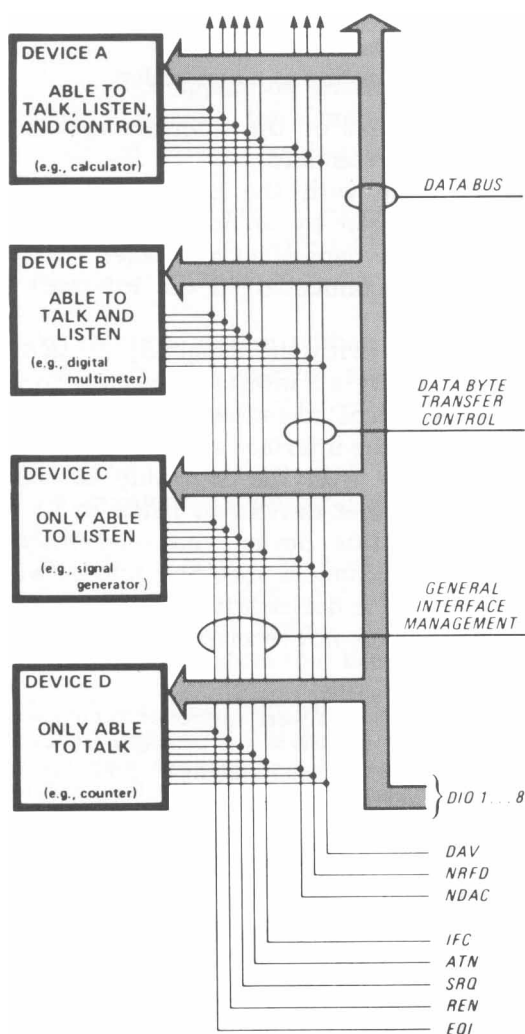


Fig 1
Interface Capabilities and Bus Structure

(3) The Device Clear and Device Trigger functions provide a device with the ability to be initialized or triggered, on command from the controller. This may occur simultaneously with other selected or all devices in a system.

(4) The remote/local function provides a device with the ability to select between program data from the bus and local data (for example, front panel controls.)

1.4.3 Message Paths and Bus Structure. The interface system contains a set of sixteen signal lines used to carry all information, in-

terface messages, and device dependent messages among interconnected devices.

Messages may be coded on one or a set of signal lines as determined by the particular message content and its relationship to the interface system.

The bus structure is organized into three sets of signal lines:

- (1) Data bus, 8 signal lines
- (2) Data byte transfer control bus, 3 signal lines
- (3) General interface management bus, 5 signal lines

Fig 1 illustrates the basic communication paths.

A set of eight interface signal lines carries all 7 bit interface messages and the device dependent messages:

- (1) DIO1 (data input output 1)
- .
- .
- .
- .
- .
- .
- (8) DIO8 (data input output 8)

Message bytes are carried on the DIO signal lines in a bit-parallel byte-serial form, asynchronously, and generally in a bidirectional manner.

NOTE: A message may be carried on an individual DIO signal line when required.

A set of three interface signal lines is used to effect the transfer of each byte of data on the DIO signal lines from a talker or controller to one or more listeners:

- (1) DAV (data valid) is used to indicate the condition (availability and validity) of information on the DIO signal lines
- (2) NRFD (not ready for data) is used to indicate the condition of readiness of device(s) to accept data
- (3) NDAC (not data accepted) is used to indicate the condition of acceptance of data by device(s)

The DAV, NRFD, and NDAC signal lines operate in what is called a three-wire (interlocked) handshake process to transfer each data byte across the interface.

Five interface signal lines are used to manage an orderly flow of information across the interface:

- (1) ATN (attention) is used (by a controller) to specify how data on the DIO signal lines

are to be interpreted and which devices must respond to the data

(2) IFC (interface clear) is used (by a controller) to place the interface system, portions of which are contained in all interconnected devices, in a known quiescent state

(3) SRQ (service request) is used by a device to indicate the need for attention and to request an interruption of the current sequence of events

(4) REN (remote enable) is used (by a controller) in conjunction with other messages to select between two alternate sources of device programming data

(5) EOI (end or identify) is used (by a talker) to indicate the end of a multiple byte transfer sequence or, in conjunction with ATN (by a controller), to execute a polling sequence

1.4.4 Interface System Elements. The primary elements of this interface system are:

- (1) Functional elements
- (2) Electrical elements
- (3) Mechanical elements

Each is described in a following section.

2. Functional Specifications

2.1 Functional Partition. A device is a physical entity designed for a particular application. It may be partitioned conceptually into three major functional areas each containing unique capabilities:

- (1) Device functions (definition is application dependent)
- (2) Interface functions (definition is application independent)
- (3) Message coding logic

All communication to or from interface functions is defined in terms of messages and state linkages (see 2.1.3).

All messages carried on the signal lines are coded according to the coding logic defined in 2.1.3.

2.1.1 Device Functions. The scope, purpose, size, content, and organization of the device function area (for example, analog signal measurement capability, range, modes of operation, etc) are beyond the scope of this standard. Fig 2 illustrates the device function area *B* for which the designer has complete freedom to define device related capability and the interface function area *A* for which the designer has no freedom to define new capability

beyond that specified in this standard.

2.1.2 Interface Function Concepts

2.1.2.1 Interface Functions. An interface function is the system element which provides the basic operational facility through which a device can receive, process, and send messages. A number of interface functions, each of which acts in accordance with specific protocol, are defined throughout this section of the standard. Each specific interface function may only send or receive a limited set of messages within particular classes of messages.

2.1.2.2 Interface Function State. Each of the interface functions is defined in terms of one or more groups of interconnected, mutually exclusive states.

One and only one state shall be active at any one time within a single group of interconnected, mutually exclusive states.

For each state of an interface function definitions are given for the following:

- (1) Messages that may or must be sent over the interface while that state is active
- (2) Conditions under which the function must leave that state and enter one of the other states in its group

These messages and conditions define the processing capability of the state.

2.1.2.3 Interface Function Repertoire. The designer is given the choice to select the particular set of interface functions necessary to fit the particular device application area. Fig 2 and Table 1 identify the available interface functions.

The total processing capability of a set of interface functions (designer selected set included in a specific device) at any moment is the logic conjunction of the processing capabilities of all those states (within each individual interface function) that are active at that moment.

2.1.2.4 Interface Function Assumptions and Perspective. The state diagrams used to define the interface functions do not indicate, either explicitly or implicitly, the intended existence of specific circuit elements to achieve the logical and physical implementation of a function. For example, not all states necessarily imply the existence of a latched flip-flop or other memory element.

The state diagrams used to define the interface functions are intended to permit the use of a wide variety of logic circuit implementa-

An interface function must ignore (not respond to) any message coding not specifically defined.

A function may stay in any state for any amount of time (including zero) after exit conditions are met if this is not in conflict with specified constraints.

2.1.3 Message Concepts

All communication between an interface function and its environment is accomplished through messages sent or received.

Fig 2

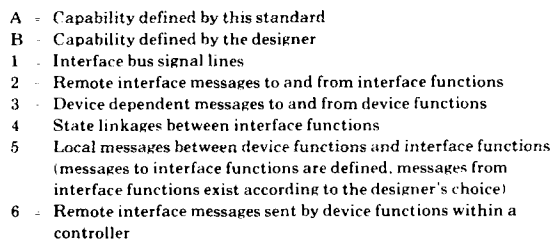


Table 1
Interface Function Repertoire

Interface Function	Symbol	Relevant Message Paths
Source handshake	SH	1, 2, 4, 5
Acceptor handshake	AH	1, 2, 4, 5
Talker or extended talker	T or TE	1, 2, 3, 4, 5
Listener or extended listener	L or LE	1, 2, 3, 4, 5
Service request	SR	1, 2, 4, 5
Remote local	RL	1, 2, 4, 5
Parallel poll	PP	1, 2, 4, 5
Device clear	DC	1, 2, 4, 5
Device trigger	DT	1, 2, 4, 5
Controller	C	1, 2, 4, 5, 6

2.1.3.2 Local Message Route and Content. Messages sent between a device function and an interface function are called local messages.

Local messages flow between device functions and interface functions; see Fig 2, message route 5.

NOTE: Certain local messages are conveyed as remote messages and vice versa.

The designer is not allowed to introduce new local messages to interface functions.

The designer is allowed to introduce a local message derived from any state of any interface function to device function(s).

Local messages sent by device functions must exist for enough time to cause the required state transitions.

2.1.3.3 Remote Message Route and Content. Messages sent via the interface between interface functions of different devices are called remote messages.

Each remote message is either an interface message or a device dependent message.

Each interface message is sent to cause a state transition within another interface function. An interface message will not be passed along to the device when received by an interface function as shown in Fig 2, message route 2.

Device dependent messages are passed between the device functions and the message coding logic via specified interface functions. These will cause no state transitions within the interface functions. Examples of device dependent messages include device programming data, device measurement data, and device status data as shown in Fig 2, message route 3.

2.1.3.4 State Linkage Route and Content. A state linkage is the logical interconnection of two interface functions where the transition to an active state of one interface function is dependent on the existence of a specified active state of another interface function as indicated in Fig 2, message route 4.

2.1.3.5 Message Coding. Message coding is the act of translating remote messages to or from interface signal line values.

A message sent over a single line is called a uniline message. Two or more of these messages can be sent concurrently.

A message that shares a group of signal lines with other messages, in some mutually exclusive set, is called a multiline message. Only one multiline message (message byte) can be sent at one time.

2.1.3.6 Classification of Multiline Messages. Multiline messages are interpreted as interface messages when the ATN message is true.

Multiline messages are interpreted as device dependent messages when the ATN message is false.

Each device, when the ATN message is true, may accept and process any or all of these specific classes of multiline messages as determined both by the extent of the interface function capability contained within the device and the current set of active interface function states:

- (1) Universal commands (all devices)
- (2) Addressed commands (all devices addressed to listen)
- (3) Addresses (all devices)
- (4) Secondary addresses or commands (all devices enabled by a primary address or command)

For a list of specific multiline commands see Table 36.

2.1.3.7 Message Transfer Conventions

2.1.3.7.1 Remote Message Transfer Conventions

(1) The value (true or false) of all remote messages capable of being sent by a device shall at all times be as dictated by active states of its interface functions.

(2) The interface signal line(s) used to send a message value shall be set to the levels specified by Table 38, Remote Message Coding.

(3) Since normal interface operation allows two devices to simultaneously send opposite values of the remote messages, a technique shall be provided for resolving these conflicts. This is accomplished by implementing two types of message transfer over the interface, active transfer and passive transfer. The interface is structured so that in all conflicts between two message values, one of them will be active and the other passive. Messages shall be transferred in such a way that the active value overrides the passive value in every conflict that arises.

(4) A remote message can be transferred in one of four ways:

(a) An active true value being sent is guaranteed to be the value received, and the device need not allow it to be overridden

(b) A passive true value being sent is not guaranteed to be the value received, and the device must allow it to be overridden

(c) An active false value being sent is guaranteed to be the value received, and the device need not allow it to be overridden

(d) A passive false value being sent is not guaranteed to be the value received and the device must allow it to be overridden

(5) Throughout the text, the terms true and false if not qualified are assumed to mean active true and active false during all discussions of remote message values sent by an interface function.

(6) For two specific remote messages, DAC and RFD, only false values are defined to be sent actively. Thus, an AND operation can be considered to be performed on the interface signal lines (see 5.4).

(7) For one remote message, SRQ, only true values are defined to be sent actively. Thus, an OR operation can be considered to be performed on the interface signal lines (see 5.4).

(8) Only the multiline message(s) to be sent

true will be specified for an interface function state since multiline messages (sent via the DIO lines) are by their nature mutually exclusive. It should be understood that all unspecified multiline messages are sent passive false while the state is active.

2.1.3.7.2 Local Message Transfer Conventions

(1) The coding of local messages is beyond the scope of this standard and is left to the discretion of the device designer.

(2) It is recommended that local messages qualifying transitions within any group of mutually exclusive states of an interface function be themselves mutually exclusive.

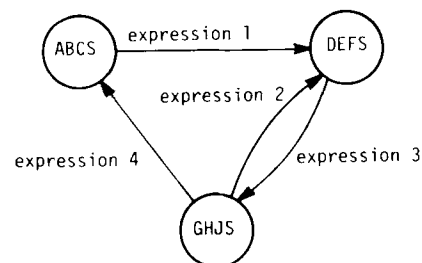
2.2 Notation Used to Specify Interface Functions

2.2.1 State Diagram Notation. Each state that an interface function can assume is represented graphically as a circle. A four-letter upper-case mnemonic, always ending in an S, is used within the circle to identify the state:



All permissible transitions between states of an interface function are represented graphically by arrows between them.

Each transition is qualified by an expression whose value is 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, one of these expressions becomes true. The new state may be entered at any time after the expression(s) become(s) true, unless a time value is specified.



An expression consists of one or more local messages, remote messages, state linkages, or minimum time limits used in conjunction with the operators AND, OR, or NOT.

A local message to an interface function is represented by a three-letter mnemonic written in lower case; for example, rdy.

A remote message (received via the interface) is represented by a three-letter mnemonic written in upper case; for example, ATN.

A linkage from another state diagram is represented by a four-letter mnemonic enclosed in an oval; for example, (LACS). A state linkage is true if the enclosed state is currently active; otherwise, it is false.

A minimum time limit is represented by the symbol T_n . This symbol achieves a true value only after the interface has been in the state originating the corresponding transition for the time value specified. It will remain true until the state is exited. The values for these time limits are contained in Table 39.

The AND operator is represented by the symbol $\wedge$.

The OR operator is represented by the symbol $\vee$.

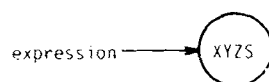
The AND operator takes precedence over the OR operator within an expression unless otherwise specified by parentheses.

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.

If a transition is further qualified by a maximum time limit (within t_n), then the state pointed to shall be entered within the specified amount of time after the expression becomes true. The values for these time limits are contained in Table 39.

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 [...].

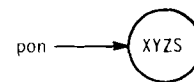
If a specific expression causes a transition to a state from all other states of the diagram, a shorthand notation is used instead of all the individual transitions being drawn. An arrow without a state at its origin is used to represent this condition, and is assumed to originate in all states (for example, IFC or REN):



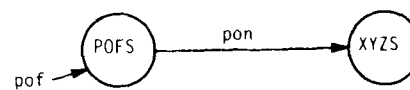
Although POFS (power-off) is a valid state of most interface functions and should nor-

mally be shown on all diagrams with a transition leading to the state to be entered at power-on time, a shorthand form is used showing the pon pseudomessage originating a transition to the first state to be entered when power is turned on:

(1) Abbreviated notation used on state diagram:



(2) Complete representation implied by preceding symbol:



2.2.2 Message Output Notation. The message output table included with each interface function state diagram summarizes only the remote messages allowed to be sent during each of the states of the function.

Rows of the table are used to indicate states of the interface function.

Columns of the table are used to indicate remote messages allowed to be sent during at least one state of the interface function.

Each table entry indicates the value of a message that shall be sent while a specified state is active:

- (1) T indicates active true
- (2) F indicates active false
- (3) (T) indicates passive true
- (4) (F) indicates passive false

One column in each table is allocated, if required, to the group of multiline remote messages allowed to be sent. The multiline message to be sent true during each state is placed in its corresponding table entry. False values are not shown since multiline messages are mutually exclusive. Parentheses around a multiline message name specify that it shall be sent passive rather than active true.

A separate column for device function interaction summarizes the corresponding types of messages (or resultant action) device function(s) are allowed to send or receive. Local message(s), beyond the scope of this standard, from the interface function to the device functions may be used to coordinate the appropriate action at the choice of the designer.

(2) The SIDS within t_2 if either:

(a) The ATN message is true and neither CACS nor CTRS is active

(b) Or the ATN message is false and neither TACS nor SPAS is active

2.3.3.3 SDYS (Source Delay State). In SDYS the SH function is waiting for a message byte to settle on the interface signal lines after the change during SGNS, and for all the acceptor functions to indicate their readiness to accept the message byte.

In SDYS the SH function shall send the DAV message false. In this state the device shall not change the multiline message being sent.

The SH function shall exit SDYS and enter:

(1) The STRS (source transfer state) only after T_1 if the RFD message is true

(2) The SIDS within t_2 if either:

(a) The ATN message is true and neither CACS nor CTRS is active

(b) Or the ATN message is false and neither TACS nor SPAS is active

2.3.3.4 STRS (Source Transfer State). In STRS the SH function indicates to the AH function that it is continuously sending a valid message byte.

In STRS the SH function shall send the DAV message true. In this state the device shall not change either the multiline message or the END message (if used) being sent.

The SH function shall exit STRS and enter:

(1) The SIWS (source idle wait state) within t_2 if either:

(a) The ATN message is true and neither CACS nor CTRS is active

(b) Or the ATN message is false and neither TACS nor SPAS is active.

NOTE: This implies an asynchronous interrupt (see 2.12.5).

(2) The SWNS (source wait for new cycle state) if the DAC message is true

2.3.3.5 SWNS (Source Wait for New Cycle State). In SWNS the SH function is waiting for

Table 2
SH Mnemonics

Messages	Interface States
pon = power on	SIDS = source idle state
nba = new byte available	SGNS = source generate state
ATN = attention	SDYS = source delay state
RFD = ready for data	STRS = source transfer state
DAC = data accepted	SWNS = source wait for new cycle state
	SIWS = source idle wait state
	(TACS) = talker active state (T function)
	(SPAS) = serial poll active state (T function)
	(CACS) = controller active state (C function)
	(CTRS) = controller transfer state (C function)

Table 3
SH Message Outputs

SH State	Remote Message Sent DAV	Device Function (DF) Interaction
SIDS	(F)	DF can change remote multiline messages
SGNS	F	DF can change remote multiline messages
SDYS	F	DAB, EOS multiline, and END messages shall not change
STRS	T	DAB, EOS multiline, and END messages shall not change
SWNS	T or F	DF requested to change multiline messages
SIWS	(F)	DF requested to change multiline messages

Table 4
Allowable Subsets to SH Function

Identi- fication	Description	States Omitted	Other Requirements	Other Function Subsets Required
SH0	no capability	all	none	none
SH1	complete capability	none	none	T1-T8, TE1-TE8, or C5-C28

Table 5
AH Mnemonics

Messages	Interface States
pon = power on	AIDS = acceptor idle state
rdy = ready for next message	ANRS = acceptor not ready state
tcs = take control synchronously*	ACRS = acceptor ready state
ATN = attention	ACDS = accept data state
DAV = data valid	AWNS = acceptor wait for new cycle state
	(LADS) = listener addressed state (L function)
	(LACS) = listener active state (L function)

*See the first paragraph of Section 2.12.3.7

the device to start a new message generation cycle.

In SWNS the SH function may send the DAV message true or false. In this state the device may change the multiline message being sent.

The SH function shall exit SWNS and enter:

- (1) The SGNS if the nba message is false
- (2) The SIWS within t_2 if either:

(a) The ATN message is true and neither CACS nor CTRS is active

(b) Or the ATN message is false and neither TACS nor SPAS is active.

2.3.3.6 SIWS (Source Idle Wait State). In SIWS the SH function is not active in the external message byte transfer process but is active in the internal process of waiting for the device to start a new message generation cycle. This SIWS allows a sequence of message byte transfers to be interrupted without loss of data over the interface while at the same time the device may continue to prepare for the new (next) message byte generation cycle.

In SIWS the DAV message shall be sent passive false.

The SH function shall exist SIWS and enter:

- (1) The SIDS if the nba message is false
- (2) The SWNS if either:

- (a) The TACS is active
- (b) Or the SPAS is active
- (c) Or the CACS is active

2.3.4 SH Function Allowable Subsets. The

only allowable subsets to the SH function shall be those listed in Table 4.

2.3.5 Additional SH Function Requirements and Guidelines. The nba true message indicates the device has generated a (new) message byte and made it available on the interface signal lines.

The nba message shall become true only in SIDS or SGNS. The nba message may become false in any other SH states.

The expression for interruption $(ATN \wedge (\overline{CACS}) \vee (\overline{CTRS})) \vee (ATN \wedge (\overline{TACS}) \vee (\overline{SPAS}))$ may be substituted by $(\overline{TACS}) \wedge (\overline{SPAS}) \wedge (\overline{CACS}) \wedge (\overline{CTRS})$ if the transition of the latter expression can be effected within t_2 after the change of ATN.

2.4 AH (Acceptor Handshake) Interface Function

2.4.1 General Description. The AH function provides a device with the capability to guarantee proper reception of remote multiline messages. An interlocked handshake sequence between an SH function and one or more AH functions (each contained within separate devices) guarantees asynchronous transfer of each message byte. An AH function may delay either the initiation of, or termination of, a multiline message transfer until prepared to continue with the transfer process. The AH function utilizes the DAV, RFD, and DAC messages to effect each message byte transfer.

Table 6
AH Message Outputs

AH State	Remote Message Sent		Device Function (DF) Interaction
	RFD	DAC	
AIDS	(T)	(T)	DF cannot receive remote multiline or END messages
ANRS	F	F	DF cannot receive remote multiline or END messages
ACRS	(T)	F	DF cannot receive remote multiline or END messages
AWNS	F	(T)	DF cannot receive remote multiline or END messages
ACDS	F	F	DF can receive remote multiline or END messages if LACS is active

2.4.2 AH Function State Diagram. The AH interface function shall be implemented so as to perform according to the state diagram given in Fig 4 and the state descriptions given throughout 2.4. Table 5 specifies the set of messages and states required to effect transition from one active state to another. Table 6 specifies the messages that shall be sent and the device function interaction required while each state is active.

2.4.3 AH Function State Descriptions

2.4.3.1 AIDS (Acceptor Idle State). In AIDS the AH function is inactive and not engaged in the handshake cycle. The AH function powers on in AIDS.

In AIDS the RFD and DAC messages shall be sent passive true.

The AH function shall exit AIDS and enter the ANRS (acceptor not ready state) within t_2 if either:

- (1) The ATN message is true
- (2) Or LACS is active
- (3) Or LADS is active

2.4.3.2 ANRS (Acceptor Not Ready State).

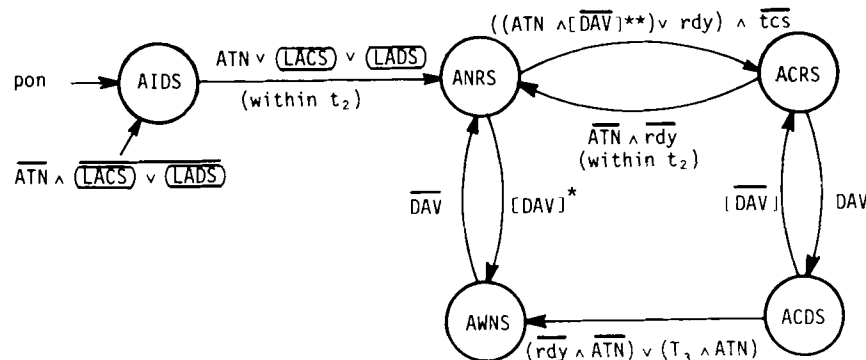
In ANRS the AH function indicates to the interface it has not yet prepared internally to continue with the handshake cycle.

In ANRS the RFD and DAC messages shall be sent false.

The AH Function shall exit ANRS and enter:

- (1) The ACRS if the tcs (take control synchronously) message is false (see the first

Fig 4
AH State Diagram



*This transition will never occur under normal interface operation; however, it may be implemented to simplify the interface function design.

**The need for the optional DAV term arises only for devices exiting AIDS and then only under take control synchronous (tcs) conditions (see 2.12.3.7). A minor revision to the controller function is under consideration that eliminates the need for the [DAV] term.

The addition of [DAV] prevents a data byte from being misinterpreted as an interface message (due to momentary coincidence of DAV and ATN) when ANRS is entered from AIDS. If ANRS has been active for more than 1.5 μ s, it can be assumed DAV is false.

paragraph of 2.12.3.7) and either:

- (a) The ATN message is true and the DAV message is false
- (b) Or the rdy (ready for next message) message is true

NOTE: Use of the DAV message is optional.

(2) The AIDS if the ATN message is false and neither:

- (a) The LADS is active
- (b) Nor LACS is active

(3) The AWNS if, optionally, the DAV message is true (note that this transition will never occur under normal interface operation)

2.4.3.3 ACRS (Acceptor Ready State).

In ACRS the AH function indicates to the interface that it is prepared to receive multiline messages.

In ACRS the DAC message shall be sent false and the RFD message shall be sent passive true.

The AH function shall exit ACRS and enter:

- (1) The ACDS (accept data state) if the DAV message is true
- (2) The AIDS if the ATN message is false and neither:
 - (a) The LADS is active
 - (b) Nor LACS is active
- (3) The ARNS within t_2 if both the ATN and the rdy messages are false

2.4.3.4 ACDS (Accept Data State). In ACDS the AH function indicates to the SH function that it shall maintain a valid message byte. This is the only state in which multiline messages on the DIO signal lines are valid. The ACDS indicates to the interface functions that an interface message is present and valid if the ATN message is true. The ACDS indicates to the device functions that a device dependent message is present and valid if LACS is active.

In ACDS the DAC and RFD messages shall be sent false.

The AH function shall exit the ACDS and enter:

- (1) The AWNS (acceptor wait for new cycle state) if either:

(a) The ATN message is true and a period of T_3 has elapsed

(b) Or the ATN and rdy messages are both false

(2) The AIDS if the ATN message is false and neither:

- (a) The LADS is active
- (b) Nor LACS is active

(3) The ACRS if, optionally, the DAV message is false (note that this transition can occur only when the controller takes control asynchronously)

2.4.3.5 AWNS (Acceptor Wait for New Cycle State). In AWNS the AH function indicates that it has received a multiline message byte.

In AWNS the RFD message shall be sent false and the DAC message shall be sent passive true.

The AH function shall exit the AWNS and enter:

- (1) The ANRS if DAV is false
- (2) The AIDS if the ATN message is false and neither:
 - (a) The LADS is active
 - (b) Nor LACS is active

2.4.4 Acceptor Function Allowable Subsets. The only allowable subsets to the AH function shall be those listed in Table 7.

2.4.5 Additional AH Function Requirements and Guidelines. The local message rdy shall not become false during the ACRS. The transition from ACRS to ANRS shall occur only at the time ATN becomes false.

The RFD message received by an SH function is the logical AND of all the RFD messages sent by all the active AH functions. Similarly, the DAC message received by an SH function is the logical AND of all the DAC messages sent by all the AH functions. The way in which the composite effects of multiple AH functions interact with an SH function to perform the logical AND function via the use

Table 7
Allowable Subsets to AH Function

Identification	Description	States Omitted	Other Requirements	Other Function Subsets Required
AH0	no capability	all	none	none
AH1	complete capability	none	none	none

of the NRFD and NDAC signal lines is explained further in 5.4.

Since interface functions need be designed only so as to perform according to the state diagrams specified, it is not required that exactly the states specified are the ones which exist in an implementation. One consequence of this statement is that interface function state transitions which are qualified by interface messages can occur after the message has been received as long as the RFD message is held false until they occur. The resulting performance cannot be distinguished from the performance of the specified diagrams in which the transitions shall occur while the interface message is being received. If this type of implementation is chosen, then the AH function should remain in ANRS even though the exit condition is true in order to hold the RFD message false (this is allowed by the last paragraph in 2.1.2.4).

In noisy environments, filter action on the incoming DAV message within a device can minimize false transitions to the ACDS state.

2.5 T (Talker) Interface Function (Includes Serial Poll Capabilities)

2.5.1 General Description. The T interface function provides a device with the capability to send device dependent data (including status data during a serial poll sequence) over

the interface to other devices. This capability exists only when the T interface function is addressed to talk.

There are two alternative versions of the function: one with and one without address extension. The normal T function uses a 1 byte address. The T interface function with address extension [hereinafter called a TE (extended talker) function] uses a 2 byte address. In all other respects, the capabilities of both versions are the same.

Only one of the two alternative T functions need be implemented in a specific device.

NOTE: Both the T function and TE function are described concurrently throughout 2.5 due to the extensive similarity between these two functions.

2.5.2 T Function State Diagrams. The T function shall be implemented so as to perform according to the state diagrams given in Fig 5 and the state descriptions given throughout 2.5. Table 8 specifies the set of messages and states required to effect transition from one active state to another. Table 9 specifies the messages that shall be sent and the device function interaction required while each state is active.

The TE function shall be implemented so as to perform according to the state diagrams given in Fig 6 and the state descriptions given

Fig 5
T State Diagram

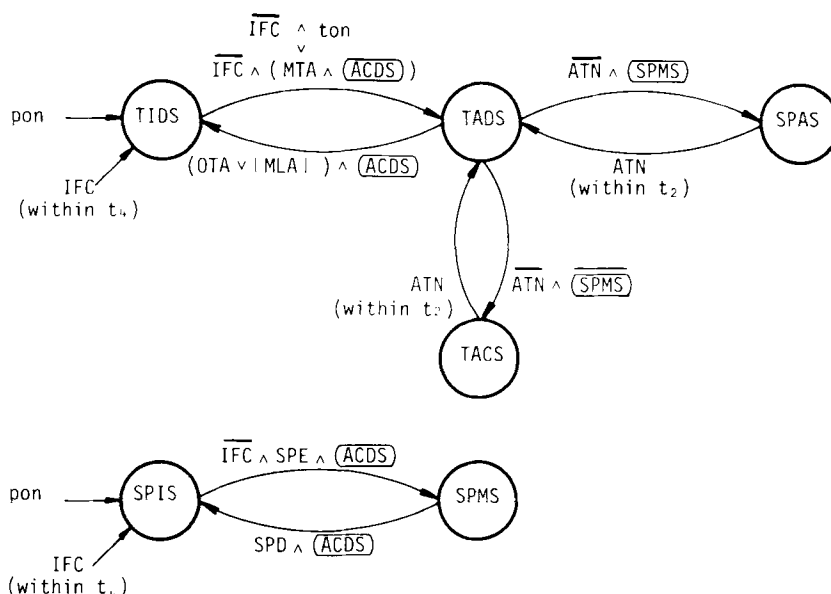


Table 8
T Mnemonics

Messages		Interface States	
pon	= power on	TIDS	= talker idle state
ton	= talk only	TADS	= talker addressed state
IFC	= interface clear	TACS	= talker active state
ATN	= attention	SPAS	= serial poll active state
MTA	= my talk address	SPIS	= serial poll idle state
SPE	= serial poll enable	SPMS	= serial poll mode state
SPD	= serial poll disable	(ACDS)	= accept data state (AH function)
OTA	= other talk address		
MLA	= my listen address		

Table 9
T or TE Message Outputs

T State	Qualifier	Remote Messages Sent*			Device Function (DF) Interaction
		Multiline	END	RQS	
TIDS		(NUL)	(F)	(F)	DF not allowed to send messages
TADS		(NUL)	(F)	(F)	DF not allowed to send messages
TACS		DAB [†] or EOS [†]	T or F [†]	(F)	DF can send DAB, EOS, or END message (if used) concurrent with DAB [†]
SPAS	APRS inactive	STB [†]	F or T	F	DF can send one STB message [‡]
SPAS	APRS active	STB [†]	F or T	T	DF can send one STB message [‡]

*See Table 38, Section 2.13.

[†]Messages enabled by the T function originating within the device functions.

[‡] Under SH control.

^{||}See 2.5.3.4.

Table 10
TE Mnemonics

Messages		Interface States	
pon	= power on	TIDS	= talker idle state
ton	= talk only	TADS	= talker addressed state
IFC	= interface clear	TACS	= talker active state
ATN	= attention	SPAS	= serial poll active state
MTA	= my talk address	TPIS	= talker primary idle state
OTA	= other talk address	TPAS	= talker primary addressed state
OSA	= other secondary address	SPIS	= serial poll idle state
PCG	= primary command group	SPMS	= serial poll mode state
SPE	= serial poll enable	(ACDS)	= accept data state (AH function)
SPD	= serial poll disable	(LPAS)	= listener primary addressed state (L function)
MSA	= my secondary address		