Recognized as an
American National Standard

IEEE
Std 1118-1990

IEEE Standard Microcontroller System Serial Control Bus

Sponsor

Technical Committee 8
of the IEEE Instrumentation
and Measurement Society

Approved December 6, 1990

IEEE Standards Board

Approved May 17, 1991

American National Standards Institute

Abstract: A serial control bus for interdevice/intrabuilding as well as intrasite interconnection of microcontrollers is described. The bus, which is defined for (but not limited to) microcontrollers and devices with limited reprogrammability, provides a multidrop bit-serial communication protocol that will allow the interconnection of distributed independently manufactured devices. The protocol is optimized for instrumentation, distributed data acquisition systems, control devices, and test and measurement. Specifications for a common architecture, generic bus services, system management, data link, and several physical media are provided. The serial control bus expands upon BITBUS™ without making existing devices obsolete. System reliability has been enhanced by the addition of a system management layer, and generic bus services have been expanded.

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

© 1991 by the Institute of Electrical and Electronics
Engineers, Inc.
All rights reserved. Published 1991
Printed in the United States of America

Library of Congress Cataloging in Publication Data
will be found on the following page.

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.

™ BITBUS is a trademark of Intel

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

IEEE Standards documents are developed within the Technical Committees of the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason IEEE and the members of its technical committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

IEEE Standards documents are adopted by the Institute of Electrical and Electronics Engineers without regard to whether their adoption may involve patents on articles, materials, or processes. Such adoption does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the standards documents.

Library of Congress Cataloging in Publication Data

IEEE standard microcontroller system serial control bus / sponsor, Technical Committee 8 of the IEEE Instrumentation and Measurement Society.
p. cm.
"IEEE Std 1118-1990."
"Approved December 6, 1990, IEEE Standards Board; approved May 17, 1991, American National Standards Institute."
ISBN 1-55937-096-3
1. Microcomputers—Buses—Standards. I. IEEE Instrumentation and Measurement Society. Technical Committee 8. II. IEEE Standards Board. III. American National Standards Institute.
TK7895.B87I44 1991
621.39'16—dc20 91-25415
CIP

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

Foreword

(This Foreword is not a part of IEEE Std 1118-1990, IEEE Standard Microcontroller System Serial Control Bus.)

In 1984, Intel recognized a need for a low-cost medium for connecting microcontrollers where distributed intelligent control had not existed. Intel called this new distributed control architecture BITBUS. Over the ensuing years, BITBUS became a defacto standard. To continue an open architecture philosophy, a consortium of companies formed a working group to address standardization requirements, openness to integration, and backward capability to existing products. This consortium included contributions from the listed companies during the development of the specification.

AMP	Horner Electric	Micro Switch
Datem	Intel	National Semiconductor
Digital Equipment Corp.	Kaye Instruments	NCR
The Foxboro Company	Lantek	Phoenix Contact
Future Concepts	Manhattan Electric Cable Corp.	RoBanks

The primary benefits that were sought in the standardization of this serial bus were system reliability, system management, expanded generic bus and Data Link Layer services, and alternate media.

IEEE Std 1118-1990 expands upon BITBUS without obsoleting existing devices. System reliability is enhanced by adding a System Management Layer. Generic Bus Services has been expanded. Some of the new services include a Define service and Suspend/Resume Task services.

The Data Link Layer has been expanded with an idle specification that increases Hamming distance and improves error detection. It now includes support for broadcast, multicast, and LSAPS. Portions of the HDLC protocols, not implemented in the original BITBUS specification, have also been added. Some of these are Identification service (XID) and Unnumbered Information (UI).

The alternate medium provides such new features as long distance operation without repeaters, operation in noisy environments, media redundancy, and hardware reset.

The goals of an open architecture and future growth of the standard were met by the working group as illustrated by the above examples. As this standard goes to press, a Project Authorization Request (PAR) for a Type C Alternate Medium is being proposed.

In 1986, Technical Committee 8 (TC 8) of the IEEE Instrumentation and Measurement Society sponsored a PAR (P1118) to develop a serial instrumentation interface. A working group proceeded with the overview functional requirements and architecture document. Sections were written by members of the working group, reviewed, and collated into the final standard. In 1990, the draft for Sponsor ballot was presented and after passing presented in September 1990 at the IEEE Standards Board. The standard is written with regard to ISO/IEC documentation requirements to support later submission to the ISO/IEC.

There were many contributors to the standard review process, but the following members deserve special mention for their active participation:

Nathan Tobel, *Draft Editor*
Shanker Munshani, *Original Secretary of Working Group*
Marge Merrick, *Original Study Group Chair*
Donald Ware, *Original Sponsor of PAR*

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

At the time that this standard was completed, the P1118 Working Group had the following membership:

R. Anderson, *Co-Chair* **D. Ware,** *Co-Chair*

L. Ames
J. Christie
J. Gallup
J. Gehenio
D. Gibson
R. Goldschmidt
P. Horner
E. Piltzecker, Jr.
N. Tobel

The working group was an outstanding example of individuals working and resolving issues together for a common goal.

At the time that it approved this document for submission to the IEEE Standards Board, the Balloting Commitee had the following membership:

L. Ames
R. Anderson
W. Bryan
J. Christie
B. Fisher
F. Frenquelli
J. Gallup
J. Gehenio
B. Goldschmidt
P. Horner
R. Landman
T. Phinney
E. Piltzecher, Jr.
B. Price
L. Sollman
N. Tobel
D. Ware
A. Weinstein

The following people also contributed to the development of this standard:

K. Boegli
C. Brill
B. Harris
R. Haubner
V. Jacobson
C. Lewis
M. Merrick
S. Munshani
G. Murdock
T. Phinney
M. Simms
C. Straub
B. Wells

When the IEEE Standards Board approved this standard on December 6, 1990, it had the following membership:

Marco W. Migliaro, *Chairman* **James M. Daly,** *Vice Chairman*
Andrew G. Salem, *Secretary*

Dennis Bodson
Paul L. Borrill
Fletcher J. Buckley
Allen L. Clapp
Stephen R. Dillon
Donald C. Fleckenstein
Jay Forster*
Thomas L. Hannan
Kenneth D. Hendrix
John W. Horch
Joseph L. Koepfinger*
Irving Kolodny
Michael A. Lawler
Donald J. Loughry
John E. May, Jr.
Lawrence V. McCall
L. Bruce McClung
Donald T. Michael*
Stig Nilsson
Roy T. Oishi
Gary S. Robinson
Terrance R. Whittemore
Donald W. Zipse

*Member Emeritus

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

Contents

SECTION PAGE

1. Overview and Architecture 1-1
   1.1 Introduction 1-1
   1.2 Architecture 1-5
   1.3 Bus Topology 1-11
   1.4 Layer Descriptions 1-13
   1.5 Conformance 1-15

2. Generic Bus Services — Service Specifications 2-1
   2.1 Introduction 2-1
   2.2 Architecture 2-2
   2.3 Functional Specifications 2-3
   2.4 User-Generic Bus Services Interface 2-5
   2.5 Conformance 2-31

3. Generic Bus Services — Protocol and Procedures 3-1
   3.1 Introduction 3-1
   3.2 Protocol Data Unit Structure 3-2
   3.3 Elements of Protocol 3-9
   3.4 Generic Bus Services Procedures 3-38
   3.5 Generic Bus Services — Operating System 3-44

4. System Management — Service Specifications 4-1
   4.1 Introduction 4-1
   4.2 Architecture 4-2
   4.3 Functional Specifications 4-4
   4.4 User-System Management Interface 4-6
   4.5 Conformance 4-21

5. System Management — Protocol and Procedures 5-1
   5.1 Introduction 5-1
   5.2 Protocol Data Unit Structure 5-2
   5.3 Elements of Protocol 5-13
   5.4 System Management Procedures 5-32

6. System Management Interfaces 6-1
   6.1 Introduction 6-1
   6.2 Application Layer Management Interface <ALMI> 6-2
   6.3 Data Link Layer Management Interface <LLMI> 6-6
   6.4 Physical Layer Management Interface <PLMI> 6-10

7. Data Link Layer — Service Specifications 7-1
   7.1 Introduction 7-1
   7.2 Data Link Layer-Upper Layer Interface 7-2
   7.3 Conformance 7-15

8. Data Link Layer — Protocol and Procedures 8-1
   8.1 Introduction 8-1
   8.2 Protocol Data Unit Structure 8-2
   8.3 Elements of Protocol 8-6
   8.4 Data Link Layer Procedures 8-22

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

SECTION PAGE

9. Physical Layer — Service Specifications 9-1
9.1 Introduction 9-1
9.2 Data Link Layer-Physical Layer Interface 9-2

10. Twisted Pair Type A Physical Layer 10-1
10.1 Introduction 10-1
10.2 Functional Specifications 10-2
10.3 Electrical Specifications 10-5
10.4 Bus Topology 10-7
10.5 Mechanical Specifications 10-8
10.6 Repeaters 10-9

11. Twister Pair Type B Physical Layer 11-1
11.1 Introduction 11-1
11.2 Functional Specifications 11-2
11.3 Electrical Specifications 11-3
11.4 Bus Topology 11-4
11.5 Mechanical Specifications 11-5
11.6 Repeaters 11-6

FIGURES

Fig 1.1-1 Applications 1-2
Fig 1.1-2 Documentation Structure 1-3
Fig 1.2-1 IEEE Std 1118-1990 Reference Model 1-6
Fig 1.2-2 Example of Data Transfer 1-8
Fig 1.2-3 Datagram Data Transfer 1-9
Fig 1.2-4 Acknowledged Datagram and Connection-Oriented Data Transfer 1-10
Fig 1.2-5 Layer Management Interface Primitive 1-11
Fig 1.3-1 Examples of Network Topology 1-12
Fig 2.1-1 Relationship of Section 2 to IEEE Std 1118-1990 2-1
Fig 2.2-1 Memory and I/O Model 2-2
Fig 2.4-1 Reset Device 2-6
Fig 2.4-2 Mark Device Offline 2-7
Fig 2.4-3 Read Device Information Service 2-8
Fig 2.4-4 GBS Protect Service 2-9
Fig 2.4-5 Get Time Service 2-10
Fig 2.4-6 Set Time Service 2-11
Fig 2.4-7 Upload Data Service 2-12
Fig 2.4-8 Download Data Service 2-13
Fig 2.4-9 Upload Code Service 2-14
Fig 2.4-10 Download Code Service 2-15
Fig 2.4-11 Read Scratch Pad Service 2-16
Fig 2.4-12 Write Scratch Pad Service 2-17
Fig 2.4-13 Read Special Function Register Service 2-18
Fig 2.4-14 Write Special Function Register Service 2-19
Fig 2.4-15 Read I/O Service 2-20
Fig 2.4-16 Write I/O Service 2-21
Fig 2.4-17 Logical I/O Service 2-22
Fig 2.4-18 Create Task Service 2-23
Fig 2.4-19 Delete Task Service 2-24
Fig 2.4-20 Suspend/Resume Task Service 2-25

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

SECTION PAGE

Fig 2.4-21 Get Task ID Service 2-26
Fig 2.4-22 Read Task Function IDs Service 2-27
Fig 2.4-23 Message Service 2-28
Fig 2.4-24 Define Service 2-29
Fig 2.4-25 Call Service 2-30
Fig 3.1-1 Relationship of Section 3 to IEEE Std 1118-1990 3-1
Fig 3.2-1 GBS_PDU Format 3-2
Fig 3.2-2 GBS Command Codes 3-4
Fig 3.2-3 GBS Response Codes 3-5
Fig 3.2-4 create_task Command Argument 3-5
Fig 3.2-5 Function ID Codes 3-6
Fig 3.2-6 Protect Codes 3-6
Fig 3.2-7 I/O Command Arguments 3-6
Fig 3.2-8 Linear Access Memory Command Arguments 3-7
Fig 3.2-9 Random Access Memory Command Arguments 3-7
Fig 3.2-10 Special Function Register Arguments 3-7
Fig 3.2-11 Device Information Command Arguments 3-8
Fig 3.2-12 Time Command Arguments 3-8
Fig 3.2-13 memory_page Command Argument 3-9
Fig 3.2-14 Define Command Arguments 3-9
Fig 3.3-1 Master Protocol State Diagram 3-10
Fig 3.3-2 POLL_LIST 3-12
Fig 3.3-3 CONNECT_DEVICE 3-13
Fig 3.3-4 FORMAT_PDU 3-14
Fig 3.3-5 RESET 3-15
Fig 3.3-6 IDENT 3-15
Fig 3.3-7 PROTECT 3-15
Fig 3.3-8 TIME-GET 3-16
Fig 3.3-9 TIME-SET 3-16
Fig 3.3-10 UPLOAD-DATA 3-16
Fig 3.3-11 DNLOAD-DATA 3-17
Fig 3.3-12 UPLOAD-CODE 3-17
Fig 3.3-13 DNLOAD-CODE 3-17
Fig 3.3-14 READ-PAD 3-18
Fig 3.3-15 WRITE-PAD 3-18
Fig 3.3-16 READ-SFR 3-18
Fig 3.3-17 WRITE-SFR 3-19
Fig 3.3-18 READ-I/O 3-19
Fig 3.3-19 WRITE-I/O 3-19
Fig 3.3-20 LOGICAL-I/O 3-20
Fig 3.3-21 CREATE 3-20
Fig 3.3-22 DELETE 3-20
Fig 3.3-23 TASK-S/R 3-21
Fig 3.3-24 READ-FIDS 3-21
Fig 3.3-25 GET-ID 3-21
Fig 3.3-26 DEFINE 3-22
Fig 3.3-27 CALL 3-22
Fig 3.3-28 Slave Protocol State Diagram 3-23
Fig 3.3-29 GBS_COMMAND 3-24
Fig 3.3-30 RESET_STATION 3-25
Fig 3.3-31 CREATE_TASK 3-25
Fig 3.3-32 DELETE_TASK 3-26

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

SECTION PAGE

Fig 3.3-33 READ_FUNCTION_IDS 3-26
Fig 3.3-34 GBS_PROTECT 3-27
Fig 3.3-35 READ_IO 3-27
Fig 3.3-36 WRITE_IO 3-28
Fig 3.3-37 SET_IO 3-28
Fig 3.3-38 UPLOAD_MEMORY 3-29
Fig 3.3-39 DOWNLOAD_MEMORY 3-29
Fig 3.3-40 OR_IO 3-30
Fig 3.3-41 AND_IO 3-30
Fig 3.3-42 XOR_IO 3-31
Fig 3.3-43 READ_SCRATCHPAD 3-31
Fig 3.3-44 WRITE_SCRATCHPAD 3-32
Fig 3.3-45 DEVICE_INFO 3-32
Fig 3.3-46 OFFLINE 3-32
Fig 3.3-47 UPLOAD_CODE 3-33
Fig 3.3-48 DOWNLOAD_CODE 3-33
Fig 3.3-49 READ_REGISTER 3-34
Fig 3.3-50 WRITE_REGISTER 3-34
Fig 3.3-51 GET_TIME 3-35
Fig 3.3-52 SET_TIME 3-35
Fig 3.3-53 SUSPEND_TASK 3-36
Fig 3.3-54 RESUME_TASK 3-36
Fig 3.3-55 DEFINE 3-37
Fig 3.3-56 GET_ID 3-37
Fig 3.4-1 GBS Message Sequence 3-38
Fig 4.1-1 Relationship of Section 4 to IEEE Std 1118-1990 4-1
Fig 4.2-1 System Management Internals 4-3
Fig 4.4-1 Set Value Service 4-7
Fig 4.4-2 Get Value Service 4-10
Fig 4.4-3 Action Service 4-13
Fig 4.4-4 Mastership Control Service 4-16
Fig 4.4-5 Identification Service 4-18
Fig 4.4-6 Communications Test Service 4-19
Fig 4.4-7 Event Report Service 4-20
Fig 5.1-1 Relationship of Section 5 to IEEE Std 1118-1990 5-1
Fig 5.3-1 Master Basic Protocol State Diagram 5-14
Fig 5.3-2 Master Local Protocol State Diagram 5-15
Fig 5.3-3 Master Remote Protocol State Diagram 5-16
Fig 5.3-4 Slave Basic Protocol State Diagram 5-18
Fig 5.3-5 LME Set Value State Diagram 5-19
Fig 5.3-6 LME Get Value State Diagram 5-20
Fig 5.3-7 LME Action State Diagram 5-21
Fig 5.3-8 Bus Mastership Transfer State Diagram 5-22
Fig 5.3-9 Initial Master State Diagram 5-22
Fig 5.3-10 Active Master State Diagram 5-23
Fig 5.3-11 Altenate Master State Diagram 5-24
Fig 5.3-12 Master Test Protocol State Diagram 5-27
Fig 5.3-13 Slave Test Protocol State Diagram 5-28
Fig 5.3-14 Overview of Identification Service 5-30
Fig 5.3-15 Identification Protocol State Diagram in Master Device 5-31
Fig 5.3-16 Identification Protocol State Diagram in Slave Device 5-32
Fig 5.4-1 Set Value Procedure 5-33

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

SECTION PAGE

Fig 5.4-2 Get Value Procedure 5-33
Fig 5.4-3 Action Procedure 5-33
Fig 5.4-4 Mastership Passing Procedure 5-34
Fig 5.4-5 Data Link Test Procedure 5-34
Fig 5.4-6 Identification Procedure 5-35
Fig 6.1-1 Relationship of Section 6 to IEEE Std 1118-1990 6-1
Fig 6.1-2 System Management Internals 6-2
Fig 6.2-1 Set Value Service 6-3
Fig 6.2-2 Get Value Service 6-4
Fig 6.2-3 Action Service 6-5
Fig 6.2-4 Event Service 6-6
Fig 6.3-1 Set Value Service 6-7
Fig 6.3-2 Get Value Service 6-8
Fig 6.3-3 Action Service 6-9
Fig 6.3-4 Event Service 6-10
Fig 6.4-1 Set Value Service 6-11
Fig 6.4-2 Get Value Service 6-12
Fig 6.4-3 Action Service 6-13
Fig 6.4-4 Event Service 6-14
Fig 7.1-1 Relationship of Section 7 to IEEE Std 1118-1990 7-1
Fig 7.2-1 Datagram Service 7-3
Fig 7.2-2 Acknowledged Datagram Service 7-4
Fig 7.2-3 Connection Service 7-5
Fig 7.2-4 Connection-Oriented Data Transfer 7-7
Fig 7.2-5 Data Transfer with Response 7-8
Fig 7.2-6 Data Transfer with Response 7-9
Fig 7.2-7 Connection Termination Services 7-11
Fig 7.2-8 Identification Service 7-13
Fig 7.2-9 Data Link Test Service 7-14
Fig 8.1-1 Relationship of Section 8 to IEEE Std 1118-1990 8-1
Fig 8.3-1 Sequence Count Notation 8-9
Fig 8.3-2 Master State Diagram 8-10
Fig 8.3-3 Master Operation in NDM State 8-11
Fig 8.3-4 Master Operation While Connecting 8-12
Fig 8.3-5 Master Operation in NRM State 8-13
Fig 8.3-6 Process TEST Response 8-14
Fig 8.3-7 Process XID Response 8-14
Fig 8.3-8 Process ACKDATA Response 8-15
Fig 8.3-9 Process RSPDATA Response 8-16
Fig 8.3-10 Master Operation While Disconnecting 8-17
Fig 8.3-11 Slave State Diagram 8-18
Fig 8.3-12 Slave State Table 8-18
Fig 8.3-13 Slave Operation in NDM 8-19
Fig 8.3-14 Slave Operation in NRM 8-20
Fig 8.3-15 Process I, RR, and RNR Frames 8-21
Fig 8.4-1 Datagram 8-23
Fig 8.4-2 Acknowledged Datagram 8-23
Fig 8.4-3 Connection Synchronization Sequence 8-24
Fig 8.4-4 Send Data and Request Reply 8-25
Fig 8.4-5 Poll Slave 8-26
Fig 8.4-6 Test Slave Buffer Status 8-26
Fig 8.4-7 Disconnect Sequence 8-27

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

SECTION PAGE

Fig 8.4-8 Identification Procedure 8-28
Fig 8.4-9 Data Link Test Procedure 8-29
Fig 9.1-1 Relationship of Section 9 to IEEE Std 1118-1990 9-1
Fig 9.2-1 Data Link Layer-Physical Layer Interface 9-2
Fig 10.1-1 Relationship of Section 10 to IEEE Std 1118-1990 10-1
Fig 10.2-1 Symbol Encoding 10-3
Fig 10.2-2 Symbol Decoding 10-3
Fig 10.3-1 Minimum Rise Time Circuit 10-5
Fig 10.3-2 Maximum Rise Time Circuit 10-6
Fig 10.4-1 Twisted Pair Type A Maximum Limits 10-7
Fig 10.5-1 Contact Pin Arrangement 10-8
Fig 10.5-2 Connector Pin Assignments 10-9
Fig 10.6-1 Repeater State Diagram 10-10
Fig 11.1-1 Relationship of Section 11 to IEEE Std 1118-1990 11-1
Fig 11.2-1 Symbol Encoding 11-3
Fig 11.2-2 Symbol Decoding 11-3
Fig 11.3-1 Device Electrical Characteristics 11-5
Fig 11-3-2 Minimum Rise Time Circuit 11-5
Fig 11.3-3 Maximum Rise Time Circuit 11-6
Fig 11.4-1 Twisted Pair Type B Maximum Limits 11-7
Fig 11.5-1 Device Connector 11-9
Fig 11.5-2 Cable Connector 11-9
Fig 11.5-3 Connector Pin Assignments 11-10
Fig 11.6-1 Repeater State Diagram 11-11

Appendix Implementation Guidelines A-1

A1. Conformance A-1

A2. Physical Layer A-1
A2.1 Type Selection A-1
A2.2 Encoding A-1
A2.3 Optional Functions A-2
A2.4 Unit Loads A-2
A2.5 Physical Media Spurs A-3
A2.6 Drops A-3
A2.7 Static Discharge A-3
A2.8 Bus Powered Devices A-4
A2.9 Signal Reference A-4
A2.10 Self-Directing Repeaters A-5
A2.11 Hardware Reset A-5

A3. Generic Bus Services A-6
A3.1 Generic Bus Services Memory Model A-6
A4.2 Message Structure A-7
A4.3 Intradevice Task Messaging A-7
A4.4 Protocol Implementation A-8
A4.5 Example A-8
A4.6 Extensions and Routing A-9

A4. System Management A-13
A4.1 Message Structure A-13
A4.2 Use of System Management A-13

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

FIGURES PAGE

Fig A-1 NRZI Encoding A-2
Fig A-2 Avoiding Spurs A-3
Fig A-3 Power Protection Technique A-4
Fig A-4 Grounding Arrangements A-5
Fig A-5 GBS Frame Structure A-7
Fig A-6 Route: Master to Slave A-9
Fig A-7 Route: Master Extension to Slave A-10
Fig A-8 Route: Master to Slave Extension A-10
Fig A-9 Route: Master Extension to Slave Extension A-10
Fig A-10 Route from Serial Interface A-11
Fig A-11 Route from Parallel Interface, Devices 1–254 ..... A-11
Fig A-12 Route from Parallel Interface, Devices 0 and 255 ..... A-12
Fig A-13 Route from Local Task, Devices 1–254 ..... A-12
Fig A-14 Route from Local Task, Devices 0 and 255 ..... A-12
Fig A-15 System Management Message Structure A-13
Fig A-16 All Device Reset Request A-14
Fig A-17 Read Buffer Size Request A-14
Fig A-18 Read Buffer Size Response A-14

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

1. Overview and Architecture

1.1 Introduction. This standard describes a serial control bus for interdevice/intrabuilding interconnection of microcontrollers. Intrasite interconnection is also provided.

This standard, which focuses on OSI layers 1, 2, and 7, may be used with other standards and practices to effect total solutions to specific applications.

1.1.1 Scope and Purpose. This standard defines an interconnect bus for (but not limited to) microcontrollers and devices with limited reprogrammability. This bus provides a multi-drop bit-serial communication protocol that will allow the interconnection of distributed, independently manufactured devices. The protocol is optimized for instrumentation, distributed data acquisition systems, control devices, and test and measurement.

The purpose of Section 1 is to serve as the foundation for the standard. It contains the overall definition of the bus network as well as a reference model for protocol sections. Protocol descriptions are contained in later sections of this document. This section contains the basis for those descriptions, such as the primitives employed to describe interactions between layers in the protocol reference model. An important part of this section is a description of what is meant by conformance to IEEE Std 1118-1990.

1.1.2 Key Concepts. The bus described herein is distinguished from other types of bit-serial data-communication buses in that it is optimized for the interdevice/intrabuilding/intrasite interconnection of microcontrollers. This includes optimization for messages ≤ 255 bytes (with a typical message being less than 128 bytes) and the selection of signaling rates in the 50–500 KBaud range.

Another key concept is the intentional restriction of bus characteristics to simplify network implementation. Examples of these restrictions are the limited selection of media and signaling rates.

1.1.3 Application and Support. This standard is technology-based, not application-based (see Fig 1.1-1). It is suitable for many different application types, primarily in the laboratory, test and measurement, and light manufacturing areas. But, it is also applicable to process control, building control, and distributed I/O ("RS 232-type" peripherals, data entry and display devices, sensors, and actuators). This standard may be used as a building block, along with other standards, in applications that could include, but are not limited to

(1) Instrumentation
(2) Test and measurement
(3) Distributed data acquisition systems
(4) Industrial automation
    (a) Manufacturing control
    (b) Process control
(5) Building controls
    (a) Residential
    (b) Commercial
    (c) Environmental
    (d) Access
    (e) Intrusion
(6) Distributed I/O
    (a) RS 232-type peripherals
    (b) Data entry and display devices
    (c) Sensors and actuators
    (d) Control devices
    (e) Human interface devices

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

This standard does not attempt to address all restrictions required by a specific application (such as building codes, national regulatory agencies, intrinsic safety codes, etc.).

```
<---------------------- TECHNOLOGY ----------------------->

                      |  IEEE Std  |
                      |  1118-1990 |                      ^
----------------------+------------+---------------------  |
                      |            |                      |
  INSTRUMENTATION     |            |                      |
                      |            |                      |
----------------------+------------+---------------------  A
                      |            |                      P
----------------------+------------+---------------------  P
                      |            |                      L
  PROCESS CONTROL     |            |                      I
                      |            |                      C
----------------------+------------+---------------------  A
                      |            |                      T
----------------------+------------+---------------------  I
                      |            |                      O
  MANUFACTURING       |            |                      N
  CONTROL             |            |                      S
                      |            |                      |
----------------------+------------+---------------------  |
                      |            |                      |
----------------------+------------+---------------------  |
                      |            |                      |
  OTHER               |            |                      |
                      |            |                      |
----------------------+------------+---------------------  |
                      |            |                      V
```

Fig 1.1-1
Applications

1.1.4 Documentation Organization. This document is organized into sections. These sections define a common architecture, generic bus services, system management, data link, and several different physical media. The sections are

(1) Section 1 — Overview and Architecture
(2) Section 2 — Generic Bus Services — Service Specifications
(3) Section 3 — Generic Bus Services — Protocol and Procedures
(4) Section 4 — System Management — Service Specifications
(5) Section 5 — System Management — Protocol and Procedures
(6) Section 6 — System Management — Interfaces
(7) Section 7 — Data Link Layer — Service Specifications
(8) Section 8 — Data Link Layer — Protocol and Procedures
(9) Section 9 — Physical Layer — Service Specifications
(10) Section 10 — Twisted-Pair Type A Physical Layer
(11) Section 11 — Twisted-Pair Type B Physical Layer
(12) Appendix A — Implementation Guidelines

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The relationship of these sections is shown in Fig 1.1-2.

```
      +-----------------------------------------------------+
      | Implementation Guidelines                           |
      | Appendix A                                          |
    +-+-----------------------------------------------------+ |
    | Overview and Architecture                             | |
    | Section 1                                             | |
  +-+----------------------+---------------------------+ | |
  | Generic Bus Services - | System Management -       | | |
  | Service Specifications | Service Specifications    | | |
  | Section 2              | Section 4                 | | |
  +------------------------+---------------------------+-----+ | |
  | Generic Bus Services - | System Management - |S M I| | |
  | Protocol and Procedures| Protocol and Proced-|y a n| | |
  | Section 3              | ures   Section 5    |s n t| | |
  +------------------------+---------------------+t a e| | |
  | Data Link Layer - Service Specifications     |e g r| | |
  | Section 7                                    |m e f| | |
  +----------------------------------------------+  m a| | |
  | Data Link Layer - Protocol and Procedures    |  e c| | |
  | Section 8                                    |  n e| | |
  +----------------------------------------------+  t s| | |
  | Physical Layer - Service Specifications      |     | | |
  | Section 9                                    |Sect.| |-+
  +----------------------------------------------+  6  | |
  | Media Dependent Physical Layer Specifications|     +-+
  | Section 10 and above                         |     |
  +----------------------------------------------+-----+
```

Fig 1.1-2
Documentation Structure

1.1.5 References. The following documents are referenced in this standard:

[1] EIA-485, Standard for Electrical Characteristics of Generators and Receivers for use in Balanced Digital Multipoint Systems. (April 1983.)[1]

[2] IEC 807-2 (1985 Part 2): Detail specification for a range of connectors with round contacts. Fixed solder contact types.[2]

[3] IEEE Std 100-1988, IEEE Standard Dictionary of Electrical and Electronics Terms.[3]

[1]EIA publications are available from the Electronics Industries Association, Engineering Department, 2001 Eye Street NW, Washington, DC 20006, USA.

[2]IEC publications are available from IEC Sales Department, Case Postale 131, 3 rue de Varembé, CH 1211, Genève 20, Switzerland/Suisse. IEC publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

[3]IEEE publications are available from the Institute of Electrical and Electronics Engineers, Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

[4] ISO 3309 : 1984, Information processing systems — Data communication — High-level data link control procedures — Frame structure.[4]

[5] ISO 4335 : 1987, Information processing systems — Data Communication — High-level data link control elements of procedures.

[6] ISO 7498 : 1984, Information processing systems — Open Systems Interconnection — Basic Reference Model.

[7] ISO 7809 : 1984, Information processing systems — Data communications — High-level data link control procedures — Consolidation of classes of procedures.

[8] ISO 8877 : 1987, Information processing systems — Interface connector and contact assignments for ISDN basic access interface located at reference points S and T.

1.1.6 Definitions. All definitions, except as specifically covered in this standard, shall be in accordance with IEEE Std 100-1984 [3].[5]

AE. Application entity.

ASAP. Application service access point.

bandwidth. The number of user data bytes (i.e., exclusive of communications overhead) that can be sent across the network per second.

BER. Bit error rate. The ratio of bits received in error to total bits received.

compatibility interface. A point at which hardware, logic, and signal levels are defined to allow the interconnection of independently designed and manufactured components.

FCS. Frame check sequence.

Hamming distance. The minimum number of bits that must be altered to change one valid message encoding into a different valid message encoding.

LSAP. Link service access point.

microcontroller. A microprocessor that integrates processor, memory, and I/O on a single semiconductor device.

PCI. Protocol control information.

PDU. Protocol data unit.

PHYSAP. Physical service access point.

response time. The total time necessary to send a message and receive an acknowledgement back at the sender, exclusive of application processing time. (Also see *system response time*).

[4]ISO publications are available from ISO, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse. ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.
[5]The numbers in brackets correspond to those of the references in 1.1.5.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

SAP. Service access point.

SDU. Service data unit.

system response time. The total time necessary to send a message and receive an acknowledgement back at the sender from every active device on the network (exclusive of application processing time).

transaction. Either (a) a datagram, or (b) a datagram and its associated acknowledgement.

TTP. Telephone twisted pair.

UCT. Universal coordinated time.

usable data rate. See *bandwidth*.

1.1.7 Conventions. To improve readability of this standard, conventions are defined for PDU formats, flow charts, and hexadecimal notation.

1.1.7.1 PDU Formats. Where PDU formats are shown, they are in the order transmitted, from left to right. Where bit layouts are shown, they have the most significant bit on the left (N.B. transmission occurs least significant bit first).

1.1.7.2 Flow Charts. The following conventions are used in flow charts in this standard:

(1) = indicates assignment (from right to left)
(2) == indicates test for equality
(3) != indicates test for inequality
(4) ++ indicates the increment operation

1.1.7.3 Hexadecimal Notation. Hexadecimal numbers are represented as follows:

(1) 0xNN represents an 8 bit value of hexadecimal NN
(2) 0xNNNN represents a 16 bit value of hexadecimal NNNN

1.2 Architecture. This section defines the IEEE Std 1118-1990 Reference Model.

1.2.1 Reference Model Description. The IEEE Std 1118-1990 Reference Model relates to the OSI Reference Model as shown in Fig 1.2-1. The applicable parts of the OSI Reference Model consist of the Physical Layer, the Data Link Layer, and certain generic Application Layer entities that have general applicability. These industry independent services (Generic Bus Services) are provided at the Application Layer while System Management is available at all layers.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

```
                OSI Reference Model    IEEE Std 1118 Reference Model

                +--------------+ - - - - - +-(A)--+---(A)---+
                |              |           |      |         |
              7 | APPLICATION  |           | GBS <A>   SM   |
                |              |           |      |         |
                +--------------+ - - - - - +-( )--+-( )-+   |
                |              |           |            |   |
              6 | PRESENTATION |           |            |   |
                |              |           |            |   |
                +--------------+           |            |   |
                |              |           |            |   |
              5 | SESSION      |           |            |   |
                |              |           |            |   |
                +--------------+           |   null     |   |
                |              |           |            |   |
              4 | TRANSPORT    |           |            |   |
                |              |           |            |   |
                +--------------+           |            |   |
                |              |           |            |   |
              3 | NETWORK      |           |            |   |
                |              |           |            |   |
                +--------------+ - - - - - +-(L)....(L)-+   |
                |              |           |            |   |
              2 | DATA LINK    |           | DATA LINK <L>  |
                |              |           |            |   |
                +--------------+ - - - - - +----(P)-----+   |
                |              |           |            |   |
              1 | PHYSICAL     |           | PHYSICAL  <P>  |
                |              |           |            |   |
            +--+--------------+--+ - - +--+------------+---+--+
            :                    :     :                      :
            :       MEDIUM       :     :   PHYSICAL MEDIUM    :
            :                    :     :                      :
            ---------------------+ - - +----------------------+
```

NOTES:
(1) Service Access Points (SAPs) are indicated by
(A) = Application Service Access Point (ASAP)
(L) = Data Link Service Access Point (LSAP)
(P) = Physical Service Access Point (PHYSAP)
(2) Layer Management Interfaces <LMIs> are indicated by
<A> = Application Layer Management Interface <ALMI>
<L> = Data Link Layer Management Interface <LLMI>
<P> = Physical Layer Management Interface <PLMI>
(3) GBS = Generic Bus Services.
(4) SM = System Management.
(5) No device is required to implement more than one LSAP.

Fig 1.2-1
IEEE Std 1118-1990 Reference Model

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

1.2.2 Service Access Points (SAPs). SAPs are logical interface ports that provide interconnection between layers. They typically have no physical implementation and exist only to provide a mechanism to describe the interface between layer entities.

Multiple data link service access points (LSAPs) are provided to support multiple higher layer users. A single physical service access point (PHYSAP) is provided to the Data Link Layer.

1.2.3 Service Primitives. In defining the services that one layer offers to another, formal service specification statements, called *service primitives*, are used. Service primitives describe the service that is being provided and specify the information that must be passed across the service access point interface to provide the service. For example, a request by a higher-layer entity (service user) of the Data Link layer (service provider) to send data to another higher-layer entity is written down as the following service primitive:

DL_DATA.request(slave_identifier, DL_SDU)

Similarly, the transfer of that data from the Data Link Layer to the destination higher-layer entity is described by this service primitive:

DL_DATA.indication(slave_identifier, DL_SDU)

The parenthesized parameters in the service primitives indicate *what* information must be passed across an interface, but they do not imply *how* that information is passed or formatted.

The list of service primitives, their parameters, and a description of their uses constitutes a complete specification of the services offered by a specific layer.

Fig 1.2-2 diagrams the relationship between service primitives and time-sequence diagrams. The example shows a datagram service being provided by the Data Link Layer and its representation as a time-sequence diagram. Henceforth, in this document, time-sequence diagrams are used to show the operation of service primitives.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

```
+--------------+             +--------------+
|              |             |              |
| Master       |             | Slave        |
| Higher-Layer |             | Higher-Layer |
| Entity       |             | Entity       |
|              |             |              |
+------+-------+             +------+-------+
       |                            ^
       |                            |
       | DL_DATA.request            | DL_DATA.indication
       |                            |
       V                            |
+-----(-)---------------------------(-)-----+
|                                           |
|             Data Link Layer               |
|                                           |
+-------------------------------------------+
```

Service Primitives

- - - is equivalent to - - -

```
  Master Higher-|   Data Link Layer    | Slave Higher-
  Layer Entity  |  (Service Provider)  | Layer Entity
                |                      |
DL_DATA.request |                      |
          \     |                      |
           \    |                      |
            \   |                      |
             \  |                      |
              -->|                      |
                 |                      |
                 |                      |
                 |                      |\
                 |                      | \
                 |                      |  \
                 |                      |   \
                 |                      |    |
                 |                      |    V
                 |                      | DL_DATA.indication
                 |                      |
```

Time Sequence

Fig 1.2-2
Example of Data Transfer

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

1.2.4 Service Primitive Types. In using service primitives to define a service specification, several distinct categories of primitives are used. These categories are called *service primitive types*. The four service primitive types are request, indication, response, and confirm.

Examples of the use of these service primitives are shown in the time-sequence diagrams in Figs 1.2-3 and 1.2-4. In these diagrams, the two layers are rearranged so that the service provider is between the two vertical lines, and the source and destination service users are to the left and right of the service provider. Each vertical line represents one SAP. Interaction between the service users and the service provider are represented by diagonal lines between the service users and the service provider. The passage of time is roughly indicated from top to bottom, although there is no time scale indicated.

```
Master        |                              | Slave        |
Service       |      Service Provider        | Service      |
User          |         (layer n)            | User         |
Entity        |                              | Entity       |
(layer n+1)   |                              | (layer n+1)  |
              |                              |              |
request       |                              |              |
      \       |                              |              |
       \      |                              |              |
        \     |                              |             T
         -->  |                              |             I
              |                              |             M
              |                              |\            E
              |                              | \            |
              |                              |  \           |
              |                              |   |          |
              |                              |   V          |
              |                              | indication   |
              |                              |              |
              |                              |             V
```

Fig 1.2-3
Datagram Data Transfer

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

```
Master      |                            | Slave       |
Service     |      Service Provider      | Service     |
User        |         (layer n)          | User        |
Entity      |                            | Entity      |
(layer n+1) |                            | (layer n+1) |
            |                            |             |
    request |                            |             |
      \     |                            |             |
       \    |                            |             |
        \   |                            |            T
         -->|                            |            I
            |                            |            M
            |                            |\           E
            |                            | \           |
            |                            |  \          |
            |                            |   |         |
            |                            |   V         |
            |                            | indication  |
            .                            .             .
            .                            .             .
            |                            | response    |
            |                            |     /       |
            |                            |    /        |
            |                            |   /         |
            |                            |<--          |
            |                            |             |
           /|                            |             |
          / |                            |             |
         /  |                            |             |
        |   |                            |             |
        V   |                            |             |
    confirm |                            |             |
            |                            |             |
            |                            |            V
```

Fig 1.2-4
Acknowledged Datagram and
Connection-Oriented Data Transfer

1.2.5 Layer Management Interface Primitive. The Layer Management Interface uses a different primitive than those used by the layer service specifications. Fig 1.2-5 diagrams the relationship between this primitive and an equivalent time-sequence diagram.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

```
+--------+             +------------+
|        |     request |            |
| Layer  |<------------| System     |
| Entity |------------>| Management |
|        | confirm     |            |
+--------+             +------------+

          Service Primitives


     - - - is equivalent to - - -


Local System      | Local Layer       |
Management (LME)  | Entity            |
                  |                   |
        request   |                   |
             \    |                   T
              \   |                   I
               -->|                   M
                  |                   E
                 /|                   |
                / |                   |
               |  |                   |
               V  |                   |
        confirm   |                   |
                  |                   V

                Time Sequence
```

Fig 1.2-5
Layer Management Interface Primitive

1.3 Bus Topology. This standard defines a multi-drop bus consisting of one or more bus segments interconnected by repeaters. Typically, networks are constructed as either linear networks or tree networks (see Fig 1.3-1). A bus segment is a passive physical medium to which two or more devices (master or slave) or repeaters are attached. The segment length is defined as the maximum distance the signal must travel without reamplification.

Physical media defined by this standard support a range of segment lengths. These segments can be interconnected by repeaters. The maximum number of addressable devices on the bus is 240.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

```
+-+                                 +-+
|T+----+--------+-----------+----+T|
+-+ +--+--+  +--+--+     +--+--+ +-+ +-----+ +-----+
    |     |  |     |     |     |     |     | |     |
    |  M  |  |  S  |     |  R  |     |  S  | |  S  |
    |     |  |     |     |     |     |     | |     |
    +-----+  +-----+ +-+ +--+--+     +--+--+ +--+--+ +-+
                     |T|----+-----------+-------+----+T|
                     +-+                              +-+
```

Example Linear Network

```
+-+                                                   +-+
|T+----+-------+-------+---------------+-------+----+T|
+-+ +--+--+ +--+--+ +--+--+         +--+--+ +--+--+ +-+
    |     | |     | |     |         |     | |     |
    |  R  | |  M  | |  S  |         |  R  | |  S  |
    |     | |     | |     |         |     | |     |
+-+ +--+--+ +-----+ +-----+ +-+ +-+ +--+--+ +-----+ +-+
|T+----+---------------+----+T| |T+----+-------+----+T|
+-+ +--+--+         +--+--+ +-+ +-+ +--+--+ +--+--+ +-+
    |     |         |     |         |     | |     |
    |  R  |         |  R  |         |  S  | |  S  |
    |     |         |     |         |     | |     |
+-+ +--+--+ +-+ +-+ +--+--+         +-----+ +-----+ +-+
|T+----+----+T| |T+----+-------+-------+-------+----+T|
+-+ +--+--+ +-+ +-+ +--+--+ +--+--+ +--+--+ +--+--+ +-+
    |     |         |     | |     | |     | |     |
    |  S  |         |  S  | |  S  | |  S  | |  S  |
    |     |         |     | |     | |     | |     |
    +-----+         +-----+ +-----+ +-----+ +-----+
```

Example Tree Network

```
                          LEGEND

+-----+                   +-----+
|     |                   |     |
|  M  |  MASTER DEVICE    |  S  |   SLAVE DEVICE
|     |                   |     |
+-----+                   +-----+

+-----+                   +-+
|     |                   |T|       TERMINATING
|  R  |  REPEATER         +-+       RESISTOR
|     |
+-----+                   ++        CONNECTOR
```

Fig 1.3-1
Examples of Network Topology

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

1.4 Layer Descriptions. This section gives a brief description of each of the layers of IEEE Std 1118-1990. The intent of presenting this information is to provide an overview of the standard and serve as a guide for reading other sections of this document.

1.4.1 Physical Layer. IEEE Std 1118-1990 Physical Layers provide

(1) Maximum compatibility
(2) Easy connection of devices into a system
(3) Protection from interconnection of incompatible devices
(4) Minimum number of physical media

Sections 10 and above of this standard define the characteristics of the Physical Layer for each specified medium. These characteristics include the physical medium, connectors, signal levels, and encoding mechanism. Each of these Physical Layers

(1) Provides a self-clocked, bit serial communications system
(2) Operates at one of the two data rates (but may operate at both)
(3) Provides a "compatibility interface" at the bottom of the Physical Layer ("media level") that includes an interconnection scheme. (A compatibility interface includes the connector, cable characteristics, and voltage/current signal parameters at the device port.)

1.4.1.1 Twisted Pair Type A. The twisted pair Type A Physical Layer defines the physical layer characteristics for use with twisted-pair wiring.

1.4.1.2 Twisted Pair Type B. An alternate twisted pair Physical Layer is defined suitable for use in industrial environments, including harsh and hazardous environments.

1.4.1.3 Other Physical Layers. Other physical layers may be specified to provide longer single-segment distances, more drops per segment, better noise immunity, and/or the option for powering devices from the same two wires that conduct the communication signals.

1.4.2 Data Link Layer. The Data Link Layer is responsible for providing the services required for communication across a single medium. These include services for transferring data on behalf of higher-layer entities and mechanisms that control the access to the physical medium.

The Data Link Layer used by this standard is a modified and restricted version of HDLC. The most significant modifications are

(1) A pre-frame synchronization field added in front of the opening flag to facilitate clock recovery.
(2) A period of no transitions between frames provides strong frame delimiting. This assists in achieving a Hamming distance of three.
(3) Octet alignment is required to detect errors that affect results of the zero insertion/deletion algorithm.

1.4.2.1 Services Provided to Upper Layers. The Data Link Layer provides three classes of data-transfer services and a uniform interface for upper-layer entities to use in accessing these services.

The three classes of service defined by this standard are

(1) Connectionless
(2) Connection oriented
(3) Device control

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

LSAPs are provided by the protocol. Some of the LSAPs are globally administered (e.g., System Management). The remainder are locally administered.

1.4.2.1.1 Connectionless Services. The connectionless services permit a message to be sent, or messages to be exchanged, without requiring a connection to be established. Two types of connectionless service are provided, datagram and acknowledged datagram.

The datagram service allows a message to be sent to any or all devices connected to the bus. The datagram service provides statistical integrity, but not guaranteed communications.

The acknowledged datagram service permits a message to be sent, and a response solicited, without requiring a connection to be established. The acknowledged datagram service provides high-integrity communications for a single data packet.

1.4.2.1.2 Connection-Oriented Services. The Data Link Layer uses connections to provide more reliable data transfers. A connection is a mechanism by which communication-state information is kept synchronized between two communicating devices. This common-state information provides, among other services, missing and duplicate message detection and, in cases of bad or missing messages, data retransmission.

1.4.2.1.3 Device Control Services. Device control services provide a mechanism to identify and test devices attached to the bus.

1.4.2.2 Access Control. The Data Link Layer is responsible for controlling access to the physical medium. A master-slave service is provided. It allows the exchange of packets between a master and one or more slave devices. The master is the only device which is capable of initiating transactions on the bus. Slave devices transmit only to the master, in response to a request from the master.

Alternate-master operation, consistent with the basic master-slave operation, is provided as an option. The procedure that defines the transfer of control to an alternate master is specified in the System Management section of this standard.

1.4.2.3 Additional Services. Broadcast (multicast to all stations) and generalized multicast (group addressing) services are provided.

The Data Link Layer also has certain additional functions by virtue of its position as the last frame-oriented entity before the physical medium. These functions include error detection and the control of optional redundant media. The error detection provided at the Data Link Layer is sufficient to achieve an undetected error rate that is less than or equal to one error every 20 years and a Hamming distance of three.

1.4.3 Generic Bus Services. This standard defines certain Application Layer entities that have general applicability and provide a set of industry-independent services to support general-purpose operations. Examples of these services are

(1) Reset
(2) Download
(3) Upload
(4) Message passing

1.4.4 System Management. System Management defines the services, protocol, standard data types, and common parameter definitions for accomplishing management of the communication system. These services include

(1) Device status including error and message counts
(2) Device and system control
(3) Device identification

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

1.5 Conformance. There are two types of conformance to this standard: component conformance and device conformance. *Describing, advertising, or promoting equipment as conforming to IEEE Std 1118-1990, in a limited or full sense, is equivalent to making a voluntary statement of conformance.*

1.5.1 Component Conformance. Component conformance defines the conformance at the individual component level. The areas of component conformance are

(1) Media
(2) Connector
(3) Transceiver (bus driver/receiver)
(4) Protocol implementing device (typically a microcontroller)

Criteria for component conformance are defined in the relevant sections of this standard. Statements of component conformance with IEEE Std 1118-1990 can be made, providing that there is a precise and easily visible statement of this conformance on the appropriate equipment or documentation. Component conformance statements related to this standard shall indicate the extent of both conformance and interoperability. Such components shall not say "IEEE Std 1118-1990 conformant," or use equivalent terms.

1.5.2 Device Conformance. Devices that meet IEEE Std 1118-1990 may be labeled as having either basic, extended, or universal conformance if they meet the requirements in the following sections.

1.5.2.1 Basic Conformance. Basic conformance with this standard requires the implementation of the mandatory portions of the Data Link Layer and at least one of the Physical Layers. Only products conforming to the above criteria may be identified as meeting "basic conformance" (e.g., "IEEE Std 1118-1990 basic conformant," or equivalent wording).

1.5.2.2 Extended Conformance. Extended conformance with this standard requires the implementation of the mandatory portions of the Data Link Layer, System Management, and Generic Bus Services required for extended conformance (see 2.5, 4.5, and 7.3), and at least one of the Physical Layers. Only products conforming to the above criteria may be identified as meeting "extended conformance" (e.g., "IEEE Std 1118-1990 extended conformant," or equivalent wording).

1.5.2.3 Universal Conformance. Universal conformance with this standard requires the implementation of all portions of the Data Link Layer, System Management, Generic Bus Services, and at least one of the Physical Layers. Only products conforming to the above criteria may be identified as meeting "universal conformance" (e.g., IEEE Std 1118-1990 universal conformant," or equivalent wording).

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

2. Generic Bus Services — Service Specifications

2.1 Introduction. This section defines the services provided by Generic Bus Services (GBS) to upper-level application layers. GBS is an optional part of this standard. If GBS is implemented, a minimum set of services is mandatory (see 2.5). The GBS specifications described in this section are intended to be used in conjunction with the other sections of this standard (see Fig 2.1-1).

```
      +-----------------------------------------------------+
      | Implementation Guidelines                           |
      | Appendix A                                          |
    +-+-----------------------------------------------------+ |
    | Overview and Architecture                             | |
    | Section 1                                             | |
  +-+-----------------------+---------------------------+ | |
  | Generic Bus Services - | System Management -       | | |
  | Service Specifications | Service Specifications    | | |
  | Section 2              | Section 4                 | | |
  +------------------------+---------------------------+-----+ | |
  | Generic Bus Services - | System Management - |S M I| | |
  | Protocol and Procedures| Protocol and Proced-|y a n| | |
  | Section 3              | ures   Section 5    |s n t| | |
  +------------------------+---------------------+t a e| | |
  | Data Link Layer - Service Specifications     |e g r| | |
  | Section 7                                    |m e f| | |
  |----------------------------------------------+  m a| | |
  | Data Link Layer - Protocol and Procedures    |  e c| | |
  | Section 8                                    |  n e| | |
  |----------------------------------------------+  t s| | |
  | Physical Layer - Service Specifications      |     | | |
  | Section 9                                    |Sect.| |-+
  +----------------------------------------------+  6  | |
  | Media Dependent Physical Layer Specifications|     +-+
  | Section 10 and above                         |     |
  +----------------------------------------------+-----+
```

Fig 2.1-1
Relationship of Section 2 to IEEE Std 1118-1990

2.1.1 Scope and Purpose. This section describes:

(1) An architecture for GBS including a reference model
(2) The functions performed by GBS
(3) The services provided by GBS to application entities including parameters passed and their definitions

2.1.2 Overview of Services. The following list summarizes the services provided by GBS:

(1) Device control services
  (a) Reset Device
  (b) Mark Device Offline
  (c) Read Device Information

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

(d) Enable/Disable Memory Space Access
(e) Get/Set Time

(2) Memory access services
(a) Upload/Download Data
(b) Upload/Download Code
(c) Read/Write Scratch Pad Memory
(d) Read/Write Special Function Registers

(3) I/O access services
(a) Read/Write I/O Location
(b) Logical I/O (AND, OR, XOR, SET)

(4) Task access services
(a) Create/Delete Task
(b) Suspend/Resume Task
(c) Get Task Function ID
(d) Read Task Function IDs
(e) Task Message Exchange

(5) Support for user extensions
(a) Define service
(b) Call service

2.1.3 Section Organization. This section is organized as follows:

(1) *2.1 Introduction.* Provides an overview of GBS.
(2) *2.2 Architecture.* Describes the logical model employed by GBS.
(3) *2.3 Functional Specifications.* Defines the functional characteristics of the four main aspects of GBS.
(4) *2.4 User-Generic Bus Services Interface.* Explains the services that this layer entity provides to tasks.
(5) *2.5 Conformance.* Explains the conformance classes that apply to GBS.

2.2 Architecture. GBS provides a standard set of high-level data and command interchange procedures for microcontrollers interconnected on a bit-serial network. GBS requires the services of a multitasking operating system, as well as the services of the IEEE Std 1118-1990 Data Link Layer. Tasks present within a microcontroller may be separately addressed for message routing. The memory model and the task model are described in an abstract manner and do not imply a particular implementation.

2.2.1 Memory and I/O Model. The GBS model is intended to provide maximum flexibility to support all processor architectures. To provide this flexibility, four octet-addressable memory spaces and a separate I/O space are supported (see Fig 2.2-1). Each address has two components, a page and an offset (displacement from the beginning of the page). Overmapping is permitted between any or all of these spaces. Some of these memory spaces may not be present.

	Address Size	
Name	Page	Offset
Code Space	16 bits	16 bits
Data Space	16 bits	16 bits
Scratch Pad Space	16 bits	8 bits
I/O Space	16 bits	8 bits
Special Function Register Space	16 bits	8 bits

Fig 2.2-1
Memory and I/O Model

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

2.2.1.1 Code and Data Memory Spaces. Individual octets within the code or data spaces are specified by a 32 bit address (16 bit page address and 16 bit offset address). The processor is assumed to fetch instructions from code space and data from data space.

2.2.1.2 I/O Space. An I/O space is defined where individual octets are specified by a 24 bit address (16 bit page address and 8 bit offset address). All, some, or none of this memory space may be present. The physical element associated with a read operation at a particular address may be different than the physical element associated with a write operation at the same address.

2.2.1.3 Scratch Pad Memory Space. A scratch pad memory space is defined where individual octets are specified by a 24 bit address (16 bit page address and 8 bit offset address). All, some, or none of this memory may be present. Individual octets of this space are read/write. The size of this space is device dependent.

2.2.1.4 Special Function Register Space. The special function register space is defined where individual octets are specified by a 24 bit address (16 bit page address and 8 bit offset address). All, some, or none of this register space may be present. Individual bits of the special function register space are read only, write only, or read/write. The definition of each octet of this space is device dependent.

2.2.2 Task Model. GBS requires several basic architectural components of the operating system. GBS exists within a task environment where messages are exchanged between tasks. GBS also requires a uniform upper limit to the size of messages within a device. Tasks on the master device may send command messages to tasks on slave devices addressable on the network. The tasks in these slave devices reply by returning response messages.

2.2.2.1 Task Identification. GBS recognizes up to 32 tasks within a device. Tasks are numbered 0–31 (GBS is task 0). GBS provides certain operations that may be performed on other tasks (e.g., creation and deletion), but does not define what a task is or does. The task identification address can be flat or structured (see Appendix A for examples).

2.2.2.2 Function ID. Each task has both a task number and an optional function ID. Since the binding of a task (other than GBS) to a task number may be nondeterministic, function IDs are provided as a mechanism for tasks to identify themselves to others.

2.2.2.3 Messages. GBS provides interdevice messaging. Intradevice messaging may also be provided, but it is outside the scope of this standard (see Appendix A).

The messaging service allows a task on the master device to send a message to a task (other than the GBS task) on a remote slave device. GBS does not interpret the data portion of these messages.

2.2.2.4 GBS Initialization Parameters. Certain parameters control the initialization of GBS. These parameters may be set at any time by a master, but only take effect upon device reset. Message length upper limit is an example of this parameter.

2.3 Functional Specifications. GBS provides five basic groups of services for user applications. The service groupings are

(1) Device control services
(2) Memory access services
(3) I/O access services

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

(4) Task access services
(5) User implemented services

2.3.1 Device Control Services. The device control services manipulate the device as a whole. Specifically, the following services are supplied:

(1) *Reset Device.* Provides a software reset of a slave device upon command from the master device. No network response is returned by the slave device.
(2) *Mark Device Offline.* Allows an application on the master to mark a slave device offline in the master's state table for a particular slave device.
(3) *Read Device Information.* Allows an application to request device description data. Certain parameters that impact master operations with the slave, such as maximum message size and firmware version number, are in this slave descriptive data structure.
(4) *Enable/Disable Memory Space Access.* A pair of services that allow or disallow GBS memory access services.
(5) *Get/Set Time.* A pair of services that allow the reading and setting of time in a slave device.

2.3.2 Memory Access Services. These services allow access to the slave device's memory spaces, scratch pad memory space, and special function register space.

(1) *Upload/Download Data.* These services provide access to the data space memory defined for the processor. Access to the data space memory is linear (address followed by a block of sequential data).
(2) *Upload/Download Code.* These services provide access to the code space memory defined for the processor. Access to the code space memory is linear.
(3) *Read/Write Scratch Pad Memory.* These services allow access to the processor's scratch pad memory space. Access to this memory space is on a random basis (one or more address/data pairs).
(4) *Read/Write Special Function Registers.* These services allow access to processor dependent registers. Access to this memory space is on a random basis.

2.3.3 I/O Access Services. These services allow access to the slave device's I/O address space. Both direct access and logical access services are provided.

(1) *Read/Write I/O Locations.* These services allow direct, random access to octets within the device's I/O space.
(2) *Logical I/O (AND, OR, XOR, SET).* These services allow random access to octets within the device's I/O space. These functions access the octet data, perform a logical operation on the data, and then rewrite the data as well as return it.

2.3.4 Task Access Services. These services allow access to the slave device's tasks.

(1) *Create/Delete Task.* The master device may modify the task structure within a slave device.
(2) *Suspend/Resume Task.* The master device may pause and continue tasks on a slave.
(3) *Get Task ID.* A task in the master device can read the task ID associated with a specific function ID.
(4) *Read Task Function IDs.* A task in the master device can read the list of task function IDs from a slave.
(5) *Task Message Exchange.* Tasks communicate by way of messages.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

2.3.5 User Implemented Services. These services allow the user to define extensions to GBS that are accessible in the same manner as the standard GBS.

(1) *Define service.* Define a new GBS bus service, its command code, and the associated procedure.
(2) *Call service.* Call a user-defined procedure to process a command.

2.4 User-Generic Bus Services Interface. This section specifies the services that GBS provides to tasks. The services are described in an abstract way and in no way imply any particular implementation or exposed interface.

Generic Bus Services are requested by an application within the master device. The application is identified with a task number. Twenty-five services, divided into five basic groups, are provided.

(1) Device control services
  (a) Reset Device — GBS_RESET
  (b) Mark Device Offline — GBS_OFFLINE
  (c) Read Device Information — GBS_IDENT
  (d) Enable/Disable Memory Space Access — GBS_PROTECT
  (e) Set Time — GBS_TIME-SET
  (f) Get Time — GBS_TIME-GET
(2) Memory access services
  (a) Upload Data Space — GBS_UPLOAD-DATA
  (b) Download Data Space — GBS_DNLOAD-DATA
  (c) Upload Code Space — GBS_UPLOAD-CODE
  (d) Download Code Space — GBS_DNLOAD-CODE
  (e) Read Scratch Pad — GBS_READ-PAD
  (f) Write Scratch Pad — GBS_WRITE-PAD
  (g) Read Special Function Registers — GBS_READ-SFR
  (h) Write Special Function Registers — GBS_WRITE-SFR
(3) I/O access services
  (a) Read I/O — GBS_READ-I/O
  (b) Write I/O — GBS_WRITE-I/O
  (c) Logical I/O — GBS_LOGICAL-I/O
(4) Task access services
  (a) Create Task — GBS_CREATE
  (b) Delete Task — GBS_DELETE
  (c) Suspend/Resume Task — GBS_TASK-S/R
  (d) Get Task ID — GBS_GET-ID
  (e) Read Function IDs — GBS_READ-FIDS
  (f) Message Passing service — GBS_MESSAGE
(5) User extension services
  (a) Define service — GBS_DEFINE
  (b) Call service — GBS_CALL

2.4.1 Common Definitions. All service primitives are defined using a common set of definitions.

ASAP. Application service access point.

bfr_size. The maximum GBS buffer size (maximum GBS_PDU length plus 2).

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

destination. The destination device address. If the destination address is zero, the destination device address is defaulted to the device of the requesting application.

GBS_PDU. The protocol data unit (PDU) that is built by GBS.

GBS_SDU. The service data unit (SDU) that is transferred across the interface between GBS and a task.

status. The service completion status. A basic set of status codes are used to report conditions common to multiple services. Several services also have additional status codes to report conditions unique to the particular service. The basic set of status codes are

(1) OK
(2) Bad destination task number/task does not exist
(3) Task overflow
(4) Register bank overflow
(5) Duplicate function ID
(6) No buffers
(7) Bad task priority
(8) Bad task descriptor
(9) Insufficient memory
(10) Bad procedure address
(11) Timeout
(12) Protocol error
(13) Bad destination device address
(14) GBS protected
(15) Unknown GBS command
(16) Bad command length
(17) Requested service not implemented
(18) Undefined error

2.4.2 Device Control Services. The device control services manipulate the device as a whole.

2.4.2.1 Reset Device. The Reset Device service provides a software reset of a slave device upon command from the master device.

The primitive associated with the Reset Device service is GBS_RESET.request. The relationship between these primitives is shown in Fig 2.4-1.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     T
     GBS_RESET.request|             |                     I
                  \   |             |                     M
                   \  |             |                     E
                    -->|             |                     |
                      |             |--> SOFTWARE RESET   |
                      |             |                     V
```

Fig 2.4-1
Reset Service

NOTE: The effect of a SOFTWARE RESET on a slave device is to reset the Data Link Layer communications and put into effect any previously set GBS initialization parameters. It may also reset some or all other functions in the slave device, but these actions are application dependent and beyond the scope of this standard.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

2.4.2.1.1 GBS_RESET.request. The primitive that requests GBS to reset a slave device is

GBS_RESET.request(destination)

2.4.2.2 Mark Device Offline. The Mark Device Offline service allows a task on the master to mark a slave device offline in the master's state table for a particular slave device.

The primitive associated with the Reset Device service is GBS_OFFLINE.request. The relationship between these primitives is shown in Fig 2.4-2.

```
Task in Master Device | Generic Bus      |
                      |   Services       |
                      |                  |
                      |                  T
   GBS_OFFLINE.request|                  I
                  \   |                  M
                   \  |                  E
                    -->|  |              |
                       |  |              |
                      /|  V              |
                     / |                 |
                    |  |                 |
                    V  |                 |
   GBS_OFFLINE.confirm|                  |
                      |                  V
```

Fig 2.4-2
Mark Device Offline

2.4.2.2.1 GBS_OFFLINE.request. The primitive that requests GBS to mark a slave device offline is

GBS_OFFLINE.request(destination)

2.4.2.2.2 GBS_OFFLINE.confirm. The primitive that acknowledges the completion of the service (mark a slave device offline) and returns the completion status and requested data to the task that initiated the service is

GBS_OFFLINE.confirm(status)

2.4.2.3 Read Device Information. The Read Device Information service allows a task to request device description data. Certain parameters that impact master operations with the slave, such as maximum message size and firmware version number, are in the slave descriptive data structure.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Read Device Information service are GBS_IDENT.request, and GBS_IDENT.confirm. The relationship between these primitives is shown in Fig 2.4-3.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
                      |             |                     |
     GBS_IDENT.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                    /|             |                     |
                   / |             |                     |
                  |  |             |                     |
                  V  |             |                     |
     GBS_IDENT.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-3
Read Device Information Service

2.4.2.3.1 GBS_IDENT.request. The primitive that requests GBS to get the device identification information from a slave device is

GBS_IDENT.request(destination, count)

count = number of octets of information to return

2.4.2.3.2 GBS_IDENT.confirm. The primitive that acknowledges the completion of the Read Device Information service and returns the completion status and requested data to the task that initiated the service is

GBS_IDENT.confirm(status, device_name, version, hwdr_cfg, bfr_size)

device_name = Six octet ASCII device name or type
version = Two octet ASCII software revision identifier
hwdr_cfg = Hardware configuration (vendor dependent)
bfr_size = GBS buffer size (maximum length of a GBS_PDU is bfr_size - 2)

2.4.2.4 GBS Protect. The GBS Protect service allows a task to remotely enable or disable memory access functions on a remote device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the GBS Protect service are GBS_PROTECT.request, and GBS_PROTECT.confirm. The relationship between these primitives is shown in Fig 2.4-4.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
                      |             |                     |
   GBS_PROTECT.request|             |                     |
                  \   |             |                     |
                   \  |             |                     T
                    -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                   V  |             |                     |
   GBS_PROTECT.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-4
GBS Protect Service

2.4.2.4.1 GBS_PROTECT.request. The primitive that requests GBS to set the PROTECT flag in a slave device is

GBS_PROTECT.request(destination, protection)

protection = no_protect
memory_protect
write_protect

When the PROTECT flag in a slave device is set to no_protect, GBS is permitted to read and write the slave memory. When the PROTECT flag is set to memory_protect, GBS may neither read nor write the slave memory. When the PROTECT flag in a slave device is set to write_protect, GBS is permitted to read the slave memory but is prohibited from writing the slave memory.

2.4.2.4.2 GBS_PROTECT.confirm. The primitive that acknowledges the completion of the GBS Protect service and returns the completion status to the task that requested the service is

GBS_PROTECT.confirm(status)

2.4.2.5 Get Time. The Get Time service allows tasks in a master device to read the time from a slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Get Time service are GBS_TIME-GET.request, and GBS_TIME-GET.confirm. The relationship between these primitives is shown in Fig 2.4-5.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
                      |             |                     |
  GBS_TIME-GET.request|             |                     |
                  \   |             |                     |
                   \  |             |                     T
                    -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                   V  |             |                     |
  GBS_TIME-GET.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-5
Get Time Service

2.4.2.5.1 GBS_TIME-GET.request. The primitive that requests GBS to read the time from a slave device is

GBS_TIME-GET.request(destination)

2.4.2.5.2 GBS_TIME-GET.confirm. The primitive that acknowledges the completion of the Get Time service and returns the completion status and requested time data to the task that initiated the service is

GBS_TIME-GET.confirm(status, time_zone, time_offset, day_of_week, year, month, day, hour, minute, second)

time_zone	= number of hours from (UCT)
time_offset	= minutes offset within time zone
day_of_week	= day of week
year	= year since 1980
month	= month of year
day	= day of month
hour	= hours in current day
minute	= minutes in current hour
second	= seconds in current minute

2.4.2.6 Set Time. The Set Time service allows tasks in a master device to set time on a slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Set Time service are GBS_TIME-SET.request, and GBS_TIME-SET.confirm. The relationship between these primitives is shown in Fig 2.4-6.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
  GBS_TIME-SET.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                    /|             |                     |
                   / |             |                     |
                  |  |             |                     |
                  V  |             |                     |
  GBS_TIME-SET.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-6
Set Time Service

2.4.2.6.1 GBS_TIME-SET.request. The primitive that requests GBS to set the time at a slave is

GBS_TIME-SET.request(destination, time_zone, time_offset, day_of_week, year, month, day, hour, minute, second)

time_zone	= number of hours from UCT
time_offset	= minutes offset within time zone
day_of_week	= day of week
year	= year since 1980
month	= month of year
day	= day of month
hour	= hours in current day
minute	= minutes in current hour
second	= seconds in current minute

2.4.2.6.2 GBS_TIME-SET.confirm. The primitive that acknowledges the completion of the Set Time service and returns the completion status to the task that initiated the service is

GBS_TIME-SET.confirm(status)

2.4.3 Memory Access Services. These services allow access to the slave device's memory spaces, scratch pad memory space, and special function register space.

2.4.3.1 Linear Access Services. Linear Access services provide access to the data and code spaces.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

2.4.3.1.1 Upload Data Space. The Upload Data Space service allows tasks in a master device to read a block of memory from a slave device's data space.

The primitives associated with the Upload Data service are GBS_UPLOAD-DATA.request, and GBS_UPLOAD-DATA.confirm. The relationship between these primitives is shown in Fig 2.4-7.

```
 Task in Master Device | Generic Bus | Slave Device        |
                       |   Services  |                     |
                       |             |                     |
                       |             |                     |
GBS_UPLOAD-DATA.request|             |                     |
                  \    |             |                     |
                   \   |             |                     T
                    -->|           | |                     I
                       |           | |                     M
                       |           | |                     E
                       |           V |                     |
                      /|             |                     |
                     / |             |                     |
                    |  |             |                     |
                    V  |             |                     |
GBS_UPLOAD-DATA.confirm|             |                     |
                       |             |                     V
```

Fig 2.4-7
Upload Data Service

The primitive that requests GBS to read a block of data from a slave device's data memory is

GBS_UPLOAD-DATA.request(destination, address, count)

address = starting address of data block
count = number of octets to read (minimum 1, maximum [bfr_size-9] if address is on the same page as that previously specified, [bfr_size-12] if not)

The primitive that acknowledges the completion of the Upload Data Space service and returns the completion status and requested data to the task that initiated the service is

GBS_UPLOAD-DATA.confirm(status, data)

data = requested data (if status is No Error)

2.4.3.1.2 Download Data Space. The Download Data Space service allows tasks in a master device to write a block of memory to a slave device's data space.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Download Data Space service are GBS_DNLOAD-DATA.request, and GBS_DNLOAD-DATA.confirm. The relationship between these primitives is shown in Fig 2.4-8.

```
 Task in Master Device | Generic Bus | Slave Device        |
                       |   Services  |                     |
                       |             |                     |
GBS_DNLOAD-DATA.request|             |                     |
                   \   |             |                     |
                    \  |             |                     T
                     -->|           | |                     I
                       |           | |                     M
                       |           | |                     E
                       |           V |                     |
                      /|             |                     |
                     / |             |                     |
                    |  |             |                     |
                    V  |             |                     |
GBS_DNLOAD-DATA.confirm|             |                     |
                       |             |                     V
```

Fig 2.4-8
Download Data Service

The primitive that requests GBS to write a block of data into a slave device's data memory is

GBS_DNLOAD-DATA.request(destination, address, data)

address = starting address of data block
data = a block of data octets (minimum 1, maximum [bfr_size-9] if address is on the same page as that previously specified, [bfr_size-12] if not)

The primitive that acknowledges the completion of the Download Data Space service and returns the completion status and requested data to the task that initiated the service is

GBS_DNLOAD-DATA.confirm(status)

2.4.3.1.3 Upload Code Space. The Upload Code Space service allows tasks in a master device to read a block of memory from a slave device's code space.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Upload Code Space service are GBS_UPLOAD-CODE.request, and GBS_UPLOAD-CODE.confirm. The relationship between these primitives is shown in Fig 2.4-9.

```
 Task in Master Device | Generic Bus | Slave Device        |
                       |   Services  |                     |
                       |             |                     |
GBS_UPLOAD-CODE.request|             |                     |
                   \   |             |                     |
                    \  |             |                     T
                     -->|           | |                     I
                       |           | |                     M
                       |           | |                     E
                       |           V |                     |
                      /|             |                     |
                     / |             |                     |
                    |  |             |                     |
                    V  |             |                     |
GBS_UPLOAD-CODE.confirm|             |                     |
                       |             |                     V
```

Fig 2.4-9
Upload Code Service

The primitive that requests GBS to read a block of data from a slave device's code space memory is

GBS_UPLOAD-CODE.request(destination, address, count)

address = starting address of data block
count = number of octets to read (minimum 1, maximum [bfr_size-9] if address is on the same page as that previously specified, [bfr_size-12] if not)

The primitive that acknowledges the completion of the Upload Code Space service and returns the completion status and requested data to the task that initiated the service is

GBS_UPLOAD-CODE.confirm(status, data)

data = requested data (if status is No Error)

2.4.3.1.4 Download Code Space. The Download Code Space service allows tasks in a master device to write a block of memory to a slave device's code space.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Download Code Space service are GBS_DNLOAD-CODE.request, and GBS_DNLOAD-CODE.confirm. The relationship between these primitives is shown in Fig 2.4-10.

```
 Task in Master Device | Generic Bus | Slave Device        |
                       |   Services  |                     |
                       |             |                     |
GBS_DNLOAD-CODE.request|             |                     |
                  \    |             |                     |
                   \   |             |                     T
                    -->|           | |                     I
                       |           | |                     M
                       |           | |                     E
                       |           V |                     |
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                   V  |             |                     |
GBS_DNLOAD-CODE.confirm|             |                     |
                       |             |                     V
```

Fig 2.4-10
Download Code Service

The primitive that requests GBS to write a block of data into a slave device's code space memory is

GBS_DNLOAD-CODE.request(destination, address, data)

address = starting address of data block
data = a block of data octets (minimum 1, maximum [bfr_size-9] if address is on the same page as that previously specified, [bfr_size-12] if not)

The primitive that acknowledges the completion of the Download Code Space service and returns the completion status and requested data to the task that initiated the service is

GBS_DNLOAD-CODE.confirm(status)

2.4.3.2 Random Access Services. Random Access services provide access to scratch pad registers and special function registers. Multiple octets from scattered memory addresses can be accessed in a single request.

2.4.3.2.1 Read Scratch Pad. The Read Scratch Pad service allows tasks in a master device to read data from a slave device's scratch pad memory space.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Read Scratch Pad service are GBS_READ-PAD.request, and GBS_READ-PAD.confirm. The relationship between these primitives is shown in Fig 2.4-11.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
  GBS_READ-PAD.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                    /|             |                     |
                   / |             |                     |
                  |  |             |                     |
                  V  |             |                     |
  GBS_READ-PAD.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-11
Read Scratch Pad Service

The primitive that requests GBS to read data from a slave device's scratch pad memory is GBS_READ-PAD.request(destination, address_list)

address_list = list of addresses (minimum 1, maximum [(bfr_size-7)/2] if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

The primitive that acknowledges the completion of the Read Scratch Pad service and returns the completion status and requested data to the task that initiated the service is

GBS_READ-PAD.confirm(status, data_list)

data_list = requested data (if status is No Error)

2.4.3.2.2 Write Scratch Pad. The Write Scratch Pad service allows tasks in a master device to write data to a slave device's scratch pad memory space.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Write Scratch Pad service are GBS_WRITE-PAD.request, and GBS_WRITE-PAD.confirm. The relationship between these primitives is shown in Fig 2.4-12.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
 GBS_WRITE-PAD.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |          V |                     |
                    /|             |                     |
                   / |             |                     |
                  |  |             |                     |
                 V   |             |                     |
 GBS_WRITE-PAD.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-12
Write Scratch Pad Service

The primitive that requests GBS to write data into a slave device's scratch pad memory is

GBS_WRITE-PAD.request(destination, address/data_list)

address/data_list = list of addresses and data values (minimum 1, maximum [(bfr_size-7)/2] if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

The primitive that acknowledges the completion of the Write Scratch Pad service to the task that initiated the service is

GBS_WRITE-PAD.confirm(status)

2.4.3.2.3 Read Special Function Registers. The Read Special Function Registers service allows tasks in a master device to read data from the special functions registers (e.g., in the processor) in the slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Read Special Function Registers service are GBS_READ-SFR.request, and GBS_READ-SFR.confirm. The relationship between these primitives is shown in Fig 2.4-13.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
  GBS_READ-SFR.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                    /|             |                     |
                   / |             |                     |
                  |  |             |                     |
                  V  |             |                     |
  GBS_READ-SFR.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-13
Read Special Function Register Service

The primitive that requests GBS to read data from a slave device's special function registers is

GBS_READ-SFR.request(destination, address_list)

address_list = list of addresses (minimum 1, maximum [(bfr_size-7)/2)] if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

The primitive that acknowledges the completion of the Read Special Function Register service and returns the completion status and requested data to the task that initiated the service is

GBS_READ-SFR.confirm(status, status/data_list)

status/data_list = list of status and data values (status uses same values as status parameter)

2.4.3.2.4 Write Special Function Registers. The Write Special Function Registers service allows tasks in a master device to write data to a slave device's special function registers.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Write Special Function Registers service are GBS_WRITE-SFR.request, and GBS_WRITE-SFR.confirm. The relationship between these primitives is shown in Fig 2.4-14.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
 GBS_WRITE-SFR.request|             |                     |
                  \   |             |                     |
                   \  |             |                     T
                    -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                   V  |             |                     |
 GBS_WRITE-SFR.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-14
Write Special Function Register Service

The primitive that requests GBS to write data into a slave device's special function registers is

GBS_WRITE-SFR.request(destination, address/data_list)

address/data_list = list of addresses and data values (minimum 1, maximum [(bfr_size-7)/2]) if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

The primitive that acknowledges the completion of the Write Special Function Registers service to the task that initiated the service is

GBS_WRITE-SFR.confirm(status, status_list)

status_list = list of status values that correspond to each register written.

2.4.4 I/O Access Services. These services allow access to the slave device's I/O. The services take as a parameter a list of addresses (or addresses and data). These lists shall be executed in the order provided.

2.4.4.1 Read I/O. The Read I/O service allows tasks in a master device to read I/O data from a slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Read I/O service are GBS_READ-I/O.request, and GBS_READ-I/O.confirm. The relationship between these primitives is shown in Fig 2.4-15.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
  GBS_READ-I/O.request|             |                     |
                 \    |             |                     |
                  \   |             |                     |
                   -->|           | |                     T
                      |           | |                     I
                      |           | |                     M
                      |           V |                     E
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                  V   |             |                     |
  GBS_READ-I/O.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-15
Read I/O Service

2.4.4.1.1 GBS_READ-I/O.request. The primitive that requests GBS to read data from a slave device's I/O is

GBS_READ-I/O.request(destination, address_list)

address_list = list of addresses (minimum 1, maximum [(bfr_size-7)/2)] if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

2.4.4.1.2 GBS_READ-I/O.confirm. The primitive that acknowledges the completion of the Read I/O service and returns the completion status and requested data to the task that initiated the service is

GBS_READ-I/O.confirm(status, data_list)

data_list = requested data (if status is No Error)

2.4.4.2 Write I/O. The Write I/O service allows tasks in a master device to write data to a slave device's I/O.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Write I/O service are GBS_WRITE-I/O.request, and GBS_WRITE-I/O.confirm. The relationship between these primitives is shown in Fig 2.4-16.

```
Task in Master Device | Generic Bus | Slave Device        |
                      |   Services  |                     |
                      |             |                     |
 GBS_WRITE-I/O.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|           | |                     I
                      |           | |                     M
                      |           | |                     E
                      |           V |                     |
                    /|             |                     |
                   / |             |                     |
                  |  |             |                     |
                  V  |             |                     |
 GBS_WRITE-I/O.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-16
Write I/O Service

2.4.4.2.1 GBS_WRITE-I/O.request. The primitive that requests GBS to write data into a slave device's I/O is

GBS_WRITE-I/O.request(destination, address/data_list)

address/data_list = list of addresses and data values (minimum 1, maximum [(bfr_size-7)/2)] if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

2.4.4.2.2 GBS_WRITE-I/O.confirm. The primitive that acknowledges the completion of the Write I/O service to the task that initiated the service is

GBS_WRITE-I/O.confirm(status)

2.4.4.3 Logical I/O. The Logical I/O service allows tasks in a master device to perform a logical operation on a slave's I/O and read back the results.

Documents
Document Sharing Hub

Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.

Full document details: IEEE 1118.1-1990

The primitives associated with the Logical I/O Service are GBS_LOGICAL-I/O.request, and GBS_LOGICAL-I/O.confirm. The relationship between these primitives is shown in Fig 2.4-17.

```
 Task in Master Device | Generic Bus | Slave Device        |
                       |   Services  |                     |
                       |             |                     |
GBS_LOGICAL-I/O.request|             |                     |
                   \   |             |                     |
                    \  |             |                     T
                     -->|           | |                     I
                       |           | |                     M
                       |           | |                     E
                       |           V |                     |
                      /|             |                     |
                     / |             |                     |
                    |  |             |                     |
                    V  |             |                     |
GBS_LOGICAL-I/O.confirm|             |                     |
                       |             |                     V
```

Fig 2.4-17
Logical I/O Service

2.4.4.3.1 GBS_LOGICAL-I/O.request. The primitive that requests GBS to perform a logical operation on a slave device's I/O is

GBS_LOGICAL-I/O.request(destination, mode, address/data_list)

mode = OR, AND, XOR, SET
address/data_list = list of addresses and data values (minimum 1, maximum [(bfr_size-7)/2)] if address is on the same page as that previously specified, [(bfr_size-10)/2] if not)

NOTE: All addresses must be on the same memory page (see Section 2.2.1).

2.4.4.3.2 GBS_LOGICAL-I/O.confirm. The primitive that acknowledges the completion of the Logical I/O service and returns the completion status and requested data to the task that initiated the service is

GBS_LOGICAL-I/O.confirm(status, data_list)

data_list = requested data (if status is No Error)

2.4.5 Task Access Services. These services allow access to the slave device's tasks. GBS recognizes up to 32 tasks within a device numbered from 0 to 31. Task 0, if present, is always GBS (multiple copies of GBS, each with their own task number, are permitted).

2.4.5.1 Create Task. The Create Task service allows the master device to create a task on a slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Create Task service are GBS_CREATE.request, and GBS_CREATE.confirm. The relationship between these primitives is shown in Fig 2.4-18.

```
Task in Master Device | Generic Bus | Operating System    |
                      |   Services  | in Slave Device     |
                      |             |                     |
    GBS_CREATE.request|             |                     |
                 \    |             |                     |
                  \   |             |                     T
                   -->|             |                     I
                      |             |--> CREATE_TASK      M
                      |             |         |           E
                      |             |<--------+           |
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                   V  |             |                     |
    GBS_CREATE.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-18
Create Task Service

2.4.5.1.1 GBS_CREATE.request. The primitive that requests GBS to create a task on a slave device is

GBS_CREATE.request(destination, descriptor_address)

descriptor_address = address of task descriptor

2.4.5.1.2 GBS_CREATE.confirm. The primitive that acknowledges the completion of the Create Task service to the task that initiated the service is

GBS_CREATE.confirm(status, task_number)

task_number = number of task created

2.4.5.2 Delete Task. The Delete Task service allows the master device to delete a task on a slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990

The primitives associated with the Delete Task service are GBS_DELETE.request, and GBS_DELETE.confirm. The relationship between these primitives is shown in Fig 2.4-19.

```
Task in Master Device | Generic Bus | Operating System    |
                      |   Services  | in Slave Device     |
                      |             |                     |
    GBS_DELETE.request|             |                     |
                  \   |             |                     |
                   \  |             |                     T
                    -->|             |                     I
                      |             |--> DELETE_TASK      M
                      |             |         |           E
                      |             |<--------+           |
                     /|             |                     |
                    / |             |                     |
                   |  |             |                     |
                   V  |             |                     |
    GBS_DELETE.confirm|             |                     |
                      |             |                     V
```

Fig 2.4-19
Delete Task Service

2.4.5.2.1 GBS_DELETE.request. The primitive that requests GBS to delete a task on a slave device is

GBS_DELETE.request(destination, task_number)

task_number = number of task to be deleted

2.4.5.2.2 GBS_DELETE.confirm. The primitive that acknowledges the completion of the Delete Task service to the task that initiated the service is

GBS_DELETE.confirm(status)

2.4.5.3 Suspend/Resume Task. The Suspend/Resume Task service allows the master device to pause or continue a task on a slave device.

Documents
Document Sharing Hub
Authorized licensed use limited to: Tsinghua University. Downloaded on February 24,2026 at 09:50:17 UTC from IEEE Xplore. Restrictions apply.
Full document details: IEEE 1118.1-1990