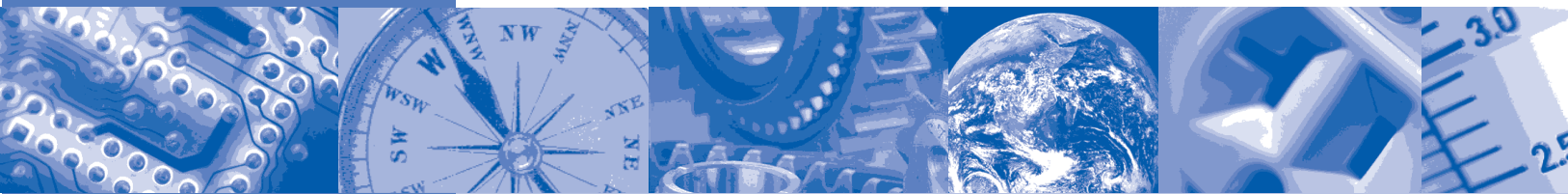


ISA-TR50.02, Part 9-2000



Fieldbus Standard for Use in Industrial Control Systems: User Layer Technical Report



**ISA—The Instrumentation,
Systems, and
Automation Society**

Copyright The Instrumentation, Systems, and Automation Society

Approved 01 January 2000



Documents

Document Sharing Hub

ISA would like to thank the former Chairman of SP50 and current SP50.4 member Cullen Langford for his outstanding contribution in editing ISA–TR50.02, Part 9, User Layer Technical Report, and for his integral role in converting the text into a suitable electronic format.

ISA-TR50.02, Part 9-2000, Fieldbus Standard for Use in
Industrial Control Systems: User Layer Technical Report

ISBN: 1-55617-718-6

Copyright © 2000 by the Instrument Society of America. All rights reserved. Not for resale. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means (electronic mechanical, photocopying, recording, or otherwise), without the prior written permission of the Publisher.

ISA 67 Alexander Drive P. O. Box 12277
Research Triangle Park, North Carolina 27709

Preface

This preface, as well as all footnotes and annexes, is included for information purposes and is not part of ISA-TR50.02, Part 9-2000.

This document has been prepared as part of the service of ISA, the international society for measurement and control, toward a goal of uniformity in the field of instrumentation. To be of real value, this document should not be static but should be subject to periodic review. Toward this end, the Society welcomes all comments and criticisms and asks that they be addressed to the Secretary, Standards and Practices Board; ISA; 67 Alexander Drive; P. O. Box 12277; Research Triangle Park, NC 27709; Telephone (919) 549-8411; Fax (919) 549-8288; E-mail: standards@isa.org.

The ISA Standards and Practices Department is aware of the growing need for attention to the metric system of units in general, and the International System of Units (SI) in particular, in the preparation of instrumentation standards. The Department is further aware of the benefits to USA users of ISA standards of incorporating suitable references to the SI (and the metric system) in their business and professional dealings with other countries. Toward this end, this Department will endeavor to introduce SI-acceptable metric units in all new and revised standards, recommended practices, and technical reports to the greatest extent possible. *Standard for Use of the International System of Units (SI): The Modern Metric System*, published by the American Society for Testing & Materials as IEEE/ASTM SI 10-97, and future revisions, will be the reference guide for definitions, symbols, abbreviations, and conversion factors.

It is the policy of ISA to encourage and welcome the participation of all concerned individuals and interests in the development of ISA standards, recommended practices, and technical reports. Participation in the ISA standards-making process by an individual in no way constitutes endorsement by the employer of that individual, of ISA, or of any of the standards, recommended practices, and technical reports that ISA develops.

CAUTION – ISA ADHERES TO THE POLICY OF THE AMERICAN NATIONAL STANDARDS INSTITUTE WITH REGARD TO PATENTS. IF ISA IS INFORMED OF AN EXISTING PATENT THAT IS REQUIRED FOR USE OF THE STANDARD, IT WILL REQUIRE THE OWNER OF THE PATENT TO EITHER GRANT A ROYALTY-FREE LICENSE FOR USE OF THE PATENT BY USERS COMPLYING WITH THE STANDARD OR A LICENSE ON REASONABLE TERMS AND CONDITIONS THAT ARE FREE FROM UNFAIR DISCRIMINATION.

EVEN IF ISA IS UNAWARE OF ANY PATENT COVERING THIS STANDARD, THE USER IS CAUTIONED THAT IMPLEMENTATION OF THE STANDARD MAY REQUIRE USE OF TECHNIQUES, PROCESSES, OR MATERIALS COVERED BY PATENT RIGHTS. ISA TAKES NO POSITION ON THE EXISTENCE OR VALIDITY OF ANY PATENT RIGHTS THAT MAY BE INVOLVED IN IMPLEMENTING THE STANDARD. ISA IS NOT RESPONSIBLE FOR IDENTIFYING ALL PATENTS THAT MAY REQUIRE A LICENSE BEFORE IMPLEMENTATION OF THE STANDARD OR FOR INVESTIGATING THE VALIDITY OR SCOPE OF ANY PATENTS BROUGHT TO ITS ATTENTION. THE USER SHOULD CAREFULLY INVESTIGATE RELEVANT PATENTS BEFORE USING THE STANDARD FOR THE USER'S INTENDED APPLICATION.

HOWEVER, ISA ASKS THAT ANYONE REVIEWING THIS STANDARD WHO IS AWARE OF ANY PATENTS THAT MAY IMPACT IMPLEMENTATION OF THE STANDARD NOTIFY THE ISA STANDARDS AND PRACTICES DEPARTMENT OF THE PATENT AND ITS OWNER.

ADDITIONALLY, THE USE OF THIS STANDARD MAY INVOLVE HAZARDOUS MATERIALS, OPERATIONS OR EQUIPMENT. THE STANDARD CANNOT ANTICIPATE ALL POSSIBLE APPLICATIONS OR ADDRESS ALL POSSIBLE SAFETY ISSUES ASSOCIATED WITH USE IN HAZARDOUS CONDITIONS. THE USER OF THIS STANDARD MUST EXERCISE SOUND PROFESSIONAL JUDGMENT CONCERNING ITS USE AND APPLICABILITY UNDER THE USER'S PARTICULAR CIRCUMSTANCES. THE USER MUST ALSO CONSIDER THE APPLICABILITY OF ANY GOVERNMENTAL REGULATORY LIMITATIONS AND ESTABLISHED SAFETY AND HEALTH PRACTICES BEFORE IMPLEMENTING THIS STANDARD.

The following people served as members of ISA Subcommittee SP50.4:

H. Dammeyer, Managing Director	The Ohio State University
G. Wood, Managing Director	Graeme Wood Consulting
R. Appleby	Du Pont
*T. Blevins	Fisher•Rosemount Systems Inc.
*E. Bristol	Foxboro Company
J. Carmody	Honeywell, Inc.
R. Caro	Automation Research Corporation
F. Charles	Consultant
T. Craeco	Consultant
E. dela Hostria	Rockwell Automation
U. Doebrich	Siemens AG
P. Early	Texas Instrument
M. Giraudi	Consultant
*J. Gray	Foxboro Company
C. Gross	Dow Chemical Company
W. Hawkins	HLQ Limited
F. Jaye	Consultant
M. Kazahaya	MMK International Inc.
R. Ketcham	Union Carbide Corporation
C. Langford	Cullen G. Langford, Inc.
R. Lasher	Bailey Controls Company
M. Ledier	Schneider Electric
K. Lindner	Endress+Hauser
T. Madden	Exxon Company
E. Messano	Deceased
*R. Prescott	Foxboro Company
F. Russo	ENEL-Ricerca/PEA
*J. Scarlett	Honeywell Inc.
E. Skabowski	Tendent Consulting
R. Szanyi	Mobil Technology Company
*C. Van Haren	James River Corporation
M. Wozniak	Bailey Controls Company
W. Wright	Moore Products Company

The following people served as members of ISA Committee SP50:

R. Caro, Chairman	Automation Research Corporation
H. Dammeyer, Managing Director	The Ohio State University
G. Wood, Managing Director	Graeme Wood Consulting
C. Akiyama	Yokogawa Electric Corporation
J. Aliperti	Chromatech Consultoria Limited
W. Baker	KML, Inc.
L. Beckman	HIMA Americas, Inc.

G. Beusch	Absolute Control Inc.
M. Borges	Petrobras Cenpes Supen Diprom
*R. Bowden	Fisher•Rosemount Ltd.
S. Boyer	Iliad Engineering Inc.
*R. Breithaupt	Siemens
*P. Brett	Honeywell, Inc.
P. Burton	Holwell Consulting
E Butteriss	Global Tradelinks Corp.
*R. Canfield	Square D Company
A. Capel	Comgate Engineering, Ltd.
B. Casner	Miller Electric Manufacturing Company
*S. Cavalieri	Universita Degli Studi di Catania
R. Crowder	Ship Star Associates, Inc.
M. Dajani	Phillip Morris
E. dela Hostria	Rockwell Automation
*U. Doebrich	Siemens AG
J. Dohoney	Jet Propulsion Laboratory
B. Dumortier	Schneider Electric SA
*J. Elias	Honeywell, Inc.
T. Feindel	R. Stahl, Inc.
M. Giraudi	Consultant
*R. Glaser	Dow Chemical Company
*J. Gray	Foxboro Company
*P. Griem, Jr.	Honeywell IAC
B. Gross	General Motors Corporation
*C. Gross	Dow Chemical Company
A. Gupta	Ficon Technology
H. Hasler	Adelcontrol
*T. Holland	Square D Company
*J. Howard	Foxboro Company
S Hwang	Automation Systems
*T. Ito	Fuji Electric
*V. Jacobson	Ronan Engineering Company
J. Jamison	Bantrel Inc.
*K. Kakizakai	Fuji Electric Company, Ltd.
M. Kazahaya	MMK International Inc.
*R. Ketcham	Union Carbide Corporation
*K. Krivoshein	Fisher•Rosemount Systems Inc.
M. Kross	RoviSys
*Y. Kumeda	Yamatake Industrial Systems
T. Kuusisto	Consultant
*T. Laine	Alstom Technology
H. Landet	FINT
C. Langford	Cullen G. Langford Inc.
*R. Lasher	Bailey Controls Company
A. Lesh	AMP Inc.
P. Leviti	Marconi Italiana Automazione
K. Lindner	Endress + Hauser GmbH
T. Madden	Exxon Company USA
*A. Mathur	Bailey Controls Company
*G. Mazzola	ENEL
A. McCauley, Jr.	Chagrin Valley Controls, Inc.
F. McKenna	FMCK Associates, Ltd.
R. Meeker	Process Control Solutions, Inc.
M. Melton	KOSA
A. Meol	Macromicro
D. Merriman	MerTech, Inc.
E. Miller, Jr.	SouthWest Research Institute
N. Miller	Intuitive Technology Corp.
*O. Mirabella	Universita Degli Studi di Catania
*D. Modell	ABB Bailey Controls
A. Mukherji	Systems Automation International Inc.
*G. Nachev	Isomatic Laboratories

L. Natiello
 *L. Neitzel
 H. Newman
 A. Nokkonen
 *P. Noury
 H. Ostling
 *E. Pageler
 P. Patel
 *T. Phinney
 J. Polon
 A. Poschmann
 D. Quay
 D. Rundle
 *F. Russo
 K. Schwarz
 *E. Sederlund
 D. Serang
 Y. Shimanuki
 E. Skabowski
 J. Slavinsky
 D. Sniezek
 J. Sprague
 R. Squires
 H. Storey
 R. Szanyi
 P. Thiessmeier
 *C. Thurston
 *N. Tobol
 *A. Tresset
 C. Vaidya
 C. Van Haren
 *J. Weiss
 C. Williams
 G. Winchester
 Assoc.
 *D. Wroblewski
 *M. Zielinski

Merck and Company Inc.
 Fisher•Rosemount Systems Inc.
 Cornell University
 Valmet Automation Inc.
 Cegelec
 Yokogawa Industrial Auto Inc.
 Fisher•Rosemount Systems Inc.
 V Automation
 Honeywell Inc.
 Neles-Jamesbury
 Ifak
 Rosemont Inc
 AAMP Inc
 ENEL-Ricerca/PEA
 Schwarz Communication Consult
 Dow Chemical Company
 MX Communications
 Toshiba Corporation
 Tendent Consulting
 Czech Standards Institute
 Lockheed Martin Federal Services
 Saudi Aramco Oil Company
 Measurement Technology Ltd.
 Equilon Enterprises LLC
 Mobil Tech Company
 Klockner-Moeller
 Union Carbide
 Ronan Engineering
 Cegelec
 Sun Microsystems
 James River Corporation
 Electric Power Research Institute
 Eastman Kodak Company
 National Electrical Manufacturers

 Bailey Controls Company
 Fisher•Rosemount Systems Inc.

This technical report was approved for publication by the ISA Standards and Practices Board on 1 January 2000.

M. Zielinski
 D. Bishop
 P. Brett
 M. Cohen
 M. Coppler
 B. Dumortier
 W. Holland
 A. Iverson
 R. Jones
 V. Maggioli
 T. McAviney
 LLC
 A. McCauley, Jr.
 G. McFarland
 R. Reimer
 J. Rennie
 H. Sasajima
 R. Webb
 W. Weidman
 J. Weiss

Fisher•Rosemount Systems, Inc.
 Chevron Petroleum Technology Co.
 Honeywell, Inc.
 Senior Flexonics, Inc.
 Ametek, Inc.
 Schneider Electric SA
 Southern Company
 Ivy Optiks
 Dow Chemical Co.
 Feltronics Corp.
 Instrumentation & Control Engineering

 Chagrin Valley Controls, Inc.
 Honeywell, Inc.
 Rockwell Automation
 Factory Mutual Research Corp.
 Advanced Architecture and Technologies
 Altran Corp.
 Parsons Energy & Chemicals Group
 EPRI

J. Whetstone
Technology
M. Widmeyer
R. Wiegler
C. Williams
G. Wood

National Institute of Standards &
EG&G
CANUS Corp.
Eastman Kodak Co.
Graeme Wood Consulting

Table of Contents

PREFACE.....	2
INTRODUCTION	12
CONFORMANCE CLASSES.....	13
REFERENCES.....	16
SP50 USER LAYER GUIDE SHEET.....	17
USER LAYER STRUCTURE	21
DATA OWNER STRUCTURE HARDWARE.....	22
DATA OWNER STRUCTURE LOGICAL NODES	45
BLOCK MODES	72
STATUS BYTE.....	86
<i>Status Byte - Attach 1 Bit 3-5 State Diagram</i>	<i>98</i>
<i>Status Byte - Attach. 1 State Diagram - Bits 5, 4, and 3</i>	<i>98</i>
<i>Status Byte - Attach 2 - Remote Handshake</i>	<i>104</i>
AGENTS	107
<i>Agent- Attach. 1 Discrete Agent Logic.....</i>	<i>126</i>
FUNCTION BLOCK STRUCTURE	132
<i>Attach. 1 Status Logic (vacant).....</i>	<i>167</i>
<i>Attach. 2 Fast Initialization Procedure</i>	<i>167</i>
APPLICATION SUPPORT SERVICES INTERFACE.....	190
<i>Attach. 1 Support Services.....</i>	<i>210</i>
<i>Attach. 2 Extended Parameters.....</i>	<i>224</i>
DATA BASE WRITE SERVICE.....	230
<i>Attach. 1 Complex Write.....</i>	<i>244</i>
<i>Attach. 2 Logic</i>	<i>259</i>
ALERT FUNCTION	280
<i>Attach. 1 Logical Node Alert Extensions.....</i>	<i>309</i>
<i>Attach. 2: Function Block Extensions For Specific Block Types.....</i>	<i>310</i>
FIELD DEVICE START AND RESTART	314
HUMAN INTERFACE CONSIDERATIONS -HIF.....	335
<i>Attach. 1 Auto-Formatting</i>	<i>353</i>
<i>Attach. 2 - Material Description</i>	<i>407</i>
ARRAY OF BASIC PARAMETERS.....	441
FUNCTION BLOCKS:	468
INTRODUCTION TO FUNCTION BLOCKS	468
OPEN BLOCKS	470
GENERIC BLOCK.....	472
<i>Interoperable Generic Block.....</i>	<i>472</i>
<i>Interworking Generic Block.....</i>	<i>489</i>
ALTERNATE AND STANDARD BLOCKS:.....	492
<i>Standard Function Block Functions</i>	<i>492</i>
<i>Null Algorithm.....</i>	<i>512</i>
ANALOG	513
<i>Analog Input.....</i>	<i>513</i>
<i>Analog Output</i>	<i>551</i>
<i>Counter</i>	<i>597</i>

<i>Dynamic Compensation</i>	605
<i>Math</i>	609
<i>PID Control</i>	617
<i>Property Conversion</i>	624
<i>Signal Capture</i>	633
<i>Signal Characterizer</i>	637
<i>Signal Selector</i>	643
<i>Splitter</i>	649
<i>Timer</i>	654
<i>Totalizer</i>	661
DISCRETE.....	671
<i>Device Control</i>	671
<i>Discrete Input Function Blocks</i>	677
<i>Discrete Output</i>	681
<i>Discrete Register</i>	685
EXPLANATION OF TIMER FUNCTIONS:.....	688
<i>Human Interface</i>	697
<i>Logic</i>	704
<i>Program Control</i>	728
<i>Register Interface Block</i>	739
SERIAL.....	750
<i>Serial Communication</i>	750
<i>Serial In Conversion</i>	752
APPENDIX	758
<i>Explanation of Field Bus Function Block</i>	759
<i>Explanation of Fieldbus Topologies</i>	764
<i>Explanation of Generic & Open FB</i>	771
<i>Timed Standard Node Design</i>	780
<i>Fundamental Control Structure</i>	784
<i>LO Mode Explanation</i>	804
<i>Bad Values</i>	815
<i>Block Structure for Cascading</i>	820
<i>Cascade Initialization & Recover</i>	825
<i>Explanation Of ASK, BASK, Acquirer</i>	832
<i>(Original) App Layer Services</i>	837
<i>Explanation Of Alarm Function</i>	850
<i>Explanation of Discrete I/O</i>	855
<i>Thermocouple Multiplexer & A/I</i>	858
<i>AO MSP Data</i>	860
<i>Explanation of Selector Bias</i>	870
<i>Explanation of Timer Block</i>	874
<i>Explanation of Device Control Block</i>	881
GLOSSARY	902
COMMENTS, PHOENIX	918
INDEX	980

Table of Figures

Figure 1: Preface, Figure 1, Protocol Models	5
Figure 2: Node	6
Figure 3: Diagram: Function Blocks / P&ID.....	9
Figure 4 Conformance Classification.....	14
Figure 5 Conformance Classification (cont.).....	15
Figure 6: Modes by Bit	17
Figure 7: Hardware Point types.....	17
Figure 8: Status Info. by Bit.....	17
Figure 9: Block Algorithms	18
Figure 10: User Layer Terms.....	19
Figure 11: Cascade Structure.....	20
Figure 12: A Field Device on Fieldbus.....	43
Figure 13 A Complex Fieldbus Device	44
Figure 14: Data Owner Structure, Figure 1, Timed Standard Cycle Structure	60
Figure 15: Attribute / Access Limits	85
Figure 16: Figure 1 Status State Diagram	103
Figure 17: State Transition Diagram	106
Figure 18: Agents Figure 1: Discrete Active Agent Logic	131
Figure 19: Function Block	132
Figure 20: Discrete Active Agent Logic (path marked in red)	160
Figure 21: Function Block Structure: Figure 1, Generalized Analog Block	164
Figure 22: Function Block Structure: Figure 2, Data Structures	165
Figure 23: Function Block Structure: Figure 3, Analog Status Parameter Detail	166
Figure 24: Generic Conditional Writes	257
Figure 25: Generic Conditional Writes (cont.).....	258
Figure 26: DBWS Figure 1, Buffer Service Logic	268
Figure 27: DBWS Figure 2, Static & Auto-format Data Base Revision Number.....	269
Figure 28: DBWS Figure 3, Log./Phy. Node Values	270
Figure 29: DBWS Figure 4, Mode	271
Figure 30: DBWS Figure 5, Data Base Values.....	272
Figure 31: DBWS Figure 6, Data Base Values (cont.).....	273
Figure 32: Application Layer - Writing to Buffer.....	278
Figure 33: Application Layer, Figure 2, Clipboard Revision Number Logic	279
Figure 34: Alert Figure 1, Multiple Hardware Failure Suppression	308
Figure 35: Field Device Start and Restart, Figure 1, Physical Node Power-Up	332
Figure 36: Field Device Start and Restart, Figure 2, Logical Node Power-Up	333
Figure 37: Field Device Start and Restart, Figure 3, Communications Interface Power-Up	334
Figure 38: HIF Figure 1, Auto-Formatting Parameter Layout	367
Figure 39: HIF Figure 2, Auto-Formatting Parameter Header	368
Figure 40: HIF Figure 3, Auto-Formatting Detail Words	369
Figure 41: Analog Input, Figure 1 Analog Input Algorithm.....	531
Figure 42: Analog Output, Figure 1, Analog Output System Diagram	578
Figure 43: Analog Output, Figure 2, Illustration of Characterization Types	579
Figure 44: Counter Figure 1, Counter Block.....	603
Figure 45: Counter, Figure 2, Counter Data Flow.....	604
Figure 46: Properties Conversion, Figure 1, Property Block.....	632
Figure 47: Timer, Timer Algorithm.....	659
Figure 48: Timer, Timer 0 Logic.....	660
Figure 49: Totalizer Block.....	669
Figure 50: Totalizer Data Flow.....	670
Figure 51: Device Block	676
Figure 52: Discrete Register, Figure 1, Discrete Register Block	694

Figure 53: Discrete Register, Figure 2, Discrete Point.....	695
Figure 54: Discrete Register, Figure 3 Discrete Value Flow	696
Figure 55: Human Interface, Figure 1, Local HI/F Block	703
Figure 56: Logic, Figure 1, Logic Algorithm	722
Figure 57: Logic, Figure 2, SPx Logic, Version 0 - Independent.....	723
Figure 58: Logic, Figure 3, SPx Logic, Version 1 - On/Off/Rev.....	724
Figure 59: Logic, Figure 4, Logic Block - Instruction Definition Word.....	725
Figure 60: Logic, Figure 5, Logic Block - Detail Words.....	726
Figure 61: Logic, Figure 6, Logic Icons - Basic Shape	727
Figure 62: Program Algorithm	738
Figure 63: Register Interface, Figure 1, Register Interface Block	748
Figure 64: Register Interface, Figure 2, Discrete Interface	749
Figure 65: Serial Communication, Figure 1, Serial Communication Block	757
Figure 66: Function Block, Figure 1, ...	762
Figure 67: Function Blocks, Matrix of Defined Block types	763
Figure 68: Fieldbus Topologies Figure 1 Simple Field Loop	767
Figure 69: Fieldbus Topologies, Figure 2, Central Control.....	768
Figure 70: Fieldbus Topologies Figure 3 Field Multiplexer	769
Figure 71: Fieldbus Topologies Figure 4 Unit Controllers.....	770
Figure 72: Generic & Open Blocks, Figure 1, Cyclic Scenarios	783
Figure 73: Fundamental Control Structure, Figure 1, Mode Concept	800
Figure 74: Fundamental Control Structure, Figure 2, Mode - Control	801
Figure 75: Fundamental Control Structure, Figure 3, PID Block Fundamentals	802
Figure 76: Fundamental Control Structure, Figure 4, Example Block Cascade	803
Figure 77 LO Mode, Figure 1, Case Definitions	814
Figure 78 Figure 1 Cascade Structure	821
Figure 79: Block Structure for Cascading, Cascade Structure	824
Figure 80: Cascade Initialization & Recovery, Figure 1	831
Figure 81: Selector Bias, Process Model	873
Figure 82: Device Control Block, Figure 1, Device Block	901
Figure 83: Mascot.....	983

Preface

FIELD BUS PROCESS CONTROL USER LAYER TECHNICAL REPORT

The Field Bus being developed simultaneously by ISA SP50 and IEC SC65C/WG6 is intended to be applied to time-critical applications in process control, distributed data acquisition and some areas of factory automation. While generally we planned to follow the "spirit and intent" of international data communications standards, our first priority is to these time-critical or real-time applications. Explicitly, we provided a standard to satisfy user needs.

This Technical Report defines the details of the field bus process control user layer for continuous and batch process control applications, as well as typical sequential and interlock control applications in the process industries. It is usable in other industries wherever similar applications exist. It establishes a data communications structure for implementing all types of field bus devices. This structure is required to achieve interoperability and interchangeability of field devices.

This paper includes the following sections:

- BACKGROUND
- FIELD BUS REQUIREMENTS
- USER LAYER DEFINITION
- DIFFERENCES BETWEEN DCS AND FIELD BUS
- FUNCTION BLOCKS
- ABOUT PROGRAMMABLE LOGIC CONTROLLERS
- FIELD BUS DEVICES
- USER LAYER CONCEPTS
- OBSERVATIONS AND CONCLUSIONS

BACKGROUND:

The top layer of the ISO (International Standards Organization) OSI (Open Systems Interconnection) Reference Model (ISO 7498-1, 1987) is numbered seven (7), and is called the "Application Layer." Above this is generally called "the application." Therefore, the Application Layer provides communications services specific to the type of application using the open system. The ISO/OSI seven layer architecture, is intentionally vague about the definition of the contents of the Application Layer, while the functions of all lower six layers are well defined in the general data communications environment. The vagueness allows application needs to define the functions of the seventh layer.

In 1986, the MAP (Manufacturing Application Protocol) Steering Committee of the MAP Users' Group requested that the major applications of interest to manufacturing should prepare "companion standards" to support the core (general) work being done at ISO. This work supports all applications, and is called MMS, Manufacturing Messaging Service, ISO 9506. The major functions of MMS are as follows:

- o Uploading and downloading
- o Task (function) management
- o Variable and data access (read, write)
- o Semaphore management
- o Event management
- o Journal management
- o File management
- o Status and identification query
- o Clock synchronization

The four companion standards are being (have been) prepared by the following organizations:

Process Control	ISA SP72
Robotics	Robotics Institute of America (RIA)
Programmable Logic Control	NEMA (National Electrical Manufacturer's Association)
Numerical Machine Control	EIA (Electronics Institute of America)

In 1985, the ISA approved the charter for the SP50 committee to develop a digital Field Bus. One of the primary directives for the Field Bus was to enable digital communications for instrumentation and control to the same level of interchange that has been achieved by analog (4-20 ma) transmission. After the first few meetings, it became obvious that digital and logical device communications were also necessary.

After review of MMS, ISA S72.02 and the other companion standards, ISA SP50, the Field Bus committee, decided that a fresh approach was required. Specifically, it was decided that the end users' needs must be considered in the definition of the application layer services. While the MMS services were probably good for low level (programmer) use of the Field Bus, more work was required to define a high level interface for the types of functions and services necessary to serve the distributed process control environment of the late 1990's.

FIELD BUS REQUIREMENTS:

The Field Bus committee began by developing a priority-ordered list of requirements that a Field Bus must satisfy. These requirements led to the definition of a two-level physical layer to satisfy both the needs of the following:

- o H1 slow speed traditional process control using existing wiring, meeting intrinsic safety requirements, and delivering field device power on the same wires, and
- o H2 high speed process and logical control using new wiring and, if possible, deliver power on the same wires and meet intrinsic safety requirements.

Many of the user requirements for Field Bus do not appear in the body of the standard, because they do not directly affect either the form or function. However, these user requirements do affect our understanding of the form and function of both the physical and user layers, and "why" the standard functions and features exist. Some of this information is contained in the first chapter of the appendix.

Many of the requirements were expressed in terms of process control functionality, rather than reduced to "communications language." Very soon, the Applications Layer subcommittee split into two organizations:

- o Applications Layer Subcommittee -- to define the OSI Application Layer (7) functions required to support user functions, and
- o User Layer Subcommittee -- to define the user needs.

In order to define the actual needs of users, it was determined that a model of the user layer software was required complete with its functionality and algorithmic behavior. From this model, the services that would be needed to support such software that corresponds to the "application" noted in the OSI model could be defined. This procedure differs from the procedures used in other communications standards which assume the "application" is only a generic systems programming environment. The functionality and real-time data structures developed for the user layer model require extensive services from the application layer.

Just what is the user layer model? We have attempted to project from today's smart transmitter environment, to the next level of smartness for field devices. It was clear in 1987, that one or more vendors would move the final closed loop control calculation to a field device; a transmitter or a valve positioner. We didn't expect these products so soon, but they are already for sale and were exhibited at the 1990 ISA Conference and Exhibit.

PROTOCOL MODEL

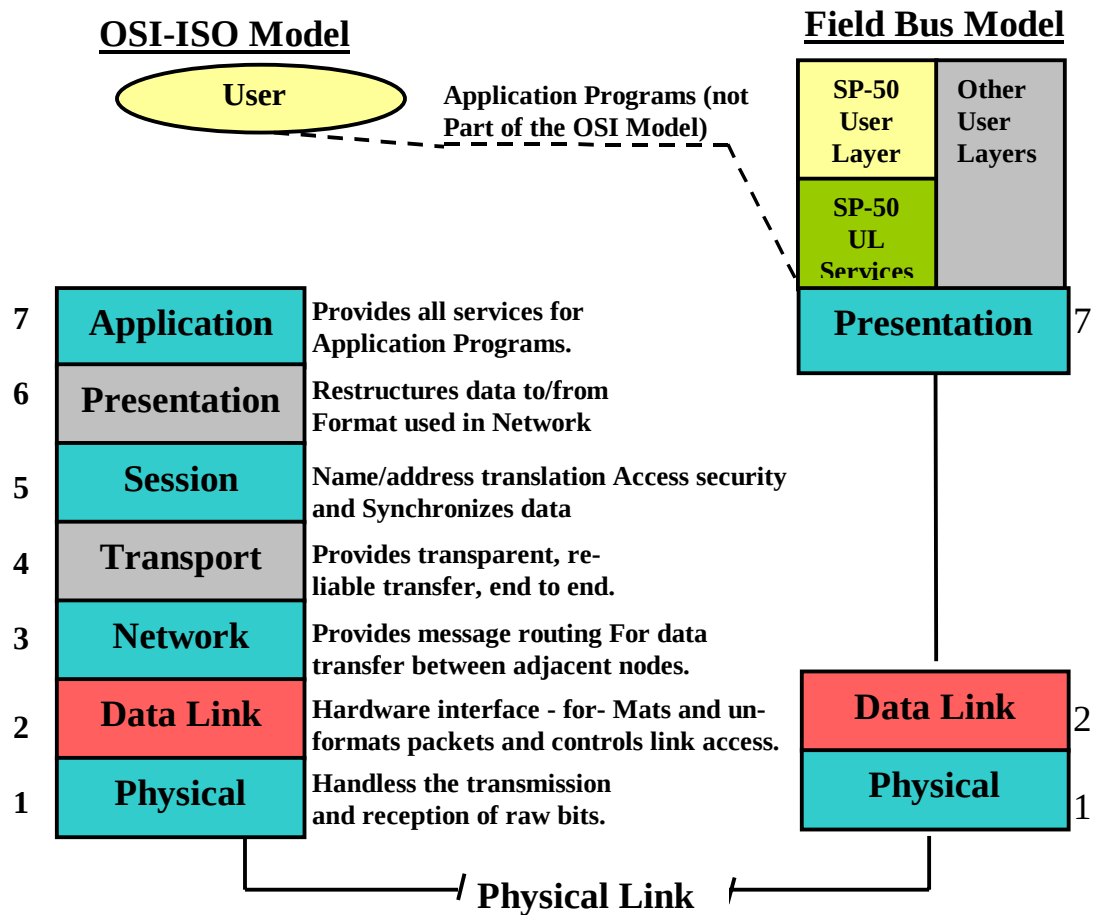


Figure 1: Preface, Figure 1, Protocol Models

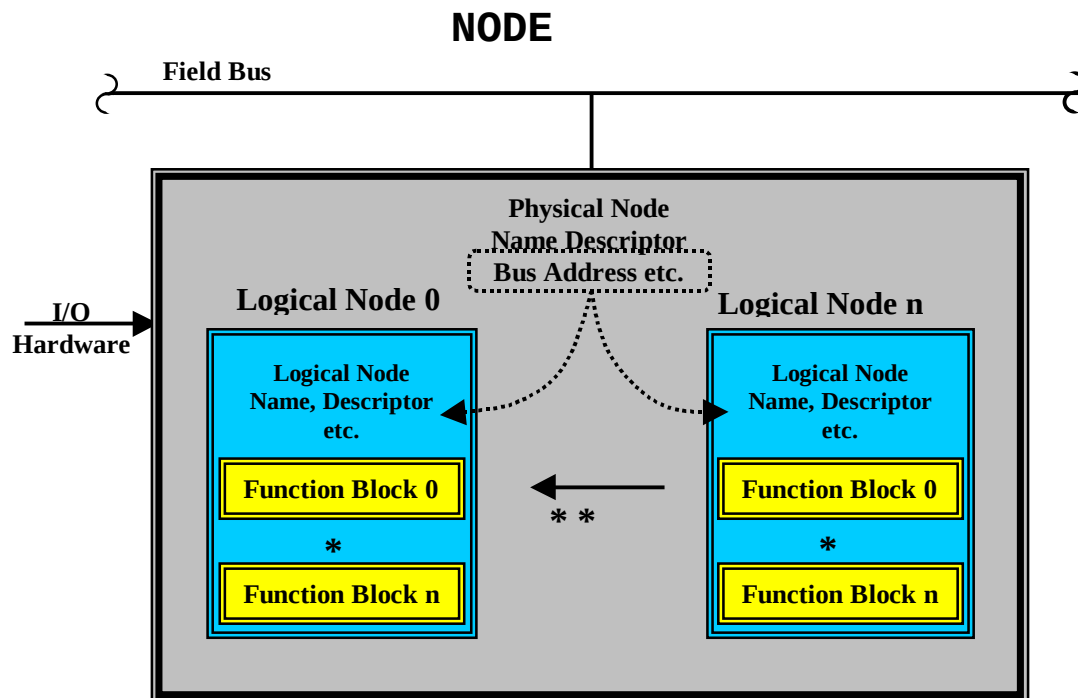


Figure 2: Node

So far these products still depend upon the use of either Proprietary or limited availability bus technology working together with 4-20 ma transmission, however, their control parameters are available on those vendors' systems products for remote tuning and control mode selection.

The ISA Field Bus User Layer is oriented about the process control problem assuming a complete distribution of the scan and control functions all within Field Bus devices. The capability is provided for measurement, control calculation and final actuation in the same or separate nodes on the Field Bus, as necessary. Rather than taking a generalized message passing structure, the User Layer has used a complete distributed object-oriented approach modeled after several existing distributed control systems (DCS) and programmable logic controllers (PLC) in common use. The choice of this model was very pragmatic -- the DCS architecture has evolved from centralized computer control with increasing distribution of functions and has been field proven to work. A model constructed using only message passing without a semantic message structure would require the invention of an unproven mechanism. The committee has chosen to adapt its semantic model from those in common use. In addition, the architecture for communicating with PLC's has been included in this same model. Methods used to configure or program DCS's and PLC's are not in the scope of this model.

USER LAYER DEFINITION:

How can the User Layer semantic model be described? In other words, what syntactical format can be used to express the process control semantic model? We have settled upon a two level procedure to describe the Field Bus User Layer as follows:

1. A description of the User Layer Machine; a description of the implementation of the User Layer on a hypothetical high level language computer base.
2. A document abstracting the User Layer Machine into a set of equivalent requirements which contain all of the critical elements of the referenced machine, but without the specific design content.

The first document is called the "User Layer Technical Report." It was written by a committee interested in preserving the design of distributed control systems (DCS) which have proven to be useful and sufficient for a wide variety of time-critical applications in data acquisition and process control. The format of the document is that of a software design specification or an implementation description, simply because it was easier for the committee to write. It reads similar to system descriptions of distributed control systems since this is the primary source material within the experience of the committee. Since considerable criticism has been offered about the "over-specification" of the Technical Report, the subcommittee has decided to prepare a separate standard document.

The second document is the true standard for the User Layer. It is written in "requirements language", but without an implied implementation. Where the method(s) of implementation are important to achieve the objectives of the standard, they are explicitly stated. This is the document which will be submitted for ISA (and eventually IEC) approval, but it will be incomplete without the Technical Report which provides examples and much of the rationale. The technical content of both documents are intended to be identical. The Technical Report will always be updated to reflect the current status of the standard for the User Layer.

DIFFERENCES BETWEEN DCS AND FIELD BUS:

The Field Bus Process Control User Layer provides an application environment not unlike that of a distributed control system (DCS). The intent was not to

replace the DCS, but to define the architecture necessary for a DCS to communicate with smart field devices when control actions occur in those devices. There are many differences between Field Bus User Layer functions and those typically performed by a DCS.

It has always been expected that the Field Bus would remove the functions of process variable processing from the DCS. Similarly, processing of outputs to proportioning process control valves and other actuator mechanisms is also included. DCS's operating with Field Bus devices will acquire process variables in engineering units form. Likewise they will send a desired valve output as a percentage.

Inclusion of control objects within Field Bus devices changes DCS functionality more substantially. We expect that many smaller systems will not require the traditional DCS at all, but will only require DCS-like operator console(s). Larger systems, which will require a DCS, will use it to provide the upper level cascade controls and feedforward controls. While Field Bus User Layer provides some capability to build complex cascades and feedforward loops, these are intended to provide the backup to more complex loops built in the DCS.

In all cases, we attempted to keep the Field Bus simple. For example, there is no "ramp and soak" algorithm, because the committee felt that any algorithm we could design would be inadequate to fulfill the needs of true batch control. Yet, we decided to provide simple setpoint ramping, determining that this time-critical function was best administered at the Field Bus level. Only one PID algorithm was chosen, to keep it simple. Our guideline was always to include the time-critical items at the User Layer, but to eliminate the supervisory and high level calculation functions.

FUNCTION BLOCKS:

The architectural base of Field Bus is the function block, which performs the work of data acquisition, control and output. Every function block contains an algorithm, a formula or rule, and a data base that is used by the algorithm. The parts of the data base that are permitted to be accessed through communications are called "attributes" or parameters. Each function block has a user-defined name, called a block tag, which must be unique. Function block data is addressed over field bus via TAG.PARAMETER.

The User Layer Standard defines a "function block" to be a process- related data structure with a "data part" and a "rule part", as described in the following illustration:

Function Block Type

A class of user elements consisting of

- (1) The definition of a data structure partitioned into input, output, and internal variables, (data part, also called "data base") AND
- (2) A set of operations to be performed upon the variables of the data structure when the element is invoked (rule part, also called "algorithm").

Function Block

An identifiable and distinct instance of a function block type.

There are four classes of function blocks as follows:

- o Standard -- the attributes and the algorithm are fully defined within the standard such that both the dynamic and static performance are known. Both the algorithm and data base may be extended as long as the standard function remains.
- o Alternative -- the attributes include those of a standard block, but the algorithm may not provide the same static or dynamic performance as the corresponding standard block.
- o Generic -- the attributes are defined only in generic terms such

that any system may access them through the use of these attributes. The algorithm is not defined by the standard

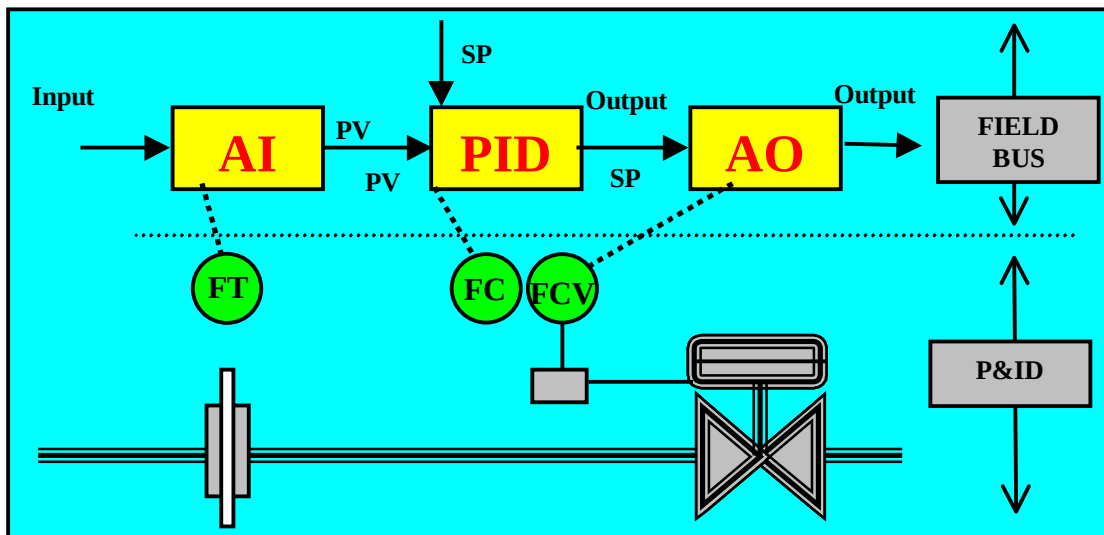
- o Open -- the attributes and the algorithm are not defined within the standard.

The intent of the function block structure is to allow devices to be delivered with pre-defined algorithms requiring only the data base to be downloaded for configuration into a specific control application. These are the Standard and Alternative blocks. A vendor may also deliver a Generic or a unique Open block complete with the algorithms pre-defined. In contrast, a vendor may also provide a start-up procedure, outside of the User Layer, to download algorithms to any type of function block, although this model is intended for use only with Generic and Open blocks.

Standard function blocks, which have been included, also provide interchangeability for the functions they define. The standard function block definitions allow extensions (i.e., functionality and parameters beyond those defined in the standard) and alternate function blocks (which use a different algorithm but the same parameters as a standard block). Extensions and alternate blocks, of course, may not be interchangeable. Generic and open function blocks accommodate both vendor-proprietary functionality and new field-device functions which are not practical to implement using combinations of standard function blocks. These classes of function blocks are not intended to be interchangeable. However, extensions and function blocks of any class which follow the user layer requirements for standard blocks, will be interoperable via field bus. The user layer also includes features of extended tag addressing and auto-formatting, which facilitate interoperability of extensions and "non-standard" blocks.

The following diagram shows the equivalency between "function blocks and conventional process and instrumentation diagrams (P&ID's):

Figure 3: Diagram: Function Blocks / P&ID



ABOUT PROGRAMMABLE LOGIC CONTROLLERS:

In the past five years, programmable logic controllers (PLC's) have become indispensable elements of all classes of process control systems. They are used to implement safety interlocks in continuous control and as the primary sequencer in batch control. Their low cost, flexibility and high energy I/O are as specialized for these applications, as the multifunction controllers of DCS's are for regulatory control of continuous processes. Therefore, the User Layer provides a rudimentary capability to communicate with PLC's modeled on the

available PLC interface techniques already in use. The PLC interface capability does not reflect the needs which might be developed in a User Layer for factory automation or machine control.

FIELD BUS DEVICES:

The Field Bus will allow many different types of devices to be logically interconnected on a single communications link. The following is a partial list of the types of devices that are anticipated:

- o Sensors and transmitters
- o Valve, damper and other position actuators
- o Single and multipoint regulatory loop controllers
- o Programmable logic controllers
- o Dual and triple modular redundant regulatory and programmable logic controllers
- o Safety shutdown systems
- o Multipoint data acquisition systems, analog and digital
- o Historical archiving devices
- o Operator (human) interface workstations
- o Chart recorders
- o Dedicated indicators and displays
- o Gateways to higher level communications networks
- o Bridges to other Field Bus segments
- o Temporarily connected devices
 - Hand-held human interface terminals
 - Communications link test devices
 -

At first, devices similar to present-day field devices, such as "smart" transmitters, positioners, single-loop controllers and indicators, and hand-held diagnostic devices will appear on field bus. In the future, new devices will appear that take full advantage of the capabilities of a standard digital field bus.

USER LAYER CONCEPTS:

The User Layer addresses a wide variety of real-time applications, but is designed specifically to support applications in closed loop process control. The control applications may be very simple arrangements of sensors, controllers, actuators and display devices all on the same Field Bus segment, or the devices can exist together in any combination. Any of the devices may also exist on a different Field Bus segment, however, it is understood that a performance or response time penalty may occur. We have also planned for the orderly connection of classical DCS or supervisory computer systems that are driven by a timing authority not under Field Bus control, yet provide essential target information.

All Field Bus devices are addressed at the User Layer by a tag name, rather than the low layer address necessary for the operation of the Data Link Layer. The translation of tag name to address is provided by the Application Layer based on User Layer definitions. The physical devices and function blocks all have unique tag names. This concept extends to initialization in which the physical device installed into the Field Bus can exist with only a unique tag name and expect that the data required to begin operation can be downloaded across the Field Bus identified only by that tag name. If a physical device contains only one logical node and only one function block (e.g. as in a single-measurement field transmitter), the device tag, node tag, and the block tag will be the same.

"Data ownership" is a primary concern of the process control user layer. It is essential from a user standpoint that there be one, and only one, location where all data for a given point exists. That location must be at the source of the data. Field devices will typically be the original source of information, and therefore the "data owner". All associated data resides in the field device, and not in a central controller, bridge, or gateway device. Each logical node or

"data owner" on field bus has a data dictionary which describes the data items "owned" by that node.

While the Data Link and Application Layers provide for a method of synchronous scheduling of function blocks, the User Layer provides its own method of scheduling centered on the time critical needs of control function blocks. Each control block reads the data it requires from a limited number of locations, and writes its output after completing its calculation. The time cycle of a control block is fixed by its configuration (data base), but the User Layer provides for an orderly recovery of missed cycles based on the process control model.

The functions defined for the field bus user layer support simple field sensors and actuators. The user layer function blocks provide for the essential interchangeability of simple field sensors and actuators, and at the same time enable manufacturers to establish product differentiation through functional enhancements. The function blocks defined in the field bus user layer also anticipate the development and use of field bus control devices. They provide a means to integrate "continuous" control, "analog" monitoring, and "discrete" status monitoring and logic control in the same system. The user layer function blocks provide a means of interoperability for presently-defined and future control devices from different manufacturers.

OBSERVATIONS AND CONCLUSIONS:

The Field Bus Application and User Layers are now emerging into a model for communications standards that has not previously been achieved. The transition from today's diverse process control architectures to a Field Bus architecture well integrated with a DCS will not be done overnight. We believe this change will be gradual over the next five to seven years, not because of the lack of technology, but more due to the conservative nature of the industry. This is the model for the field portion of that architecture.

Introduction

INTRODUCTION FIELD BUS PROCESS CONTROL USER LAYER TECHNICAL REPORT

The User Layer Technical Report is written as a functional design document for a hypothetical process control system. As such it is very complete in its description of data base structures and algorithmic implementation. The experience of the committee creating the User Layer was more easily focused on a document in this form. From the fixed structure and language in this Technical Report, the User Layer Standard could be formed by determining which parts of the "design" were the hard requirements and which reflected a model of an implementation. The completeness of the definition was also more easily tested using a model complete design, although the testing was primarily a "desk check".

The Technical Report can thus be considered to be a model of an implementation. That model is assumed to also form the base for the later effort of preparing a conformance test suite. Any implementation of a Field Bus device that responds to the conformance test in the same way as the model will be completely acceptable.

Document Arrangement

The immediately preceding Preface provides a brief history of the Field Bus and the User Layer as well as a tutorial overview. Immediately following this Introduction is a "Guide" to the Field Bus User Layer. The Guide is a very condensed "cheat sheet" that compresses many of the important data structures and concepts that are often necessary while referring to several sections. (The guide is not a part of the Hypertext version.)

The body of the Technical Report is divided into three main sections:

- o Environment -- Each paper describes a major concept and its construct. The concepts are roughly sequential, and often build upon previously introduced concepts.
- o Function blocks (algorithms) -- The first paper describes functions and features available to all standard function blocks. Then each standard function block is separately described with the exceptions to the shared features and the specific additional features necessary to define the block. The specific block algorithm follows. The language of the algorithm is a stylized form of structured BASIC, that should be very clear to any high level language computer programmer, although there is no exact language definition or an attempt to formally adhere to one.
- o Appendix -- The Appendix provides illustrations and rationale necessary to understand each major concept and provides in-depth analysis. There is nothing in the Appendix that is not contained in the first two sections. Much of the Appendix are the working papers used by the committee to prepare the Technical Report, and often explains "why" a particular concept was chosen over alternatives.

Conformance Classes

CONFORMANCE CLASSIFICATION

The Process Control User Layer (PCUL) is not intended to be a rigid standard, but is designed to allow vendors to implement some of the features, but not all of the features, and still conform. However, vendors choosing to implement less than all of the features defined by the PCUL must be able to declare their level of conformance so that users will have a rational expectation for such a system.

It is not the intent of this standard to imply that the highest level of conformance is necessarily better than any other level. Our intent is to define these levels in unambiguous terms.

It is the intent of this Conformance Classification to link to the definitions of "interchangeable", "interoperable", "interworkable", and "interconnectable" as defined in the paper ["Data Owner Structure - Logical Nodes"](#), pages 2 and 3.

In order that users may have an even clearer understanding of conformance classes, the diagram which follows provides for sub-classes of each of the main classes defined above. It should be noted that devices which do not meet the requirements for at least interconnectable are classified as "No Class", and cannot reliably be operated on the same Field Bus with any conforming device.

[Figures 1](#) and [2](#) detail the system that is to be used to describe the specific level of conformance.

CONFORMANCE CLASSIFICATION

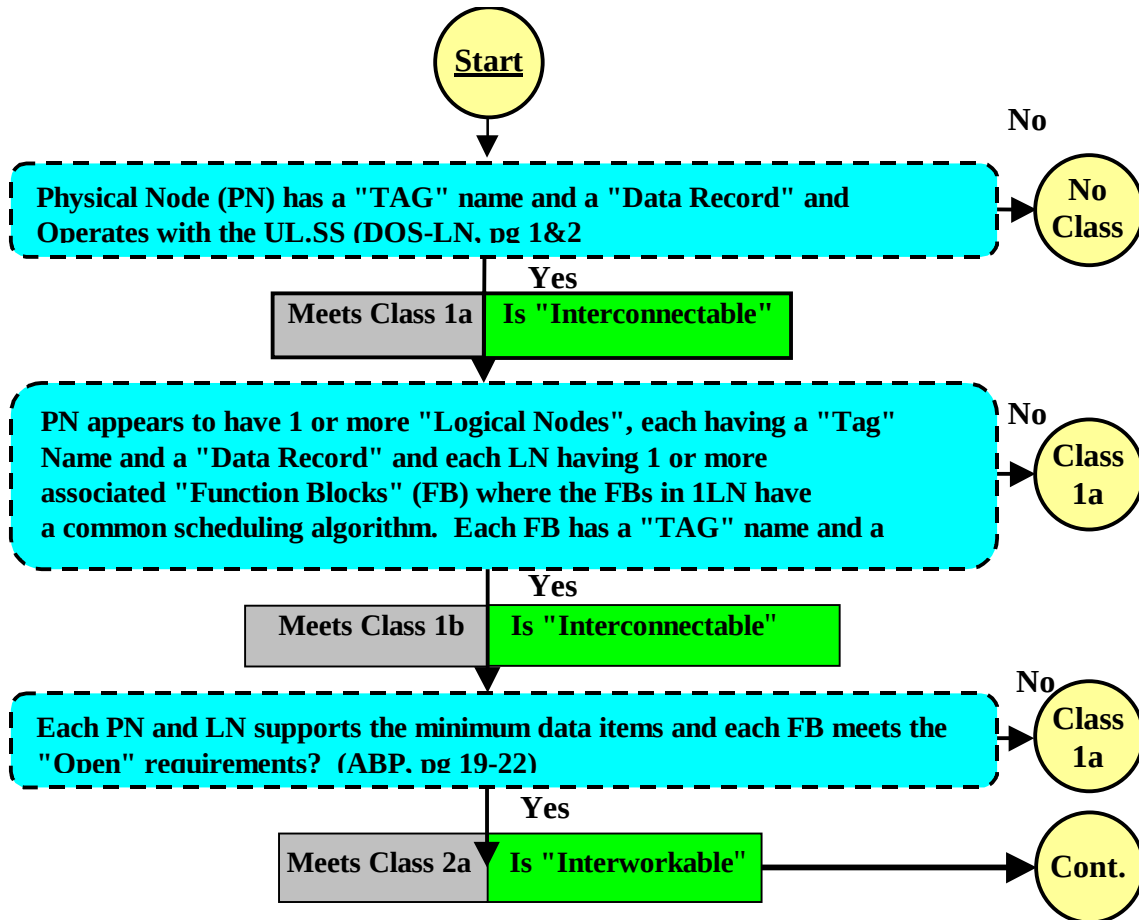


Figure 4 Conformance Classification

CONFORMANCE CLASSIFICATION-cont.

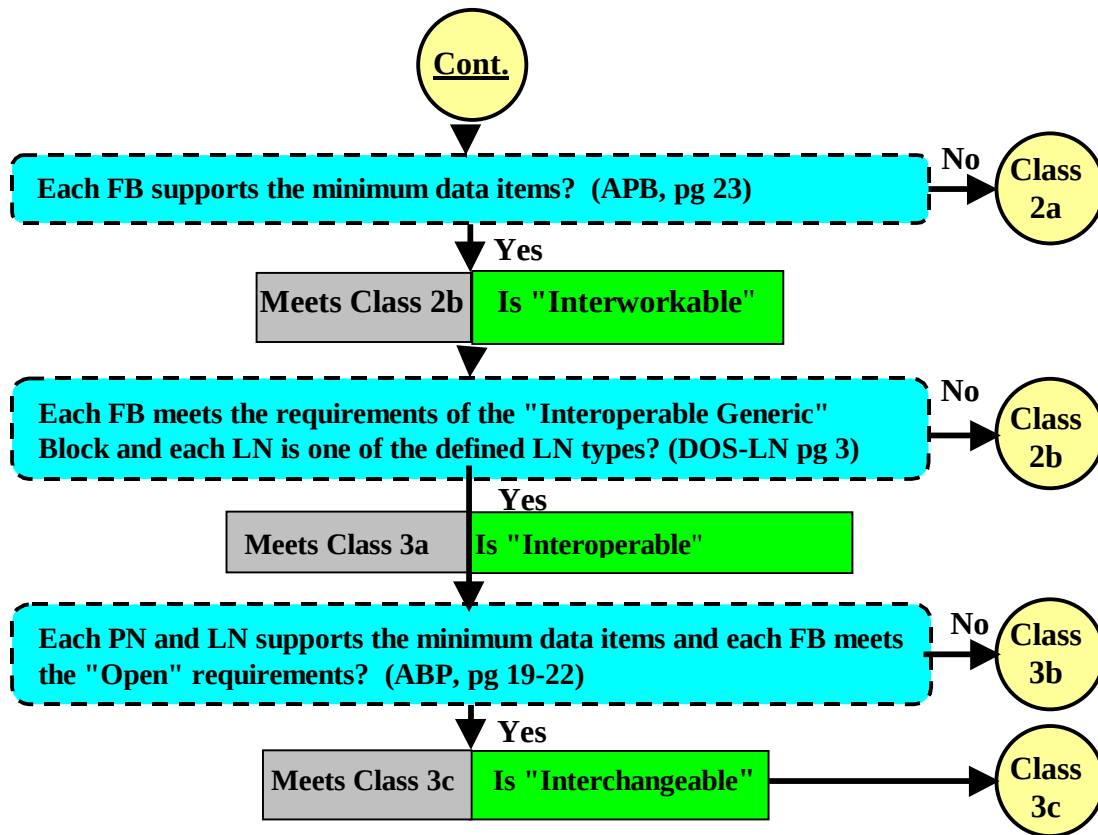


Figure 5 Conformance Classification (cont.)

References

References

NORMATIVE REFERENCES:

The following standards contain provisions which, through reference in this text, constitute provisions of this part of this Technical Report. At the time of publication, the editions indicated were valid. All standards are subject to revision and parties to agreements based on this part of this Technical Report are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

IEC 1158	Field Bus Standard
ISO 8824	Definition of ASN.1
ISO 8825	Basic Encoding Rules for ASN.1
ISO 7498/1	Open Systems Interconnection - Basic Reference Model
ISO 646	Binary Encoding for Visible Characters [Probably this will disappear in favor of our own list due to recent revisions of ISO 646.]
IEEE 754-1985	Binary Representation for Floating Point Numbers, ISO<***??
ISA S5.1	Instrumentation Loop Naming Standards

DENORMALS DEFINITIONS:

The handling of the denormal floating point numbers in Standard Field Bus devices will be as specified here:

- the 32 and 64 bit representations of -0 will be considered semantically equivalent to +0
- the 32 and 64 bit representations of - infinity and + infinity will be considered semantically equivalent.
- the 32 and 64 bit representations of NaN (all bits set in exponent, 1 or more mantissa bits set) will be retained.
- a negative NaN will be considered semantically equivalent to a positive NaN.

This Technical Report will define the handling of NaN.

SP50 User Layer Guide Sheet

Bit	Mode (Priority Order)
7	O/S - Out of Service
6	Iman - Initialization Manual
5	LO - Local Override
4	Man - Manual
3	Auto - Automatic
2	Cas - Cascade
1	Rcas - Remote Cascade
0	Rout - Remote Output

Figure 6: Modes by Bit

HARDWARE POINT TYPES	
Ct	- Counter Inputs
DR	- Discrete Registers
HI	- Human Interface
ME	- Foreign Memory
PO	- Pulse Outputs
SI	- Scalar Inputs
SO	- Scalar Outputs
SR	- Serial Comm. Ports
VD	- Voted Discrete Out
VS	- Voted Scalar Out
Multiplexors	
CO	- Composition
TK	- Tank Gauging

Figure 7: Hardware Point types

Bit	Status Info
0	No-Com - No Communication
1	Bad - Bad Value
2	NFP - Not from Process
3	Spl - Remote Blk; Hedw - LO
4	Fail - Failure in Path to Proc.
5	No-Path - No Path to Proc.
6	ATW-h - Wound-up-High
7	ATW-l - Wound-up-Low

Figure 8: Status Info. by Bit

Environment

BLOCK ALGORITHMS:	
<u>Analog Blocks</u> Analog Input Analog Output Counter Dynamic Compensation Math On/Off Control PID Property Conversion Pulse Width Modulated Signal Capture Signal Characterizer Signal Selector Splitter Timer Totalizer Tuner <u>Memory I/F:</u> Scalar (4) Register I/F	<u>Discrete Blocks:</u> Device Control Discrete Input Discrete Output Discrete Reg. I/O Human Interface Logic Program Control <u>Special Blocks:</u> Batch Collector Composition Tank Gauging <u>Serial</u> Serial Communications Serial In Conversion

Figure 9: Block Algorithms

LOGICAL NODE TYPE:

Undefined
 Unscheduled Standard
 Cycle / Phase Std
 Cycle/Phase
 CBX & PN by Block
 Timed Standard
 Cyclic Standard
 Cyclic

MASTER NODE**DATA**

Mfg. Set:
 MX=min exp
 MB=Maximum #
 Blocks
 MBC=Max # blocks/cycle
 User Set:
 CX=Time exp.
 CBC= current blocks/cycle

BLOCK TYPE:

Standard (base)
 Alternate(data of base, rule part
 free)
 Generic (min. data part defined
 generically, rule part free)
 (Any of the above can be
 Extended)
 Open (data and rule parts free).

CYCLE TIME=

2^{CX} sec
 -----or-----
 $2^{(CX+BX)}$ sec & phase = PN

Figure 10: User Layer Terms

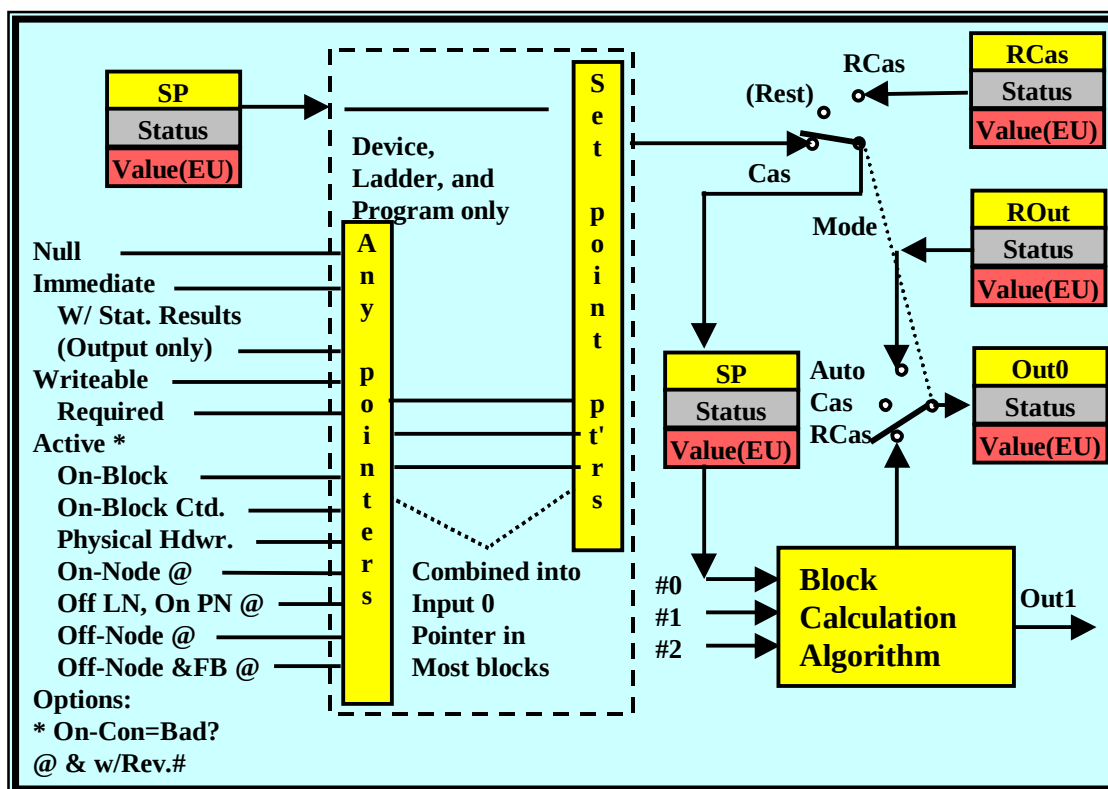


Figure 11: Cascade Structure

USER LAYER STRUCTURE

Overall structure of the data-owning
field devices and their interaction with
the data consumers on Field Bus

Data Owner Structure Hardware

Each of the devices that will connect to Field Bus will be a "physical node". From the perspective of Field Bus, it will have one Field Bus connection and a single bus address.

Some of the devices that connect to Field Bus will contain a set of data that pertains to a "tagged object". When those devices need to make the tag's data available to other Field Bus devices, they will be known as "data owners". This paper will define the physical node of a "data owner": the physical process connections, the structure and timing of the process input and output signals, and certain data and services pertaining to the whole device.

This paper is divided into the following sections:

- PHYSICAL NODE
- COMPLEX DEVICE
- LOGICAL NODE
- PHYSICAL PROCESS DATA CONNECTIONS
- PROCESS SIGNAL ACCESS
- PROCESS I/O SERVICE TIMING
 - Input Execution - CT Points
 - I/O Execution - DR Points
 - I/O Execution - FM Points
 - I/O Execution - HI Points
 - I/O Execution - PO, SI, and SO points
 - I/O Execution - SR Points
 - Input Execution - CO
 - Input Execution - TK points
 - Output Execution - VD and VS
- PHYSICAL NODE I/O PARAMETERS
- UNUSED PHYSICAL OUTPUT HARDWARE
- FAIL-SAFE TRIGGERS
- PHYSICAL NODE CYCLE TIME
- SYSTEM CLOCK
 - Field Bus Time
 - Private Clock
 - Independent User Layer Engine
 - Alarm Clock
 - Basic Alarm Clock
 - Alarm Clock Retention
 - Time Parameters
- PHYSICAL NODE OPTIONS BIT STRING
- MISCELLANEOUS PHYSICAL NODE VARIABLES
- MISCELLANEOUS HIGHERLEVEL SYSTEM VARIABLES
- Figure 1 - A Field Device on Field Bus
- Figure 2 - A Complex Field Bus Device

PHYSICAL NODE:

Any device connected to Field Bus will be considered, from the perspective of the Field Bus, as having one, and only one, physical connection to a particular Field Bus. Such a device will be identified as a "physical node". A physical node is uniquely related to a Field Bus physical media connector. The application layer protocol will define the mechanism that is employed to assign a Field Bus ADDRESS to a physical node. From the perspective of the User Layer, there is only one address for a total physical node. (The "address" stored in the User Layer memory will have the "node number" in the high byte and the "Subnode Number" in the lower byte - see the lower communication layers for definitions.)

There will be certain data that will be accessible to Field Bus that is related to the physical node itself, not to any subordinate structures. Therefore, the physical node will contain, and relate uniquely, to a particular data record. The data items will be addressable by the physical node "TAG" plus the appropriate "parameter code".

The physical node TAG will conform to the naming conventions of the Field Bus protocol. Specifically, the TAG will be a contiguous string of up to 16 visible characters, padded to a full 16 characters with blanks (character 32[10]) in the right-most character positions. The only visible characters that can be used in the TAG are the upper case Latin letters (visible characters 65[10] - 90[10]), the 10 numbers (visible characters 48[10] - 57[10]), and a dash ("-"), visible character 45[10]. In addition, the dash can not be the first nor the last character in the tag name and at least one of the visible characters must be a Latin letter or a dash. The TAG will be set using the special Application Layer service (see the paper "[Application Support Services Interface](#)"). A thirty-two character visible string variable, "TAG_DESC", will be supported for every physical node; it will be a User entered descriptor of the node.

There is much controversy about the use of the dash in the tag name. Specifically, the dash is interpreted by some parsers as a minus sign. On the other hand, ISA S5.1 specifically recognizes the dash. This Field Bus Standard is in accord with S5.1: it specifically accepts the use of the dash and requires the absence of the underbar in all tag names in all field devices. However, some higher level systems will intercept all "upward" transmissions of tag names and convert all dashes to underbars (and likewise change underbars to dash in all "downward" transmissions). It is strongly suggested that Field Bus human interfaces such as hand held communicators should provide a method of selecting (on a universal basis) between the two characters in any display or entry of a tag name. The ultimate requirement for the field device's memory, however, is the dash, NOT the underbar.

The parameter code will be a 13 bit integer with three attribute bits associated with it. Many parameters are defined in the papers "[Array of Basic Parameters](#)" and "[Array of Standard Block Parameters](#)".

COMPLEX DEVICE:

Two or more physical nodes may be associated together to form a "complex" device. In such an arrangement, each physical node is associated with its own Field Bus physical connection and unique address and has its own data records. The complex device will not be known to the Field Bus in that it will not have an address nor data records nor any accessible data.

Two types of hardware connections will be defined below, the voted discrete output and the voted scalar output, that will recognize that the output hardware signals of the physical nodes are voted by the complex device and the result of the vote is the actual output of the device. In addition, certain input signal connections will be defined in a later paper (Off Field Bus Agents in the paper "Agents"), that allow parameters to be accessed across physical nodes in one complex device.

LOGICAL NODE:

Within a physical node, there will be one or more "logical nodes". From the perspective of the Field Bus, each logical node will be a separate software structure in the Field Bus environment.

Each logical node within a physical node will have an assigned TAG. This name will conform to the naming conventions of the Field Bus protocol. It will be entered as the value of a parameter of the physical node. Once the logical

node's TAG is configured, the data parameters of the logical node will be addressable by the logical node's TAG.

One or more "Function Blocks" will be coordinated by a logical node. The function blocks execute a set of rules (their "algorithm") against their inputs and data items to generate a set of outputs.

A separate document ("Data Owner Structure - Logical Nodes") defines the Logical node structure.

PHYSICAL PROCESS DATA CONNECTIONS:

The process data will be modeled as connected to the physical node. The Field Bus Standard recognizes the following classes of process signal physical connections to a Field Bus physical node:

CT - Counter Inputs	PO - Pulse Output
DR - Discrete Registers	SI - Scalar Inputs
FM - Foreign Memory	SO - Scalar Outputs
HI - Human Interface	SR - Serial Communication Ports

The Standard incorporates provision for devices that are data multiplexers. These provisions include special node types and special blocks designed for the following types of data/applications:

- CO - Composition
- PS - Property Scanners
- TK - Tank Gauging

The standard Analog Input block and the standard Scalar Input physical point type can be used in a multiplexed AI system as defined.

The Standard also recognizes two classes of physical outputs that can be used in (and only in) a complex device. They are:

- VD - Voted Discrete Outputs
- VS - Voted Scalar Outputs

There can be uncertainty in definition for several of the I/O classes:

The Counter Input is intended to refer to a connection that includes a counting register that counts the number of OFF TO ON transitions of the input between executions of the associated block and reports the INCREMENTAL counts, not the current state, to the block. Any field device that counts off to on PLUS on to off transitions will be non-standard. The design is based on the assumption that there are many counts per logical node cycle. There may well be cases where a discrete signal is connected to a block that, itself, counts transitions. However, in such a case, the transitions must occur at a rate that is slower than the cycle time of the node; such signals would be classified as Discrete Register inputs, not counter inputs, despite their ultimate use.

A Discrete Register can contain up to 16 discrete points. The manufacturer defines the exact number. For example, the manufacturer might choose to divide 16 discrete points across 4 registers of 4 points each. The discrete points are modeled as being configurable by point for either input or output service but the manufacturer can force the definition.

The Foreign Memory class refers to an association with the contents of a register or discrete in a memory that is remote from Field Bus and is not designed according to the Field Bus model. An agent for a function block will identify the specific location in the foreign memory using only an integer. The agent will be able to read and/or write to the foreign memory.

The Pulse Output shares many characteristics with a single point in a discrete register. As with the Counter, the difference is a matter of timing. The Pulse Output is intended for applications where the hardware can or does manipulate the output signal on a time basis that can be faster than the logical node scan time. The term "manipulated" implies an action more complex than a simple delay-off timer.

The "Scalar" hardware types are those hardware connections that connect 1 primary scalar variable to the physical node. There may be secondary variables associated with the primary variable but those secondary variables will not automatically have their own tag names. The scalar hardware type does NOT include variables interfaced using industry standard serial data communication protocols.

The Composition and Tank Gauging blocks are not only designed for multiplexer service but are also designed for a specific class of application.

From the perspective of a physical node, the VD class is essentially the same as the DR class except all of the points in the VD class are output signals and the complex device is known to be voting the outputs of two or more physical nodes to determine the final physical output. Therefore, there are different alarm conditions to be detected and reported.

The VS class is similar to the SO class except that the complex device is known to be voting the outputs of two or more physical nodes and, as for the VD points, there are different alarm conditions.

If an interoperable Field Bus physical node is built with any of the above classes of physical connections and they will be addressed by the structures within the logical nodes, then the external physical connections will be labeled on the physical device with a hierarchical structure of the form:

{signal class}.{connection number}.{subidentifier} The "signal class" will be an identification consistent with, or an extension of, the above signal classes. The "connection number" will be the index number of the signal within the class of signals, starting with the number zero. The "subidentifier" will be used for compound signals and for the individual points in a discrete register.

A complex device will use labeling that is consistent with the above rules. Each array of output points will be clearly labeled as being associated with one of the physical nodes (i.e., of a class other than VD or VS) or as being a voted output (i.e., of the VD or VS class). The output points that are brought together by voting must have the same identification in each of the individual physical nodes (i.e., each VD-x for the complex device must correspond with the same VD-x in each physical device in that complex device where x is a single integer - zero based). If one or more of the physical devices in one complex device does not participate in the voting, then those particular physical devices must not use the corresponding VD or VS connection number.

The attached Figure 1, "A Field Device on Field Bus", illustrates the overall arrangement of an interoperable Field Bus physical node. Note that the primary hardware signals do not explicitly connect to any Field Bus visible data base structure. Rather, specific block configuration data are used to identify the connections. As shown by the dotted lines, some of the physical node data will be mapped into each of the logical node's data bases. Figure 2, "A Complex Field Bus Device", illustrates the arrangement of three physical nodes into one complex device. Note that the three Field Buses are separate and that one output connection is shown as being voted.

PROCESS SIGNAL ACCESS:

Each of the process signal "connections" will appear to Field Bus software structures as single entities. They will normally be addressed by the "blocks" (see the separate paper ["Data Owner Structure - Logical Nodes"](#) for a description of blocks) within the associated logical node as:

{connection number}

The block hardware pointers do not have to explicitly identify the signal class because that is implicit in the block algorithm. All Standard block types that can address hardware are limited to addressing only one type of hardware.

The connections can only be made between the process connections and hardware type agents (i.e., blocks) 1) of the appropriate signal type and 2) from blocks within the same physical node. For example, the connection between an AI block and its physical data, an SI input, is made with one hardware type agent. The agent's pointer definition resides in the AI block data base. A number of values are probably transmitted in both directions over that single connection. The data structure of the software connections to the process signals need not be defined by Field Bus since the connection is within the physical node: the manufacturer has complete control of both ends of the connection.

The connections in one signal class will be defined in the physical node as a single dimensioned array. It will frequently be assumed, for variable definition purposes, that each output has a feedback self-check circuit (this does not mean a feedback from the process, only from the device's own outputting circuit). Also, some signals will be assumed to have multiple external variables associated with them. For example, there might be a feedback of the valve stem position associated with a modulated valve output; it will be labeled on the hardware as associated with the primary scalar output using the "subidentifier" notation.

Some system designers may wish to design the scalar physical hardware so that the user can configure it as either an input point or an output point. This will be allowed and implemented in the following way. The physical point will be designated as SI/SO-x where x is a zero based integer. In that physical device, all values of x are then eliminated from the numbering scheme for the simple SI's and SO's. For example, if there is an SI/SO-3, there can not be an SI-3 nor an SO-3. The user then can configure either an AI or an AO function block to SI/SO-3 but not both.

The discrete points will be structured in sets of up to 16 individual points. A set will be called a "register". The "subidentifier" designator will identify the discrete point within the set of 16; it will be an index number starting with zero and will be referred to as "bit".

The Discrete Register block is defined on the basis that the discrete points within a register can be identified by the user as inputs or outputs on a per-point basis. If that is not the case, the data base identifier for this option (within the connected Discrete Register block's data base) can be forced by the manufacturer to be read-only and fixed at the predetermined value providing a write to that location receives a response indicating that that value is "read-only". In addition, a Discrete Input block must not be allowed to be connected to any I/O connections that are Output only and vice versa (a violation will generate a "configuration" notification).

The Discrete Register block is also defined on the basis that the discrete points, when configured to be outputs, will be able to be configured to implement feedback checking of the discrete output on a point by point basis. This checking is assumed to be a check of the output circuit of the physical node, not a feedback from the process equipment being driven by the discrete output. It is assumed that this checking can be optionally disabled. The "disable checking" option is a defined data base item within the connected block's data base. If the manufacturer chooses to not offer the checking option, he can force the disable option to always be true providing a write to that location receives a response indicating that that value is "read-only".

PROCESS I/O SERVICE TIMING:

The timing of the physical input and output of data is loosely defined for some applications but it is carefully defined for applications in which it is desirable to be able to "interchange" Field Bus devices from different manufacturers. The following discussion will indicate the sections of the I/O design that, with careful design, can achieve interchangeability.

In the following discussions, all references to "block time" will be understood to refer to the block's scheduled time period as defined by the current configuration of the block cycle times and number of blocks.

Input Execution - CT Points:

The physical capture of counter input data will be driven by the execution of the logical node cycle of the logical node in which the associated blocks reside, not the individual execution of the blocks associated with the I/O. It follows, then, that this timing definition can not apply to the cycle/phase nor the unscheduled node types. In those nodes, the timing will be defined by the manufacturer and the device will not be interchangeable.

The incremental count value will be captured after the start of the logical node cycle but before the execution of Block 0. The capture for all counters in the physical node that have their associated blocks in one logical node will occur "at the same time". The incremental value will be available to any number of function blocks in the same logical node that wish to access it. However, if a block in any other logical node in this physical device attempts to access the same hardware, the second access will cause an immediate block configuration error notification and O/S mode for the second block. However, the user must be able to change the hardware pointer(s) on the block(s) in the first logical node and then address the physical hardware point from a function block in a different logical node in the same physical node without receiving an alert.

These points are potentially "interchangeable".

I/O Execution - DR Points:

The Standard defines three different types of blocks that can interface to discrete I/O. One of those blocks can interface to a complete discrete register, the other two can interface to a sub group of the points (1 to 16 bits wide) in a single register or across adjacent registers. Two of the blocks are unidirectional (Input only or Output only) while one can contain a mixture of both Input and Output discretely within its one register. The manufacturer is expected to support all of the variations required by the algorithms that are offered in the node.

A discrete register can contain up to 16 discrete points. The manufacturer defines the exact number. For example, the manufacturer might choose to divide 16 discrete points across 4 registers of 4 points each. If there are only 4 points in a register, then the associated Discrete Register block would service 4 points, not 16. It is required by the Standard that the hardware type agent of the Discrete Register block addresses a single, full register. That register may contain fewer than 16 discretely; the "first" discrete in the register corresponds to bit position 0 in the block's register. The succeeding bits follow in order. The Discrete Register block includes a mask that

can mask off one or more of the bits but the mask can not shift the positions of the bits. The Discrete Input and Discrete Output blocks can define the number of bits in their groups and the location of the first bit. The range thus defined can span across adjacent I/O registers.

It is required that the (user) configuration of the device must arrange the blocks associated with the discrete I/O in such a way that each discrete register is accessed by blocks from only one logical node. If a block in any other logical node in this physical device attempts to access the same hardware, the second access will cause an immediate block configuration error notification and O/S mode for the second block. However, the user must be able to change the hardware pointer(s) on the block(s) in the first logical node and then address the physical hardware point from a function block in a different logical node in the same physical node without receiving an alert.

The physical capture of discrete input data (as distinct from output feedback data) will be driven by the execution of the logical node cycle, not the execution of the blocks associated with the I/O. The raw input values will be captured after the start of the logical node cycle but before the execution of Block 0. All discrete input signals in a physical node that have their associated blocks in a particular logical node will be captured "at the same time". However, the "finished" data values in the discrete input function blocks must not be available to the ultimate user blocks until the input blocks execute.

The physical output of discrete data can occur at any time in the logical node cycle between:

- 1) the start of the execution time of the discrete I/O function block driving the output
- AND
- 2) the start of the execution time of function block 0 in the immediately next cycle of the logical node.

The acquisition of the output feedback data can occur at any time in the logical node cycle as determined by the manufacturer. There may be cycles in which feedback checking can not be done because a recent change in the output may not have returned in the feedback circuit. The Field Bus functionality model assumes that the miscompare data is available at the beginning of the block execution time of the discrete I/O block.

The input and output of data associated with discrete physical I/O will involve the execution of procedures by both the hardware and the block and will also involve the transfer of data to/from the physical I/O. If a second block, in this logical node or any other logical node in the physical device, attempts to access the same hardware later in the same cycle, the second access will cause an immediate block configuration error and O/S mode for the second block. Note that this restriction applies to each discrete individually and must recognize the mask that is provided in the Discrete Register block. (The discrete register block allows selected bits in a hardware register to be masked out so they can be addressed by the Discrete Input and Discrete Output blocks).

These points are potentially "interchangeable".

I/O Execution - FM Points:

The function blocks that use hardware agents that access foreign memory are based on a model in which the foreign memory access is accomplished by one of two methods:

- 1) "prefetching" and "poststoring", just like communications over the Field Bus.

2) the use of a "read handler" in the node that can execute the "remote read" as part of the communications service. The exact timing of reading and writing from/to the foreign memory is not defined by Field Bus. It is not anticipated that the devices that use this type of hardware agent can be interchangeable devices.

There is no requirement that the Field Bus device must assure that access to the foreign memory data items is unique.

I/O Execution - HI Points:

The exact timing of signal capture and release in a Human Interface block is not defined by Field Bus. However, if a second block, in this logical node or any other logical node in the physical device, attempts to access the same hardware later in the same logical node cycle, the second access will cause an immediate block configuration error and O/S mode for the second block.

These points are potentially "interchangeable" since the application will not require tight timing.

I/O Execution - P0, SI, and S0 points:

The P0, SI, and S0 data types will be driven by the execution of their associated block. Consider an Analog Input block and the SI hardware to which it points. The actual physical "snap" of the SI measurement is to occur after the start of the execution of the AI block but before the end of the first 10% of the block's time. This requirement recognizes that there may be an analog filter ahead of the A/D converter: its effect is not included in the requirement. Also, if the A/D conversion uses any kind of counting mechanism, the "snap" refers to the termination of the count.

For an A0 function block, the conversion of the digital output to its physical form is to take place before the end of the first 25% of the block's execution time. However, it is acceptable that the physical circuit not have completed slewing to the indicated value before the end of this time limit.

For a Pulsed Output point, the physical output will be operating according to the new request from the block by the end of the first 25% of the block's execution time subject only to the constraints of the definition of the pulse.

It is assumed that the execution associated with the P0, SI, and S0 physical hardware will involve both the execution of procedures by the hardware and the block and also the transfer of data between the block and the hardware. If a second block in this logical node, or a second block in any other logical node in this physical node, attempts to access the same hardware, the second access will cause an immediate block configuration error notification and O/S mode for the second block. However, the user must be able to change the hardware pointer on the first block and then address the physical hardware point from a different function block in any logical node in the physical node without receiving an alert.

These points are potentially "interchangeable".

I/O Execution - SR Points:

The exact timing of signal capture and release is not defined by

Field Bus, even for interchangeable devices. The Standard Serial communication blocks assume that the hardware port is "freerunning" and may or may not have room in its buffer for an outgoing message and may or may not have an awaited response. They also include provision for several blocks driving the same port, each block with its own "channel" number. All of this applies across all logical nodes within the same physical node. However, if a second block with a hardware pointer to the same port and the same channel number is executed later in the same logical node cycle, the second access will cause an immediate block configuration error notification and O/S mode for the second block. Of course, if the manufacturer does not support channel numbers, then any second access will cause the error condition.

These points are potentially "interchangeable" since the application will not require tight timing.

Input Execution - C0:

The composition point is assumed to reside in "unscheduled" type logical nodes. Therefore, these points do not have defined timing for their input and are considered "interoperable".

If a second block in this logical node attempts to access the same component, the second access will cause an immediate block configuration error notification and O/S mode for the second block. However, the user must be able to change the component pointer on the first block and then address the component from a different function block in the logical node without receiving an alert.

Input Execution - TK points:

The TK input type is assumed to be multiplexer type data. Its associated function blocks are assumed to be in a cycle/phase type node. Each point will be input in the logical node cycle specified by the cycle and phase data configured into the associated individual function blocks. The exact time within the logical node cycle is not defined. Therefore, these points are considered "interoperable".

It is assumed that the execution associated with the TK physical hardware will involve both the execution of procedures by the hardware and the block and also the transfer of data between the block and the hardware. If a second block in this logical node attempts to access the same hardware, the second access will cause an immediate block configuration error notification and O/S mode for the second block. However, the user must be able to change the hardware pointer on the first block and then address the physical hardware point from a different function block in the logical node in without receiving an alert.

Output Execution - VD and VS:

The voted outputs, both VD and VS, are modeled as being voted external to the operation of the individual physical/logical nodes. Therefore, the output from a particular physical node (going to its voting "circuit") will follow the timing constraints of the corresponding non-voted output as defined above.

There are no defined constraints on the timing of the output signal going through the voting "circuit". The individual I/O algorithms will assume that the physical node hardware will handle, in a manufacturer defined way, the timing considerations of the feedback checking signals. These points are potentially "interchangeable".

PHYSICAL NODE I/O PARAMETERS:

It is expected that manufacturers will design a method of calibrating the physical node's hardware. If the user is to interact with the calibration data, there must be a defined method of accessing it. This will be supported using a special data type called the "auto-formatting variable"; it will be defined in the "Human I/F Considerations" paper. There may be one of these parameters in the physical node's data base for EACH physical hardware point and/or one for each TYPE of hardware. They are called I/O_AF and I/O_TYPE_AF respectively in the "Array of Basic Parameters" paper. The method of indicating the presence of this data will be defined in the "Application Support Services Interface" paper.

The "Application Support Services Interface" paper will also define the method of indicating the presence of an additional (3 times 64) auto-formatting variables that can be in the physical node's data base. These variables are not defined in association with the physical node's hardware. They are labeled PN_AF_x in the "Array of Basic Parameters" paper.

There may be a need for another addressable parameter in the physical node's data base for a hardware point. Consider a device design in which it is possible for the block execution to fail but communication services to survive the failure. In that case, it may be advantageous to provide a communication path from the Field Bus directly to the hardware. Given such a path, the user could retain emergency control of the process. This path is supported by an optional defined parameter in the physical node that can directly address the hardware. The parameter can only be written when the computing engine driving the function block processing is stopped.

Support for this parameter will be indicated by a method defined later in this paper. The data format of this parameter will be defined in the function blocks that connect to the particular hardware. The parameter number will be defined in the "Array of Basic Parameters" paper.

UNUSED PHYSICAL OUTPUT HARDWARE:

A service must exist in the physical node to set all physical outputs that are not addressed by a function block to their no-power state. This is of no significance to the communication port or to foreign memory interfaces but it is significant to the Discrete Registers, the Human Interface, the Pulse Output, and the Scalar Outputs.

The logic defined in the "Field Device Start and Restart" paper includes a detailed description of the logic that determines if any output hardware points are unaddressed and sets them to their no-power state if it can be done safely. That logic is automatically done at start-up of the physical node.

A single discrete value called "CLEAR_OUT" will exist in the physical node's dynamic data base. When that value is set, the physical node will:

- 1) Reset CLEAR_OUT
- If the physical node contains any DR, DO, PO, or SO hardware, then
- 2) execute the data base checking portions of the start-up logic necessary to confirm that all function blocks have a valid data base,
- If all data bases are valid, then
- 3) determine if any physical hardware capable of outputting are unaddressed.
- If any are unaddressed, then
- 4) set them to their no-power state.

This operation will be executed when "CLEAR_OUT" is set. It is expected that time to execute this infrequent function will be generated by having all of the logical nodes in the physical device suspend for a short time.

FAIL-SAFE TRIGGERS:

The manufacturer of a Field Bus device may optionally include the capability to connect discrete signals to the device that can be configured to cause the outputs in the device to assume their "fail-safe" state.

The external fail-safe trigger signals will be attached to the physical device in such a way that they will not be confused with the discrete register process signals. There may be a single external signal that will trigger all of the logical nodes of the device or there may be separate external signals for each (or selected) logical node. If an external signal does not trigger all logical nodes in a physical node, it will be labeled with the logical node number (a simple index number starting with 0).

The Fail-safe Logical will be a parameter associated with each logical node. There will be no "Fail-safe" logical in the physical node. The logical will be set by the manufacturer optional hardware input signal or may be written over Field Bus. See the paper "[Data Owner Structure - Logical Nodes](#)", page 20 for the details of the software implementation of the Fail-safe function.

Although the physical node does not appear to control the Fail-safe logic, there are two parameters associated with the Fail-safe logic that are housed in the physical node's data base - SFAIL-SAFE and RESTART. See the paper "[Field Device Start and Restart](#)" for descriptions of these parameters. These two parameters must be supported by every physical node. Note, however, that SFAIL-SAFE is self-defining; a manufacturer can indicate in it that it does nothing. The RESTART parameter is another discrete that must exist but the SFAIL-SAFE parameter can define that it does nothing.

When the external signal OPENS (current flow below some manufacturer set minimum), the logical will be set by the leading edge of the change in current flow. The logical will not reset when the external signal closes. Rather, the logical will remain set until a Field Bus command resets it. In the paper "Field Device Start and Restart", the details will be given for the operation of the Fail-safe system during a device restart. Briefly, in either a "warm" or "cold" restart, but not during a "hot" restart, the logical will be initialized based on the external signal's STATE. That is the only time that the logical's operation will depend on the state.

PHYSICAL NODE CYCLE TIME:

Most of the work of a physical node is scheduled by and within the logical nodes. However, the paper "Alert Function" will define a method for the physical node to report its alerts. As part of that method, there is a need for the concept of a cycle time for the physical node (see the section "[Buffer Overload Indicator](#)" in the paper "Alert Function"). The "alert cycle time" variable in the physical node's data base will define the time period of the major self checking diagnostics of the physical node.

This standard does not define nor require any physical node self diagnostics - it only provides a method of communicating such alerts if needed. Within that scope, it is assumed that there are two types of self diagnostics conducted by the physical device. One might be called "slow speed" - it is assumed to involve a relatively long series of tests, perhaps conducted as a background function, to check the memory of the device and other low priority or calculation intensive operations. The other might be called "high speed" - it is assumed to involve

relatively frequent checking of the physical I/O, the microprocessor itself, power supply levels, etc.

```

Every physical node will have a read only, 8 bit signed integer variable "AX"
in its data base. To define AX:
  IF the physical node does high-speed self diagnostics, then
    let x = the high speed self diagnostic cycle time, in seconds
  ELSEIF the physical node checks for changes in the value of TIME_BIAS
    defined in the next section, then
    let x = the frequency of those checks, in seconds
  ELSE
    let x = 2**-127
  ENDIF
AX = an integer such that
      2**(AX-1) < x <= 2**AX

```

SYSTEM CLOCK:

The primary basis for "time" in a Field Device will be the time information that is available from the communication services (see the paper "[Application Support Services Interface](#)"). In addition, the manufacturer may optionally support a "private" clock. This section will detail how that data will be used. In addition, the manufacturer's option to allow the User Layer logic engine to operate even though the communication engine has failed generates some clock duties for the physical node. Finally, there is an option for an "alarm clock" service.

Field Bus Time:

The "time" recognized by the Field Bus system will be based on the current International Standard Time. It will not be concerned with local time nor with any form of daylight saving time. The system's concept of the current International Standard Time is called "Social Time". It is the total time since January 1, 1970 at 00:00 AM International Standard Time till the current time, measured in 1/32 of milliseconds.

Two parameters that relate to system time will be available from the communication services. They are:

- 1) NODE_TIME = the elapsed time since the physical device was last powered up
- 2) NODE_TIME_OFFSET = the difference in time between January 1, 1970 at 00:00 AM, International Standard Time (i.e., the arrival of the new year, Greenwich Mean Time), and the last start-up of the physical node of the field device.

Both of these values are 48 bit integers expressing time in one-thirty-seconds of milliseconds (rollover after 279 years). Note that the base of January 1, 1970 is the same base used by UNIX systems.

There are no Standard function blocks that can access 48 bit integer values that are defined here.

At initial power-up, the communication service (see optional enhancement below) will reset both of these values, then begin to "tick" the NODE_TIME as defined in the Link Layer Standard. Since NODE_TIME_OFFSET can never have a legitimate value of zero, the reset value will be the indicator to all users that communication has not been established.

When communication is established, it is assumed by this paper that NODE_TIME_OFFSET will be set by the communication services.

The alert reporting system, the primary User Layer user of time, will access the communication layer service to get `NODE_TIME` for inclusion in each alert report that it prepares. The access will be done as part of the servicing of the alert at the time it is discovered. The time reported in alerts will be a 29 bit integer value formed by shifting the 48 bit time four bits to the right and using the resulting 29 low bits (a value that has a resolution of one-half of a millisecond and rolls over after 3.1 days). See the paper "[Alert Function](#)" for the details of the various alert reports to be issued by the physical node.

The physical node will inspect the value of `NODE_TIME_OFFSET` at least every "AX" cycle (see [section above for the definition of AX](#)) and report on any significant change using the alert reporting system. To support this operation, there will be a 16 bit integer parameter in the physical node's data base, called `OFFSET_DB` (required if alert reporting is supported), that will define the deadband on the reporting of changes in `NODE_TIME_OFFSET`. `OFFSET_DB` will also be expressed in terms of one/quarter of milliseconds, thus giving a maximum possible dead band of 16 seconds. If `OFFSET_DB` is reset, it will be changed to 1B.

In order to clarify the meaning of these terms and to show a minimum existence proof of the concept, assume that there is a value local to the physical node that stores the last reported `NODE_TIME_OFFSET` - call it "LR_OFFSET". Then:

```
At power up,
    LR_OFFSET = 0
Once each cycle:
    access NODE_TIME_OFFSET from the communication layer.
    IF NODE_TIME_OFFSET is available, then
        TEMP = (NODE_TIME_OFFSET - LR_OFFSET)
        TEMP = Absolute(TEMP)
        Shift TEMP right 4 bits
        IF OFFSET_DB = 0, then
            OFFSET_DB = 1
        IF TEMP > OFFSET_DB, then
            generate an alert report of the type defined in the
                paper "Alert System".
            LR_OFFSET = NODE_TIME_OFFSET
        END IF.
    END IF.
```

Private Clock:

There is a manufacturer's option to support a "private" clock. This clock will be assumed to be running independent of the Field Bus clock and will be associated with a LOGICAL node, not the physical node.

ALL time references in the Standard will be based on the Field Bus clock EXCEPT for a single logical node variable. That variable will be called `PRIVATE_OFFSET` and will be reported as a 32 bit signed integer. Assuming `PRIVATE_CLOCK` is expressed in the same terms as Field Bus time, it will be the value of (`PRIVATE_CLOCK - NODE_TIME - NODE_TIME_OFFSET`). Its low order bit will correspond to 1/32 of a millisecond.

If the manufacturer supports a `PRIVATE` clock and the alert system in the same logical node, the variable `P_OFFSET_DB` will be supported. It will be used to send an alert concerning the adjustment of the private clock by an amount greater than `P_OFFSET_DB`. The meaning of `P_OFFSET_DB` and the alert will be exactly the same as defined above for the corresponding system clock alert; a logical node code and subcode will be used for the alert.

Independent User Layer Engine:

There is a manufacturer's option to provide a logic engine for the User Layer that is separate from the communication service(s). This option affects the time and alert reporting services; its existence will be shown in the physical node's "option string" defined below.

When this option is present, the design of the entire User Layer of the device will be such that at least 128 bytes of alarm records can be accumulated while the communication services engine is not functioning. See the paper "[Alert Function](#)" for details concerning the alert system.

As defined above, when a physical node powers up, NODE_TIME will be reset. If the User Layer engine is separate, then, as soon as the User Layer is ready to process function blocks, but before doing so, the User Layer will request NODE_TIME from the communication layer(s). If none of the communication layers is able to supply NODE_TIME, the User Layer will begin to function and to "tick" the NODE_TIME. It will do so the best it can (to the quoted resolution and accuracy of its "User Layer clock"). When any Field Bus communication layer is ready to start, it will send a request to the User Layer for its NODE_TIME. If the User Layer has a non-zero node time, it will pass it to the requesting communication layer and, from then on, use the NODE_TIME from that communication layer.

If any communication layer can supply NODE_TIME when the User Layer starts, then the User Layer will use the NODE_TIME of that communication layer rather than its own.

If, during normal operation, NODE_TIME becomes unavailable from the communication layer, then the User Layer will try to get it from any other communication layers in the physical device. If it is not available anywhere, the User Layer will again use its own clock.

The detailed service will be defined based on the assumption that there is a local variable called TEMP_TIME in the physical node's data base. It is assumed to be a 48 bit integer but not available to Field Bus. No such variable is actually required but the device must function AS IF the following logic were present. Once each cycle:

```

At physical node start-up,
    TEMP_TIME = 0
At start up and every physical node cycle thereafter,
    IF NODE_TIME is available from the communication service that
        was used at the last cycle, then
        TEMP_TIME = NODE_TIME
    ELSE
        attempt to find other communication services that have a
        NODE_TIME.
        If available, then
            TEMP_TIME = NODE_TIME
At every event occurrence,
    IF NODE_TIME is available from the communication service that
        was used at the last cycle, then
        use it for the alert time stamp.
    ELSE IF NODE_TIME is available from another communication
        service, then
        use it for the alert time stamp.
    ELSE
        use TEMP_TIME.
```

The physical node is assumed to "tick" TEMP_TIME at all times.

It is assumed that, when there is more than one communication layer in a physical device, they will coordinate among themselves so that there is only one value each for NODE_TIME and NODE_TIME_OFFSET.

The User Layer will generate (probably into a static buffer!) an alert when it starts its own `NODE_TIME` that records that event (actually, this alert will be a RETURN of alert code 0). When the User Layer first gains access to a communication services version of `SOCIAL_TIME`, that event will generate another alert report (actually, this alert will be a RETURN of alert code 2).

Alarm Clock:

Since the time values discussed above can not be accessed by any of the standard function blocks, there is a need for some way for them to recognize and operate on a social time basis. That is accomplished by the optional "alarm clock".

Basic Alarm Clock:

This service requires two additional parameters. One of those parameters, "ALARM_TIME", is a 48 bit unsigned integer and the other, "ALARM", is an 8 bit binary string that can be accessed by function blocks. Bit 0 of the string will be set any time "ALARM_TIME" is non-zero and less than the sum of `NODE_TIME` and `NODE_TIME_OFFSET`, ELSE it will be reset.

Alarm Clock Retention:

In order to provide the highest degree of integrity for the alarm clock service, there will be a suboption associated with the alarm clock - called "alarm clock retention". In general, this feature will attempt to retain its clock setting through a short power outage and to retain the alarm service in spite of communication isolation after restart. If it fails to do so, Bit 1 in "ALARM" will be set. Bit 1 can be reset by a masked write. It will also be reset by the "Data Base Write Service" any time that `ALARM_TIME` is written.

This feature will be described on the assumption that there are internal variables called "TEMP_TIME" and "TEMP_BIAS". No such variables are required; they are used to illustrate the required functionality. These 48 bit integers are assumed to normally contain the `NODE_TIME` and `NODE_TIME_OFFSET` from the last physical node cycle and are not accessible from Field Bus. The physical node may or may not be "ticking" `TEMP_TIME`.>

```

    This option is best defined as a chain of logic:
    IF the communications layer can provide NODE_TIME_OFFSET, then:
        TEMP_BIAS = NODE_TIME_OFFSET
    ELSE
        IF this is the first cycle of the User Layer after starting up,
            then
                IF (the User Layer has a method of determining that its
                    TEMP_TIME and TEMP_BIAS were retained during the outage)
                    AND [the "Warm Restart Timer" did not expire during the
                        outage (see the paper "Start and Restart")], then
                        TEMP_BIAS = "Retained" TEMP_BIAS +
                            "Retained" TEMP_TIME + b
                        where b = the minimum time that it takes this
                            User Layer, in this device, to
                            execute a restart.
                        TEMP_TIME = 0
                ELSE

```

```

        TEMP_BIAS = 0.
        TEMP_TIME = 0
        set Bit 1 in ALARM.
    END IF.
    If the User Layer can not assure that its physical node
    static data base was retained, then
        ALARM_TIME = 0
        reset Bit 0 in ALARM.
        set Bit 1 in ALARM.
ELSE
    IF the physical node does not tick TEMP_TIME, then
        TEMP_TIME = TEMP_TIME + physical node cycle time
    END IF.
END IF.
IF the communications layer can provide NODE_TIME, then
    TEMP_TIME = NODE_TIME

```

Time Parameters:

- The actual parameters in the physical node data base will be:
- NODE_TIME
 - + required
 - + reports time since the Field Device started.
 - + 48 bit unsigned integer.
 - + time in 1/32 milliseconds since device start-up till present.
 - + defaults to zero.
 - + stored in the physical node's dynamic data base.
 - + read-only from Field Bus.
 - PRIVATE_OFFSET
 - + required if private clock supported
 - + reports (private time - Field Bus time).
 - + 32 bit signed integer.
 - + defaults to minus full scale.
 - + read-only from Field Bus.
 - OFFSET_DB
 - + required if physical node supports alerts.
 - + dead band for reporting changes in the NODE_TIME_OFFSET (and the private clock if there is one).
 - + 16 bit unsigned integer.
 - + differential time expressed in one-quarter of a millisecond.
 - + defaults to 3/4 millisecond.
 - + stored in the physical node's static data base.
 - + read/write from Field Bus.
 - ALARM_TIME
 - + optional (existence of option given in the "options bit string".
 - + if present:
 - @ must be readable and writeable from Field Bus
 - @ a constant stored by the physical node of the field device.
 - @ 48 bit unsigned integer.
 - @ is considered inactive when set to exactly zero.
 - @ is stored in the physical node's static data base.
 - ALARM
 - + required if the ALARM_TIME option is present.
 - + an 8-bit binary string - only 2 bits defined.
 - + stored in the physical node's dynamic data base and reset at device restart (may be immediately set as a result of the above logic calculation).
 - + Bit 0 is controlled by the field device every physical node cycle as follows:
 - @ reset at physical node power-up unless the dynamic memory is known to have survived the outage.
 - @ IF ALARM_TIME <> 0, then
 - IF NODE_TIME and NODE_TIME_OFFSET available and neither are = 0, then
 - IF (NODE_TIME + NODE_TIME_OFFSET) > ALARM_TIME),

```

        then
            Set Bit 0 of ALARM
        ELSE Reset Bit 0 of ALARM
    ELSE IF TEMP_TIME and TEMP_BIAS available and neither
        are = 0, then
        IF (TEMP_TIME + TEMP_BIAS) > ALARM_TIME, then
            Set TIME_EVENT
        ELSE Reset Bit 0 of ALARM
+ bit 0 is forced to be read-only from Field Bus by the "Data Base
  Write Service".

```

PHYSICAL NODE OPTIONS BIT STRING:

Every physical node connected to Field Bus will have a "time critical" options bit string, "TIME_CRIT0". Standard physical nodes will use the following definition for the 16 bit TIME_CRIT0:

- Bit 0 = set if this physical node is part of a complex device (see above).
- Bit 1 = set if this physical node supports a User Layer logic engine that can run independent of any communication engines (see above).
- Bit 2 = set if this physical node supports restart timers and Fail-safe (see the paper ["Field Device Start and Restart"](#)).
- Bit 3 = set if an "alarm clock" is supported.
- Bit 4 = set if "alarm clock retention" is supported (see above).
- Bit 5 = set if the physical node supports a clipboard (see the paper ["Human I/F Considerations"](#) for a description of this option).
- Bit 6 = set if the physical node supports an associated clipboard (see the paper ["Human I/F Considerations"](#) for a description of this option).
- Bit 7 = set if primary clipboard "coded" (i.e., all material of construction information in coded form) (see ["Human I/F Considerations"](#) paper for a description of this option).
- Bit 8 = set if primary clipboard "complete" (i.e., all parts that normally contact the "process fluid/material" have materials of construction identified) (see ["Human I/F Considerations"](#) paper for a description of this option).
- Bit 9 = set if associated clipboard "coded" (i.e., all material of construction information in coded form) (see ["Human I/F Considerations"](#) paper for a description of this option).
- Bit AH = set if associated clipboard "complete" (i.e., all parts that normally contact the "process fluid/material" have materials of construction identified) (see ["Human I/F Considerations"](#) paper for a description of this option).
- Bit BH = set if this physical node supports direct writes to output hardware when the block processor is stopped (only defined for DR, PO, and SO type hardware).
- Bits CH - EH = reserved for future standards.
- Bit FH = set if this physical node supports any extended parameters. If this bit is set, parameter EXP0 must exist in the Data Dictionary. See Attachment 2 to the paper ["Application Support Services Interface"](#).

MISCELLANEOUS PHYSICAL NODE VARIABLES:

There are several additional parameters that are required in the physical node's data base for Standard physical nodes plus other parameter names that are defined for possible use beyond the Standard. Some of the parameters are conveniently defined in subsequent papers of the Technical Report. Others will be defined here; all of them are combined in the "Array of Basic Parameters" paper.