THE

FOUNDATION™ fieldbus

PRIMER

Prepared

by

fieldbus inc.

Revision 1.1
Released June 24, 2001

Prolog

Fieldbus Inc. has been involved with FOUNDATION™ fieldbus and its precedent technologies since 1993. Since 1996 we have worked with clients worldwide in developing, and registering as interoperable, FOUNDATION™ fieldbus devices. Fieldbus Inc. is a total solution supplier, capable of providing all required software, hardware, tools, engineering and project management to assure fast, economical development programs.

As a result of years of experience, and discussions with many fine clients, we have prepared this introductory manual to help those new to the field gain a basic understanding of fieldbus. We sincerely hope our readers find this primer useful, and we wish you success in mastering this revolutionary new technology.

Fieldbus Inc.
9390 Research Boulevard
Suite I-350
Austin, Texas 78759

Tel: 512 794-1011
Fax: 512 794-3904
www.fieldbusinc.com

CONTENTS

THE FOUNDATION™ fieldbus PRIMER

PREFACE

This primer is written for plant operators, engineers and managers who have at least a working understanding of process control and plant operations, and are looking for an introduction to the basic principles of FOUNDATION™ fieldbus. The focus will be on what the technology does and how such fieldbus systems behave, using examples to illustrate overall concepts.

Introduction

There are many digital communication technologies being promoted as the future replacement for the venerable 4–20 mA analog standard, and most are self-described as *fieldbus*. With the exception of FOUNDATION™ fieldbus, virtually all of these technologies were developed for non-process environments such as automotive manufacturing, building automation, or discrete parts manufacturing, and later adapted to process control. Generally, they are well suited to the applications for which they were originally developed. Some of these technologies are open, some are proprietary. Most are a combination, having some open aspects but requiring the use of proprietary hardware or software components. All but FOUNDATION require a proprietary application to provide a complete control solution.

Every communication technology provides a method for transmitting data between various devices and a host, and some provide communications directly between devices. The various schemes differ in how well they are optimized for moving data quickly, their suitability for real-time control, the cost of hardware implementations, their networking capability for branches, spurs and long distances, and for how power is distributed.

Comparisons among "fieldbus technologies" typically reduces to comparisons of data rates, message length, number of devices on a segment, etc. These are all important communications issues and each technology represents a particular set of trade-offs which adapt it to its original application, and each is rooted in the technology that was available or in vogue at the time of its development.

Using a strategy exactly opposite of FOUNDATION fieldbus, these various communications technologies *minimize* dependence on local intelligence in deference to minimum device cost, and maximize reliance on a centralized control architecture. Measurement instruments in such structures communicate to a central computing system at the request of that central system. A *proprietary control application*, running on the central system processes the field data and distributes control signals to other devices back in the field. Regardless of how open the communication scheme may be, the control application is *always* proprietary.

The key distinctions between these technologies and FOUNDATION fieldbus are;

- FOUNDATION fieldbus provides an open specification for both communications *and* the control application.
- FOUNDATION fieldbus distributes control functionality across the bus, making maximum use of local intelligence to improve performance and reduce total system cost.
- Devices are required to be interoperable, providing the user with tools to implement a control system with products from multiple manufacturers without custom programming.

With FOUNDATION fieldbus, *the network is the control system*. The major goals of this primer are to illustrate how this new architecture behaves, what it means to users, and what it requires of control equipment manufacturers.

Background

The technology of Fieldbus[1] began to evolve in August of 1984 when the International Electrotechnical Commission (IEC) and The International Society for Measurement and Control (ISA)[2] met to initiate work on a new international standard. Their objectives and requirements were presented in a Draft Report issued three years later,[3] in September of 1987. The report presented functional guidelines which would be refined, tested and committed to a specification over the following decade.

The primary stated objective of the '87 report was, in part, "…to specify a digital, serial, communications link between primary automation devices deployed in a manufacturing/process area (the field) and higher-level automation/control devices located in a production control area (the control room)". The committees producing the draft were staffed about equally with end users and suppliers. The report dealt predominantly with physical layer issues and [control] application requirements, the two features most obvious to users.

Two areas of [physical] application were identified and labeled as H1 and H2. The H1 application area was intended as a digital replacement for the 4 to 20 mA analog standard in widespread use in the fluid process industries, later called *process automation*. The H2 application area was intended to extend the H1 concept to systems having a different set of requirements including high speed discrete control and data collection applications, later called *manufacturing automation*. The objective was to develop one standard with the flexibility to satisfy the needs of both application areas, though with different physical layers and data rates.

The H1 requirements included the need to power field devices from a single unshielded, twisted wire pair, which would also be used for communication, and to be able to meet low power, intrinsically safe standards already in use. Other major requirements included bus lengths up to 1,900 meters (1.18 miles) without repeaters, unlimited spurs, and operation in electrically noisy environments. It was demonstrated that a Manchester encoded signal at 31.25 kbits per second fulfilled all requirements.

Contrary to the usual view, the H2 application requirements were in some ways less demanding. For example, separate wires for power and signal were acceptable, the required distances were as little as 500 meters (0.31 miles), complex spur topographies were unnecessary, and higher quality, Ethernet cabling was the accepted practice. On the other hand, the data rates agreed to for the H2 were 1 and 2-1/2 Mbits per sec, significantly higher than for H1. A commonly accepted model for process automation was that H2 would serve as a *home run* connection with numerous H1 segments attached along the way. The expectation was that manufacturing automation would rely exclusively on H2.

The final specification[4] for H1 was published in August of 1996. Subsequent development of products and the application of H1 networks in process control is accelerating as anticipated.

Testing of the H2 specification was suspended in March of 1998 when the Fieldbus Foundation shifted its goal for the high speed network to 100 Mbit/sec Ethernet, also called high speed Ethernet (HSE). The H1/H2 architectural concepts remain the same, with H2 being displaced by HSE. The primary advantages of HSE are:

- Higher speed
- Low cost, off-the-shelf hardware
- Smoother integration with management information systems

The HSE final specification, FF-044-1.0-1, was published in April 2000, including everything but redundancy. The preliminary specification for redundancy has been released, but requires validation, which is currently scheduled to occur in three phases.

[1] We shall use Fieldbus with a capital "F" as shorthand for FOUNDATION™ fieldbus.
[2] In 1994 the Instrument Society of America changed its name to, *The International Society for Measurement and Control*, but retained use of the abbreviation, *ISA*.
[3] Instrument Society of America Standards and Practices 50 Functional Guidelines, Sept 9, 1987, ISA-SP50-1987-17-G. See Appendix I.

[4] Part 1, FF 890 and Part 2, FF-891

The focus of the IEC/ISA committees was clearly on how to solve application requirements. What the physical layer needed to do and the environment in which it had to perform was spelled out in some detail. The expectations of the control and monitoring systems were identified. Requirements such as the need to "hot wire" devices on an operating bus segment and the need for prioritized access privileges to device parameters on the network were specifically cited.

necessary to understand the ISO model to understand Fieldbus, but the comparison may be useful to some readers.

The OSI model represents all communication functions as a set of seven layers, as shown on the left side of the figure. It does not address any application that may need to use communications, shown here as it often is as "Layer 8".

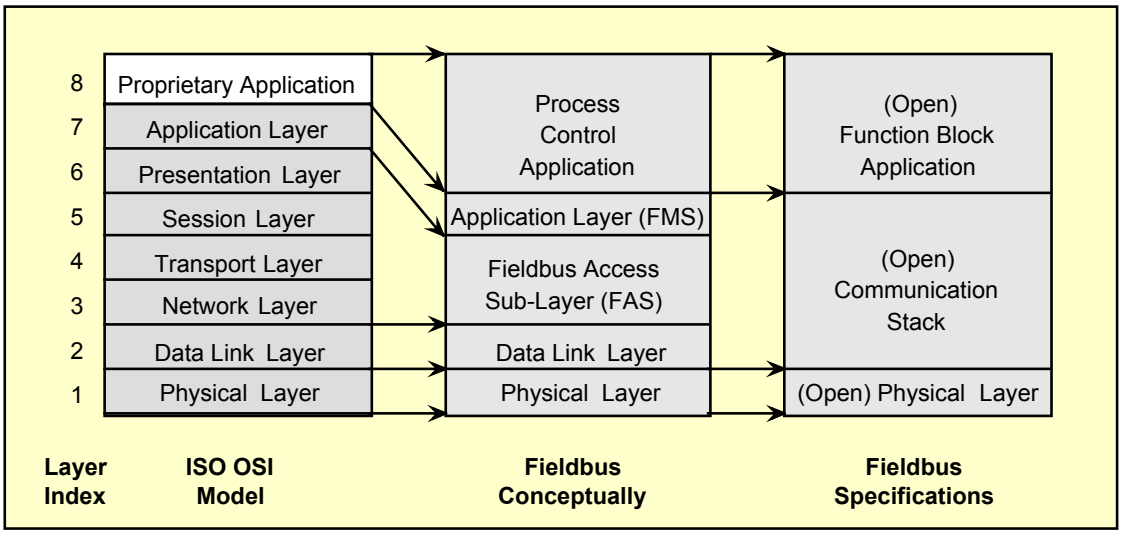

Figure 1. Relationship Between Fieldbus and OSI Model.

Furthermore, the customary ability in the analog world of being able to replace a device from one manufacturer with a similar device from another manufacturer without loss of functionality, and without requiring special engineering, was an expectation of Fieldbus. Why should a newer technology offer less capability than the older one it was displacing? Multi-manufacturer interoperability was a natural expectation.

Specifying a complete, well designed control application, the so-called *user layer,* held equal priority with specifying the communication scheme, whose only real purpose was to support the control application anyway.

The Fieldbus specifications that emerged can be related to the International Standards Organization (ISO) Open System Interconnect (OSI) model, as shown in Figure 1. It is not

Fieldbus is essentially a local area network. Therefore, much of layers 3 through 6 are not required. Those elements that are needed, were compressed into what is called the fieldbus access sublayer, plus some additional entities such as system management and network management. As the diagram shows, this, combined with the application and data link layers, forms the communications aspect of Fieldbus.

The layer not defined by OSI, and sometimes also called the *user layer*, is defined in Fieldbus as the Function Block Application.

The entire functionality represented by these three segments; physical layer, communication stack, and function block application, must reside in every active device on a Fieldbus network. In addition, in field devices, the tightly specified

function block application must interface with the unique technology of the device. This is achieved through *transducer blocks*. A schematic of a simple network illustrates these concepts in Figure 2, below.

A synopsis of the physical layer, communications stack, and function block application follows.

Physical Layer

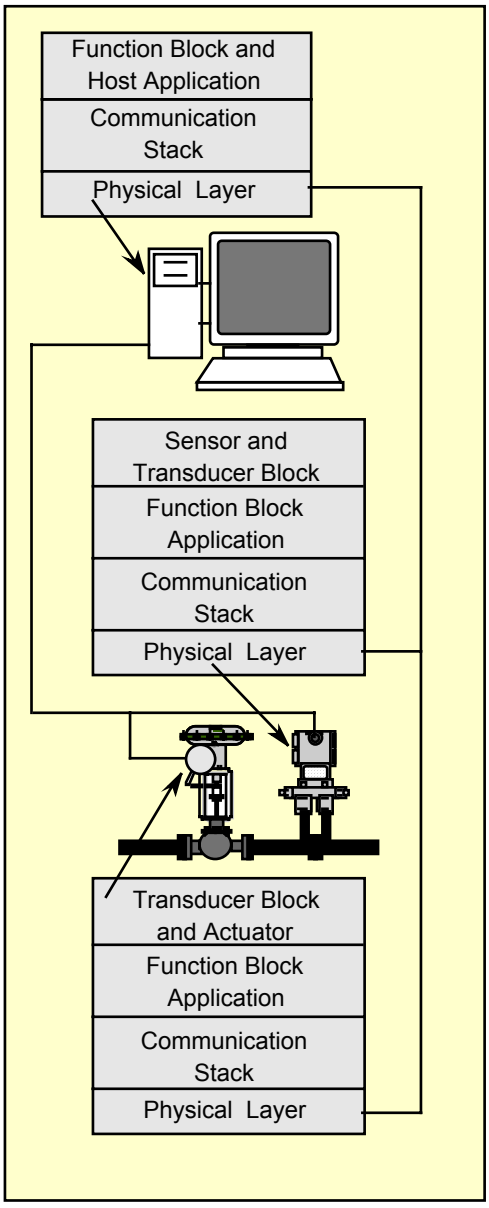

Figure 2. System Schematic.

Early Fieldbus committee work resulted in international approval of a physical layer standard in 1992. The specification is available either as IEC 1158-2: 1993, or as ISA-S50.02-1992. Only the H1 physical layer will be reviewed here. The Fieldbus Foundation publishes its FF-816, *FOUNDATION™ Specification 31.25 kbits Physical Layer Profile*, which defines the requirements for H1 busses. Also published is FF-830, *31.25 kbits Physical Layer Conformance Test Specification*, which defines the physical layer requirements that devices must pass to be registered as interoperable. These documents are all of importance to device developers.

The Fieldbus Foundation publishes two Application Guides which are relevant to the physical layer; *Wiring and Installation 31.25 kbits, Voltage Mode, Wire Medium*, and *31.25 kbits Intrinsically Safe Systems.* These should be of interest to developers and end users alike. The physical layer specifications define the mechanical and electrical requirements for transmitting signals between Fieldbus devices.

The relatively low H1 data rate of 31.25 kbits was intentionally selected to minimize constraints on the cabling system.[5] The result is that existing twisted wire pairs used in 4-20 mA installations can, in many cases, be used for Fieldbus signals. Higher quality, lower capacitance wiring is specified if maximum distances are required.

H1 allows for fairly liberal topologies, as illustrated In Figure 3. Spurs are allowed and the branch points may occur anywhere along the trunk, without restriction.

The number of devices possible on a single segment depends on factors such as the power consumption of each device, the type of cable used, whether the devices must be intrinsically safe (IS), and whether they are bus powered.

[5] 31.25 kbits is also easily obtained by dividing down from a 1 MHz oscillator.

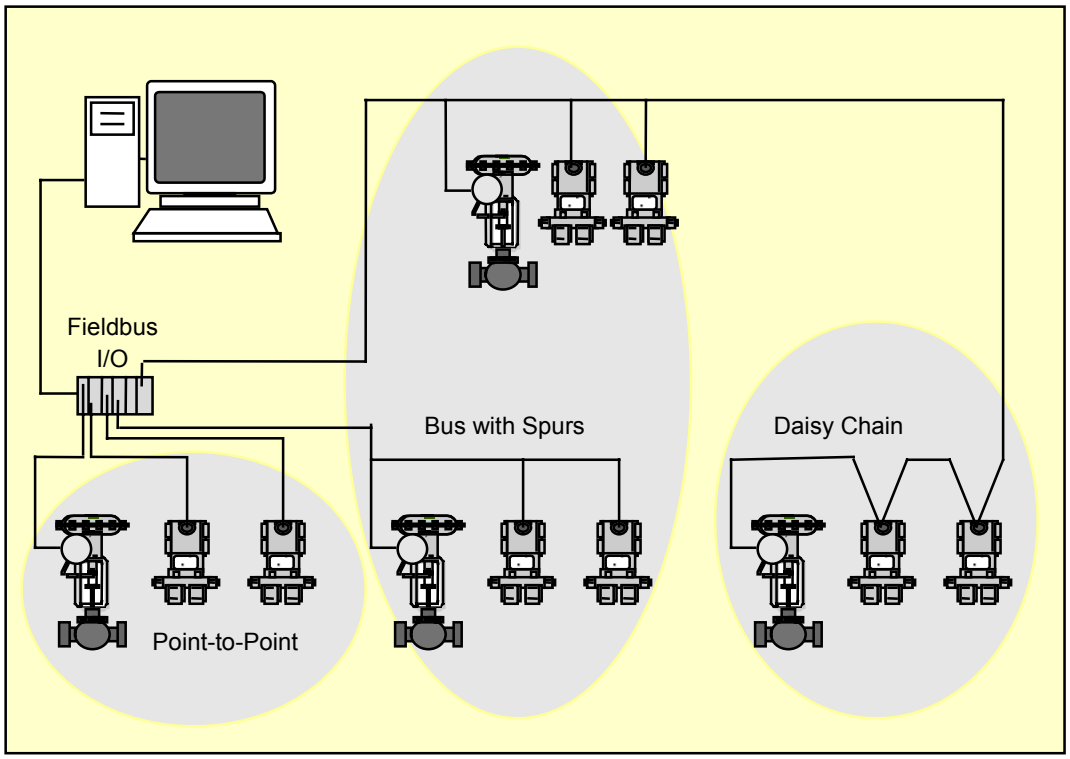

Figure 3. Example H1 Fieldbus Technology.

The maximum number of devices is often listed as six if bus powered and IS, 12 if non-IS, and 32 if neither bus powered nor IS. These limits are based on assumed power requirements and supplies, but are not specified.

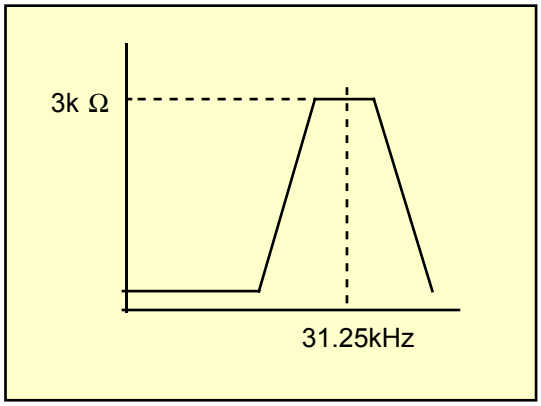

Figure 4. Power Supply Source Impedance.

The total cable length including spurs is 1900 meters (6234 ft). Up to four repeaters are permitted, which extends the cable length to nearly five miles. The network must also have a terminator at each of the two most distant ends.

A Fieldbus network requires a special fieldbus power supply. The standard voltage level is the familiar 24 volts[6] dc. A current capability of 100 mA to 120 mA is appropriate for a typical H1 bus segment, and can meet IS requirements.

A unique requirement of the power supply is that it must offer low source impedance at dc, but high impedance at the signaling frequency, as illustrated in Figure 4.

The signal is Manchester encoded and an idealized signal wave form is illustrated in Figure 5. Some of the characteristics of Manchester encoding are that the signal frequency is used to synchronize the data clocks between communicating devices. Also, a logical zero or one is indicated by whether the signal is falling or rising at the center of a "bit time".

A bus powered Fieldbus device will typically draw between 10 mA and 20 mA of current for steady-state operation. During signaling, it will "toggle" the current draw above and below the

[6] The specification requires devices to operate on voltages from 9 VDC to 32 VDC.

steady state value by 0.25 to 0.33 mA. Against the 3000 ohm source impedance of the power supply, this results in a 0.75 to 1.0 peak-to-peak voltage signal on the bus.

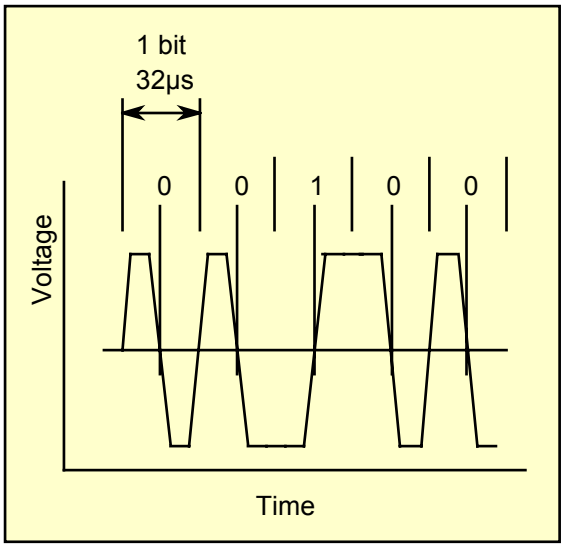

Figure 5. Ideal Signal Wave Form.

Low source impedance of the power supply at dc and low frequencies, allows devices to be added or removed from the bus with a minimal affect on steady state bus voltage. Fieldbus devices are required to operate on voltage levels from 9 V to 32 V.

The relatively low frequency, high signal levels, and low dc impedance, make possible the use of low-cost twisted wire pairs for power distribution and reliable communications. The suitability of the technology for in-plant use was tested prior to the acceptance of the physical layer standard. The primary test beds were an Exxon refinery in Linden, New Jersey and a BP refinery in Sudbury, England. For additional information on the Fieldbus physical layer, refer to the references listed in the beginning of this section.

Communications Capability

The requirements of the communications scheme for multiple devices on a common cable or network depend on the system requirements that the network must satisfy. Process automation presents a variety of sometimes contradictory requirements.

For example, data required for real time control must be transmitted on a precisely periodic time schedule to provide satisfactory dynamic control of plant processes. There is no merit in confirming a transmission since, in a dynamic system, a data point only has relevance at a particular instant in time. Re-transmitting an out of date value serves no purpose. Devices performing control must rely on valid periodic data, yet be able to cope with an occasional missing or faulty value.

In contrast, the occurrence of alarm conditions should be communicated at the earliest possibility, independent of periodic time intervals. Furthermore, if an alarm message is corrupted or lost, it is essential that the same message be re-transmitted, and methods of confirmation and acknowledgement are required.

As illustrated in Figure 6, there is also the question of who should communicate with whom? Some messages must be sent to multiple

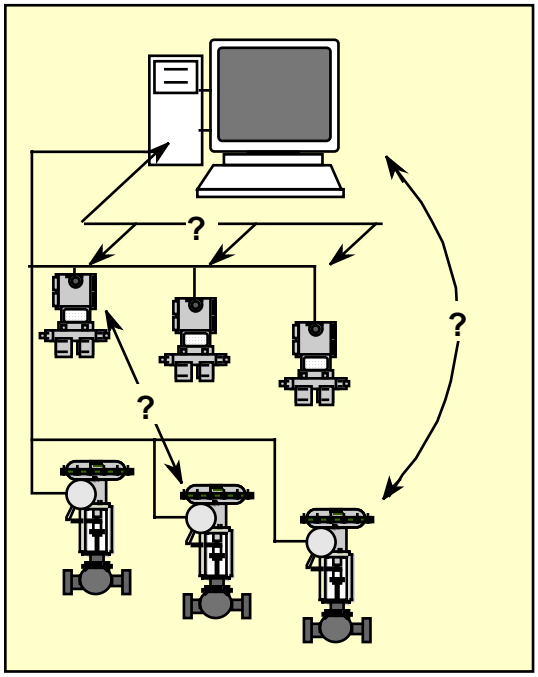

Figure 6. Communication Paths.

receivers, or broadcast. In other cases, only two devices may be involved, peer-to-peer.

In Fieldbus, the device that is located in the control room is typically called a *host*. It needs capabilities that are usually not required of other devices on the network. These include the ability to configure and set-up a network, and may include the ability to control communications.

All communications in Fieldbus are controlled by a single device called the *link master*. The link master contains a function called a *Link Active Scheduler*, or LAS. Most host devices will have an LAS. How the LAS controls network communications is discussed in later sections.

There are several classes of field devices including; basic, link master, linking device and gateway. A basic device is capable of all the basic communications required of a field device. The relationship between a host and a basic device is that the host is a *client*, the basic device is a *server*. A client makes requests/demands, a server obliges.

If LAS capability is added to a basic device, it becomes a link master (LM) device. This does not give it all the capabilities of a host, but it does permit it to control communications. Fieldbus permits multiple, redundant link masters. The LAS device having the lowest bus address is automatically in control.

A Linking Device (LD) is used to connect an H1 segment to an HSE segment. More likely, an LD will be able to connect several H1 segments to HSE. The primary functions of the LD are to insert or remove Fieldbus messages into or from Ethernet messages, and to provide a routing function among the various segments and devices. This will be discussed further in a later section.

A gateway device is used to connect Fieldbus to some other, non-fieldbus communications protocol, such as Modbus. Gateways will not be discussed further.

The Fieldbus communications technology provides scheduled communications for precise, periodic transmission of data used in control, and non-scheduled communications for "earliest opportunity" transmission of other data. Messages need not be routed through the host, but can pass directly from the device producing the data to the device that requires the data. Message confirmation may be required, but only where it is appropriate. In either case, messages contain an internal error checking mechanism so that an error will be detected by the receiving device.[7]

Behavior of the communication mechanisms will be more fully explained by examples in later sections. It is important to understand that the Fieldbus protocol was designed in cooperation with the control application, so the requirements of each are matched by design. The control application is discussed next.

Function Block Application

In process control the prevailing model for describing a control system is the function block diagram. Because of its relative ease of use and worldwide acceptance, this became the model for the Fieldbus user layer, which is a standardized application running on a standardized communications protocol.

Control systems require various interfaces to their environment, as illustrated in a most general sense in Figure 7. Inputs consist of various measured parameters such as; pressure, temperature, switch position, shaft speed, discrete array patterns, etc. Outputs may be; shaft positions or velocity, switch closures or other modulation variables generally used to control some of the inputs.

In the usual case, some type of human/machine interface (HMI) is required at some level of the

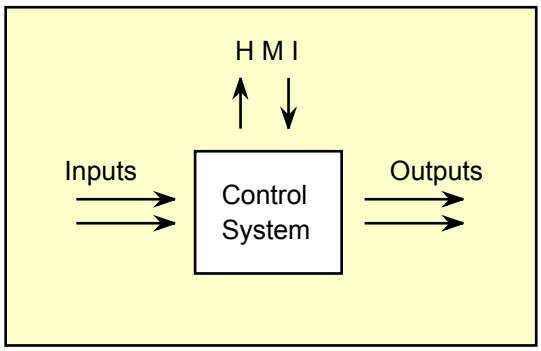

Figure 7. General Model of a Control System.

operation. The control system itself is modeled as an interconnected set of functions, or function blocks, as illustrated by the cascade control loop represented in Figure 8.

[7] A cyclic redundancy algorithm is used which yields an undetected error rate of less than one error in 21 years of continuous operation.

Fieldbus function blocks are software encapsulations designed so that each perform a specified set of operations on one or more inputs, and generate one or more outputs. Blocks reside in the various devices on a Fieldbus network to implement the control strategy. Blocks are evaluated, or executed, on a periodic basis under control of the (physical) device in which they reside. Communication among blocks takes place through linkages that are established by a configuration procedure, before startup or during operation. Custom programming or special file generation is not required to configure these linkages.

Some blocks, such as a PID controller, may be located almost anywhere on the network. Other blocks perform functions that are associated with

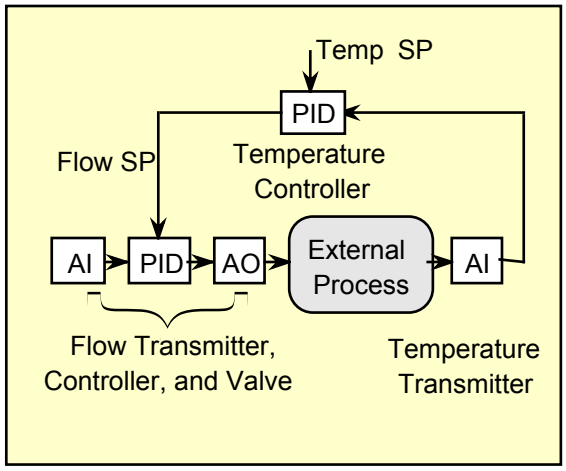

Figure 8. Function Block Diagram of Cascade Loop.

specific items of equipment.

For example, an *Analog In* (AI) block accepts a digitized representation of a particular analog parameter, the measured variable, and performs specified operations on the data. The primary operations include checking engineering units, filtering, optionally extracting square root, checking for alarm conditions and range checks. Every transmitter measuring an analog variable must have at least one AI block to bring the process value into the network. In a similar way, every control valve must have an *Analog Out* (AO) block to produce a digitized representation of the desired valve stem position, and to move the value out of the network (and into the valve positioner).

Other functions, such as a PID controller, may reside in any device on the network including the valve, the transmitter, or the host. At this time there are 21 function blocks in the final specification and 12 more are at the preliminary specification stage. These are defined by;

- Part 1, FF-890 Revision 1.4
- Part 2, FF-891 Revision 1.4
- Part 3, FF-892 Revision 1.4

The current revisions are final specifications (FS) for Parts 1, 2, and 3, and were released in June of 1999.

Table 1 lists the first ten blocks (Part 2) and briefly describes their primary purpose. Additions to the standard blocks are possible as users of the technology define new requirements. Custom function blocks are permissible with the use of specified files called *Device Descriptions,* which will be discussed in a later section.

HSE will support all H1 function blocks. In addition, an extended set of blocks specifically for HSE applications is in development.

Block Symbol	Primary Functions
AI	Analog In: Accepts digitized representation of an external analog signal from external hardware. Performs scaling, filtering, and provides high and low alarm function.
AO	Analog Out: Provides digitized representation of an analog signal to external hardware. Includes scaling, range and rate of change limits, and provides several fault-state (fail-safe) options.
DI	Discrete In: Accepts 8-bit unsigned input value from external hardware. Provides filtering, optional inversion, and discrete alarms.
DO	Discrete Out: Passes 8-bit unsigned output value to external hardware. Provides optional inversion, mode shedding and fault-state (fail-safe).
PID	Proportional, Integral, Derivative Controller: Provides filtering, set point limits and rate limits, feedforward support, output limits, error alarms, mode shedding, and anti-wind-up capability, but the PID algorithm is manufacturer specific.
PD	Proportional + Derivative Controller: Provides filtering, set point limits and rate limits, feedforward support, output limits, error alarms, mode shedding, and manual bias, but the PD algorithm is manufacturer specific.
ML	Manual Loader: Accepts digitized representation of an analog signal from an AI block or a computer. Provides high and low output limit function and an output to other blocks. Conceptually, a controller with a human for the control algorithm.
BG	Bias/Gain Station: A simple calculation block with output limits, that connect to other blocks.
CS	Control Selector: Selects among highest, lowest, or average of two or three inputs (from other blocks). Provides balanceless transfer of signals.
RA	Ratio Station; Accepts signals from two AI blocks (or other source) and computes, as its output, the correct set point to be used by a controller block for the purpose of controlling a ratio of the input parameters.

Table 1. Abbreviated Description of First Ten Fieldbus Function Blocks.

Fieldbus specifications call for a *resource block* in each device. The purpose of the resource block is to retain information about the device itself. Required information includes: manufacturer ID, type of device and revision level, memory size, free memory space, available computational time, declaration of available features, and the state of the device, i.e., initializing, on-line, standby, failure, etc. These data must be stored in non-volatile memory. A manufacturer may choose to add value by providing additional data such as materials of construction, calibration and repair records, and other useful information.

Though the user will have access to this information, some of it may only be used by configuration tools or other utilities residing on the host. However, the existence of this data in a standardized form provides a rich data base for supporting various plant maintenance, inventory, and other operational programs.

The function block behavior described in Table 1 indicates the purpose of each block, but there are a number of essential behaviors not easily described in a table. Examples in the following section will be used to better reveal these characteristics and their significance.

A Systems Perspective

To this point we have described Fieldbus as consisting of three primary components; the physical layer, a communications stack, and the function block application. Without an overall perspective, it becomes increasingly tedious to discuss further the many important details of these components. Fieldbus networks are control systems, so it is useful to look at Fieldbus in the context of a system. In this section we will discuss one continuous and one discrete control

application as a means of illustrating the most important capabilities. A later section will elaborate on the major, specific functions and features.

Continuous Control Process

A common process application, and one which illustrates many control system requirements, is a simple steam heater. The example we will use is a heat exchanger where steam is flowing through a coiled tube in an enclosed vessel. A process fluid is being pumped through the vessel and the objective is to heat this fluid to a precise temperature. This is illustrated in the equipment schematic in the lower half of Figure 9. In the upper half of Figure 9, a block diagram depicts the control strategy for the system.

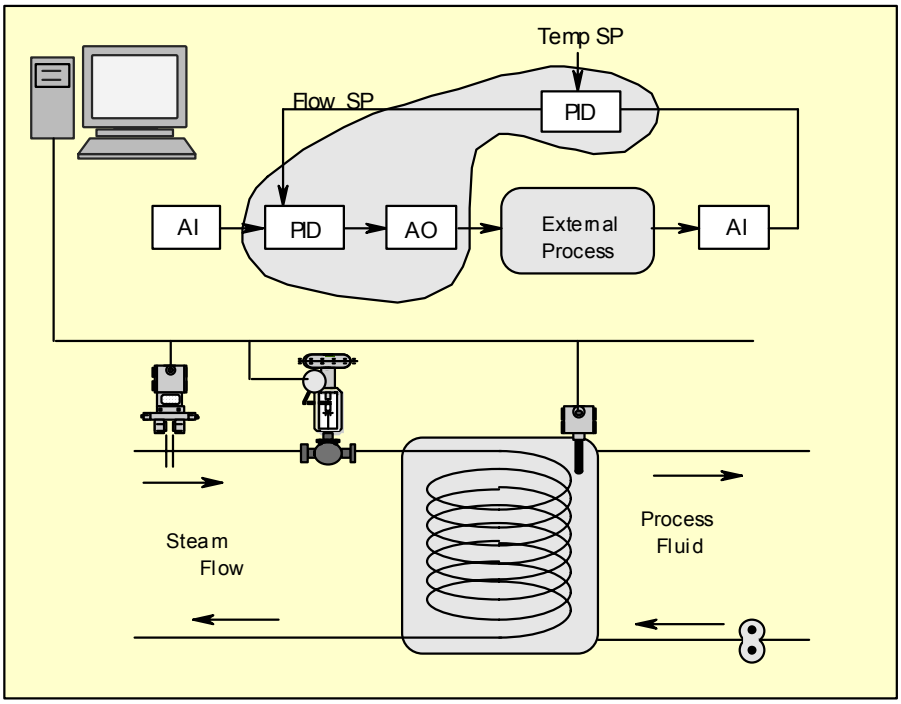

Figure 9. Schematic and Block Diagram of System.

Referring to the block diagram, the steam flow measurement is brought into the network by the Analog In (AI) block on the left. This value is linked to a PID block, which is the flow controller. The flow controller is receiving its Set Point (SP) from elsewhere (for now). The output of the flow PID is linked to an Analog Out (AO) block which will manipulate an external physical variable, in this case the valve.

The valve will be manipulated as necessary to adjust steam flow to make it agree with the flow SP. Heat liberated by the steam in the tube coil will raise the temperature of the process fluid. The fluid temperature measurement is brought into the network by the AI block on the right of the diagram. This value is linked to another PID block, which is the temperature controller. The temperature PID compares the measured process temperature with the temperature SP and adjusts steam flow to achieve or hold the process temperature at this SP. The output of the temperature PID is used as the flow SP and is linked to the flow PID.

To summarize, the purpose of the flow control loop is to maintain steam flow at a specified value regardless of variations in steam pressure. The purpose of the temperature control loop is to adjust the steam loop SP as needed to maintain the specified process temperature regardless of variations in process flow or process inlet temperature.

This is a well known cascade control strategy used frequently in process control. The equipment schematic illustrates the major hardware components of the system and also shows a host console on the far left. This console is located where a human operator can observe the behavior of the system and make input commands such as setting the desired process temperature and controller tuning parameters. The console may be several hundred feet from the process and is typically out of the plant environment. The schematic also shows the flow and temperature transmitters, valve, and console, all connected on a single bus.

In traditional centralized systems, the flow and temperature measurements would be transmitted to the host where the control strategy is executed. The desired valve corrections are then transmitted back to the field. This is a FOUNDATION fieldbus installation so the control strategy shown in the block diagram will be distributed across the network. In Fieldbus*, the network is the control system*. The functions of the AI and AO blocks must reside in the transmitters and valve, because this is where the network meets the external world. We are free to locate the remaining functions where it is most advantageous.

The AI block on the left is located in the flow transmitter, the second AI block is located in the temperature transmitter. The AO block is located in the valve. All other blocks required to achieve the control strategy may be located anywhere on the network; in the host, the transmitters, the valve, or an ancillary device not shown. In this example both PID blocks will be located in the valve, we shall see why in a moment.

With Fieldbus, all devices on a network have a common sense of time. Execution of function blocks takes place on a precise schedule so that the blocks operate in a planned sequence, and data is transmitted just in time for use. This synchronization eliminates the need for anti-aliasing filters between function blocks, which would otherwise reduce bandwidth by orders of magnitude (or, if filters are not employed, would result in serious aliasing problems).

As mentioned earlier, communication between devices on the bus is controlled by one device on the network through a mechanism called the *Link Active Scheduler*, or *LAS*. The device which contains the LAS is called the *Link Master*. Usually this is the host device. The Fieldbus specification also provides for redundant Link Masters. In an application such as the example, it would not be unreasonable to include a redundant LAS in the valve, or in a field mounted auxiliary device. This would permit continued operation of the control system in the event that the host failed or became temporarily disconnected. During such an event, the only loss would be the operator's view of the process.

To more completely describe interaction between communications and function block execution, a timing diagram is presented in Figure 10. A list of the function blocks is shown on the vertical left axis. The horizontal axis is time. Execution of the function blocks is illustrated by a small rectangular icon. The thick, dotted vertical lines represent instances when scheduled data are being transmitted on the bus.

To explain the sequence we will start with the execution of the AI block in the flow transmitter. The LAS has the function block execution schedule and causes the flow data to be *published* on the bus just after fresh data becomes available. Any device needing the flow data will have been configured to *subscribe* as the data appears. In this case, the host may or may not subscribe for various reasons. The valve will need to subscribe because the flow PID requires the information.

After the flow data is published, the function block schedule (which resides in the device) will call for the flow PID to execute. The result of the PID calculation is passed immediately to the AO block, which is also in the valve, and at that time the AO block will be scheduled to execute. Execution of the AO block will result in an update of the signal to the valve Positioner (servo), outside the bus.

For this example, the entire flow control sequence is repeated every 200 milliseconds, or five times per second. Each execution of the flow control loop requires only one transmission of data on the bus, and no deadtime is accumulated while data is passed from buffer to buffer in some remote, centralized host system.

Execution of the temperature AI block is shown on the third row of the diagram. In this example, it has been scheduled to execute prior to the flow AI block. The temperature process value (PV) has been scheduled to publish at about the same time as the flow AI block executes. Since block execution occurs entirely within a device, there is no conflict with these events occurring simultaneously. As before, devices that are users of the data will have been configured to subscribe. Again in this example it is the valve, or more precisely, the PID block in the valve positioner.

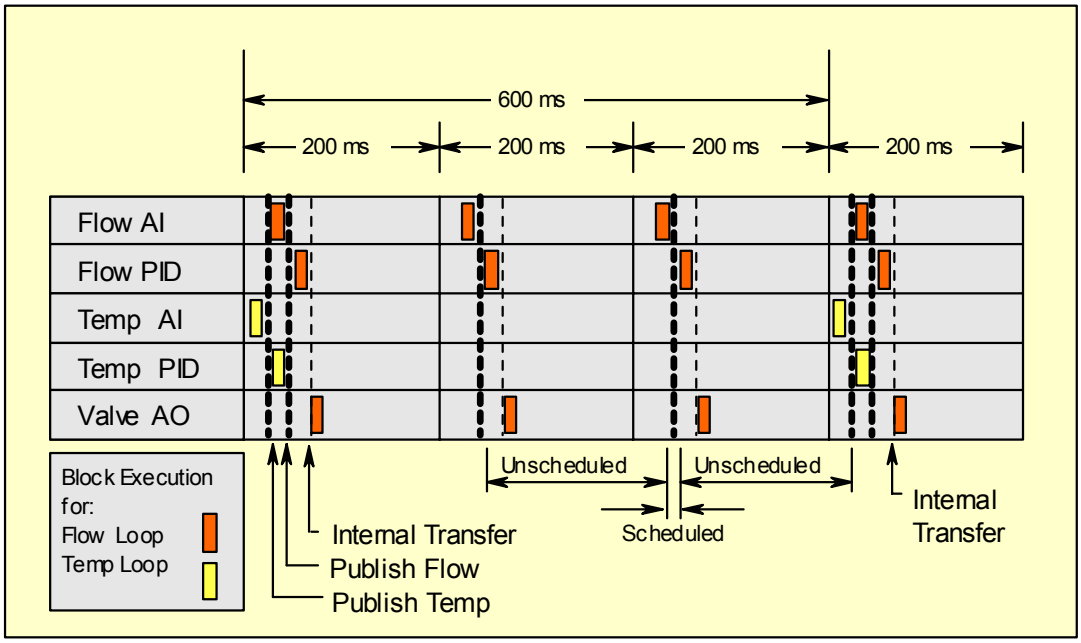

Figure 10. Timing Diagram for Cascade Example.

The temperature PID will be scheduled to execute and will then immediately pass its output value to the flow PID to serve as its new flow SP. As shown, this occurs just prior to the flow data being published. The flow PID will thus have the most current SP information possible.

Since thermal lags in the system make temperature a more slowly changing variable, we have elected to execute the temperature sequence only every 600 milliseconds, or every 1.67 second. This is the slowest scheduled sequence on this bus segment, so this is called the *macrocycle*. The schedules in this example are illustrative only. Fieldbus function blocks and publisher/subscriber communications are configured to meet the dynamic requirements of the application.

The scheduled communications shown in Figure 10 occur at specific times in the macrocycle. In this example, only a few percent of the total available bus bandwidth is scheduled. In heavily loaded configurations, as much as 20 percent of the bus time may be scheduled for control communications. The remaining, unscheduled, time is used for network maintenance and other non-control requirements. Examples are; operator commands such as set point changes, checks for new or missing devices, and reports for alarms, which will be discussed in next section.

a) Temperature Device Failure

Earlier we mentioned that a redundant LAS could keep the system in operation in the event some conditions associated with field device failures.

Suppose that the thermocouple in this transmitter failed. Ignoring the fact that this is a smart instrument and could have redundant sensors, let us assume that it has a singe temperature probe. Obviously the transmitter can no longer publish correct process temperature values.

Parameters that are passed between function blocks are called *linked* parameters. Every linked parameter in Fieldbus contains a *value*, in this case temperature, and a *status*. Status indicates the quality of the value, which is either GOOD, UNCERTAIN, or BAD. Status also contains sub-status information giving added detail as to the general status condition.

In this example, the value picked up by the temperature controller will carry a status of BAD, sub-status: Sensor Failure. Local intelligence will cause the temperature controller to transition to Manual (MAN) mode. The output will remain at its last value and will be passed to the flow controller. Normally, the steam loop could then be controlled by manual adjustments to the temperature controller. Optionally, the BAD status indication can be propagated to the Flow controller causing it's mode to go from Cascade (CAS) to Automatic (AUTO). It will then continue to hold steam flow and will become receptive to operator applied Set Point changes.

Meanwhile, the AI block in the temperature transmitter will generate an *Event Notification* as a result of a *Block Alarm*. The Event Notification is a message which contains the device tag number, the fact that there was a sensor failure, time-stamped to within 1/32 of a millisecond as to when the failure was detected, and other relevant information. This message will be broadcast at the earliest opportunity short of interfering with the transmission of data used for control. The message will be periodically re-sent until receipt is confirmed by the appropriate authority. The same mechanism is used for process alarms and other "events".

These actions all occur under the control of local intelligence, without the need for higher levels of intervention. The host does not poll the device for alarm information. The message preparation and control mode changes are all done locally, as defined in the Foundation Fieldbus Specification.

We have, in this example, illustrated some aspects of the power of distributed, parallel processors performing local functions and passing high level information to the central system.

b) Flow Device Failure

As a brief variation of the example, assume that it is the flow device that fails. In this case the flow controller will get a flow value with the status of BAD; sub status, Sensor Failure. Its mode will immediately go from Cascade (CAS) to Manual (MAN).

Information of this mode change is immediately propagated back to the temperature controller which will then transition to the mode of IMAN. This is a manual mode in which the controller is continually looking downstream and initializing itself to be ready for resuming control when conditions warrant.

What is different from the previous example is that the flow controller now does not have the information needed to control flow. It will remain in MAN and allow (only) manual control from an operator console. The temperature controller, being in IMAN, will keep its output aligned with the flow SP as part of its initializing procedure. When the status of the flow value again becomes Good, the system will resume control without bumps or discontinuities.

As an option, The BAD status may be propagated forward to the AO block in the valve. If this option is selected, the AO block will hold the last value for a pre-configured period of time and then initiate fault-state (similar to fail-safe). Depending on the user's selection, it will move the valve open, closed, or hold last position. All failures automatically generate and transmit a time-stamped Event Notification, as was done previously in the temperature transmitter example.

The example chosen for this discussion is relatively simple. Alternative designs for this system could have employed transmitters with redundant sensors, and/or redundant transmitters, calculation functions, feedforward, model predictive and multi-variable controls and even redundant controllers and valves. The same principles would apply.

Two key points should be made regarding the preceding discussion. In more complex control strategies involving; ratio controls, override with control selectors,

- The behavior described is completely defined in the Fieldbus User Layer Application, which is an open specification. No other digital bus solution has an open, defined, User Layer specification.

- The behavior described is mandatory and can be trusted to exist whether the devices and host are supplied by one manufacturer or several. Every Foundation registered device must support the behavior described, and must interoperate with any other registered device, and must have passed an automated (interoperability) test to prove that it will.

The preceding examples do not provide a comprehensive explanation of all specified behavior, but are intended to illustrate the general level of the specifications.

c) Maintenance Concepts

We will continue the failed transmitter example and discuss the maintenance activities that might follow such an occurrence. Using the temperature transmitter as an example; the device will be removed from the process and transferred to a maintenance area, and a replacement will be supplied from inventory.

In the maintenance shop, the problem transmitter will be connected to a test bus segment (separate from the process segment) and diagnosed.

Disposition of the instrument may be;

- Return to the factory
- Dispose of the instrument
- Repair device and move to inventory

If it is repaired and returned to inventory, or if a replacement device is purchased for inventory, the device may be momentarily or continuously connected to another bus segment, separate from the process and maintenance segments.

The three host computers associated with the process, maintenance and inventory fieldbus segments are connected on a plant *backbone,* shown as an HSE segment in Figure 11. It is now possible for an *enterprise management* computer (not shown), to continuously track every device in the plant, and to determine the current condition or usage of each device. Also, if maintenance is performed on the failed transmitter, that information may be stored in the device and remain with it, but will be available over the bus to a plant-wide maintenance program.

The resource block in the User layer application of every Fieldbus device contains a manufacturer ID, device type, device revision, DD revision and special features.

By specification, this data must be in a format which is accessible and readable by any Fieldbus conformant interface device. The same maintenance and inventory tracking system is thus equally applicable, without custom programming, to all Fieldbus devices independent of manufacturer.

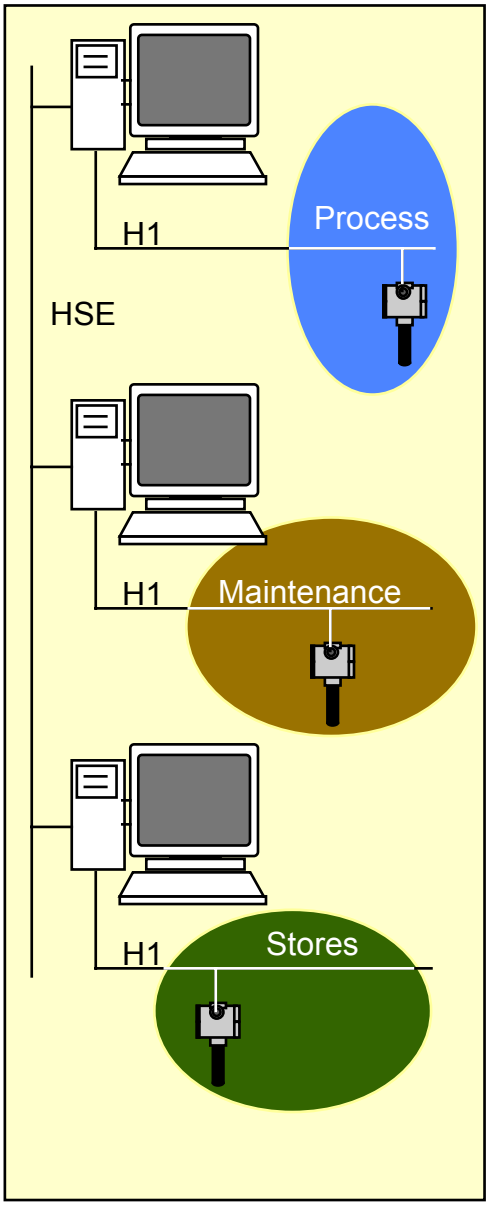

Figure 11. Device Tracking.

Discrete Control Example

This next example is presented not only to illustrate a discrete application, but to introduce several additional features of Fieldbus. The application is a container filling operation where bottles on a conveyor are sequentially parked at a filling tube, a valve is opened and closed to fill the bottle, and then the conveyor is incremented forward to position the next bottle for filling. Figure 12 schematically illustrates the process.

Numerical values will be used to better illustrate the nature of the problem, and the power of the solution. We will assume that the bottles are to be filled with 64 ounces of fluid and that the fill rate is nominally 32 oz/sec. Obviously, the average fill time is approximately two seconds plus the time required to increment the conveyor.

The control strategy is implemented in the three blocks shown in Figure 13. The flow meter has a totalizing feature and uses a standard Analog In (AI) block. The on/off valve uses a standard Discrete Out (DO) block to operate the valve,

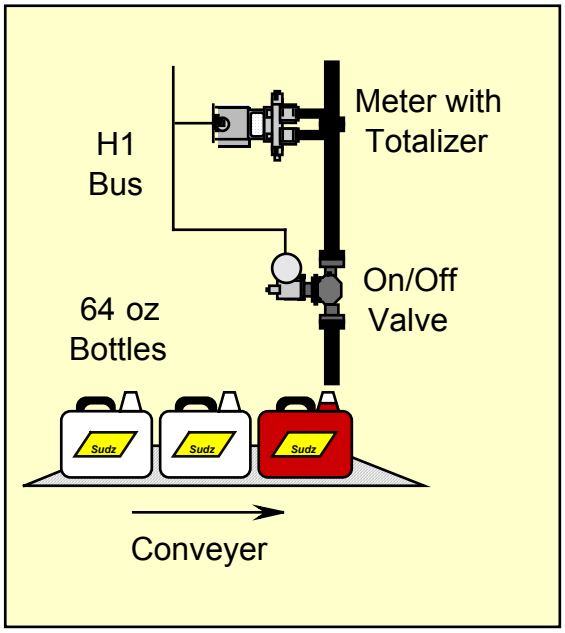

Figure 12. Discrete Process.

and a unique Discrete Control (DC) block to provide control.

Control of continuous processes depends on continuously changing control actions which are, in turn, based on precisely periodic calculations. These are time dependant control applications. The synchronized, periodic communications under the direction of the LAS were designed for this requirement.

Discrete processes, on the other hand, tend to be event driven. The required time to stop filling a container occurs when the desired amount of fluid has been added. The LAS can not synchronize the execution of a function block with an event that may occur at any instant throughout the macrocycle–but Fieldbus provides other tools that can.

If we were to execute the blocks in Figure 13 exclusively on a time-based schedule, as was done in the steam heater example (recall Figures 9 and 10), all three blocks would execute periodically and in sequence at some configured frequency. In the heater example the flow loop executed five times per second, or every 200 milliseconds. In the bottle filling application with a two second fill time, this gives a resolution of only one part in ten, or an error band of ±10%.

While we haven't stated an accuracy requirement, we will assume that ±10% isn't satisfactory. Increasing the execution rate (shortening the macrocycle) will improve accuracy in direct proportion. At some point, however, the execution rate will be limited by the hardware, and probably before adequate control (in this example) is achieved.

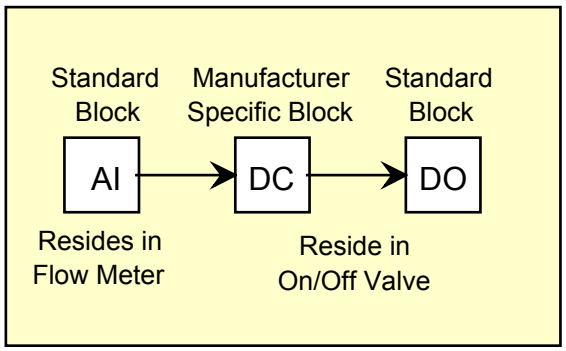

Figure 13. Control Strategy.

Fieldbus actually defines three ways to schedule function block execution, only one is precisely by the clock. A second method is that a block may execute in response to the execution of a connected block, and the third method is manufacturer specific. We will use all three.

The flow meter will run on a time schedule, periodically publishing under control of the LAS every 200 milliseconds. At the start of a fill cycle the DC will record the start time. The DC will then receive periodic values of total flow over the bus, from the meter. For every new data point the DC will forecast (calculate) a future time, tx, when the bottle is projected to be full. Figure 14 illustrates a series of measurements over one fill cycle.

The DC will continuously compare the current time to the current forecast of tx., and send a change state signal to the DO when tx is reached. The DO will immediately execute in response, causing the valve to close. With this technique a resolution of less than one millisecond is easily achieved, which contributes to an error of less than ±0.05%.

Once the valve is closed, the next total flow data point will tell the DC exactly how much fluid went into the bottle. Any error will be used to create a compensation factor for latency in valve closure and other lags, and will be included in the calculation of tx for the next fill cycle. The overall accuracy is determined by the consistency of the latency period, and the accuracy of the total flow meter.

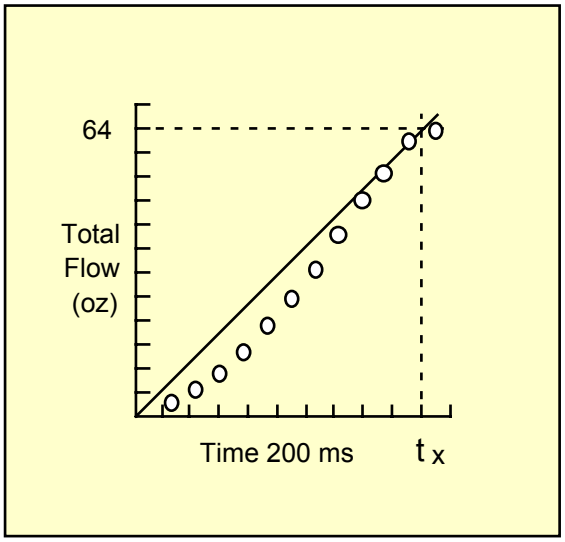

Figure 14. Control Algorithm.

The DC is executing on a manufacturer specific schedule which, in this case, is to calculate tx upon receiving each new total flow value and to initiate action via an interrupt when current time equals tx. The DC does not use the bus for sending information because its output goes directly to the DO block located in the same device. It does, however, subscribe to the total flow value, which is being published by the flow meter every 200 milliseconds.

There are three key points in this example. First, the valve can be actuated at *any* point in time, its execution is decoupled from the LAS macrocycle. Second, the valve communicates with other devices on the network, such as the flow meter, using standard function blocks, and under control of the LAS. Third, all standard Fieldbus behaviors for mode, status and alarms are observed. Any unique parameters needed by an operator to use the manufacturer specific DC block will be accessible by hosts via Device Description technology discussed in the following section.

There are some additional overhead tasks. For example, after a small delay following closing the valve, the DC will send a switch signal to a second DO (not shown) which will be used to increment the conveyor line and get a new bottle in position. Once the new bottle is in position, the conveyor will provide a switch closure which will be read into the DC through a DI block (also not shown) to start the next fill cycle.

By making use of local intelligence, i.e., a totalizer in the flow meter and a discrete controller in the valve, the control system avoids the traditional requirement of very high data rates communicating with a centralized computer to achieve high speed, discrete control. Four such bottle filling stations could be handled on one H1 Fieldbus segment, and many segments can be managed by a simple PC based operator station.

Device Descriptions

From the earlier discussions on function blocks, mode, alarms, etc., it may appear that if the functionality of Fieldbus products is so tightly specified that distinctions among different manufacturer's products could be severely restricted. It could seem that future improvements and innovations might simply be locked out because the Fieldbus specifications could not comprehend all possible variations and future developments.

That is not an unreasonable first expectation; and it would be correct, were it not for the use of Device Description technology. The fieldbus specifications provide for a formal, C-like programming language called Device Description Language (DDL). Using DDL, a device developer will prepare a Device Description (DD) for all parameters and other functions in the device.

The device's DD is readable and interpreted by a Fieldbus specified utility called DD Services. The DD contains information necessary to allow a host using DD Services to communicate with any special parameters or features that a manufacture may choose to incorporate.

a) DD Example

As an example, valve manufacturers find it useful to accumulate the total distance a valve stem has traveled against it's packing seal. Fieldbus has no specified parameters that even relate to this subject. However, each manufacturer uses this type of information in its own proprietary way and if a manufacturer wishes to make some special parameter available to users, it can be accomplished by defining it in the Device DD.

To use a simple illustration, suppose a valve supplier names a parameter WIDX (Wear Index), and wants users to be able to access its value. The Fieldbus protocol supports a "Tag-Dot-Parameter" search service, where *Tag* is a device (or block) identifier and *Parameter* is the name of the parameter in question. This service, in conjunction with the DD technology, will allow a user to type in FCV423.WIDX, for Flow Control valve No. 423, and get a display of the value of the parameter, WIDX, with engineering units and displayed to whatever number of decimal places the DD specifies.

As illustrated in Figure 15, the parameter request is made at a host console. The Fieldbus communication services automatically locate the device and parameter, and read the value. The host uses DD Services to interpret the meaning of the parameter from the DD file for that device. It

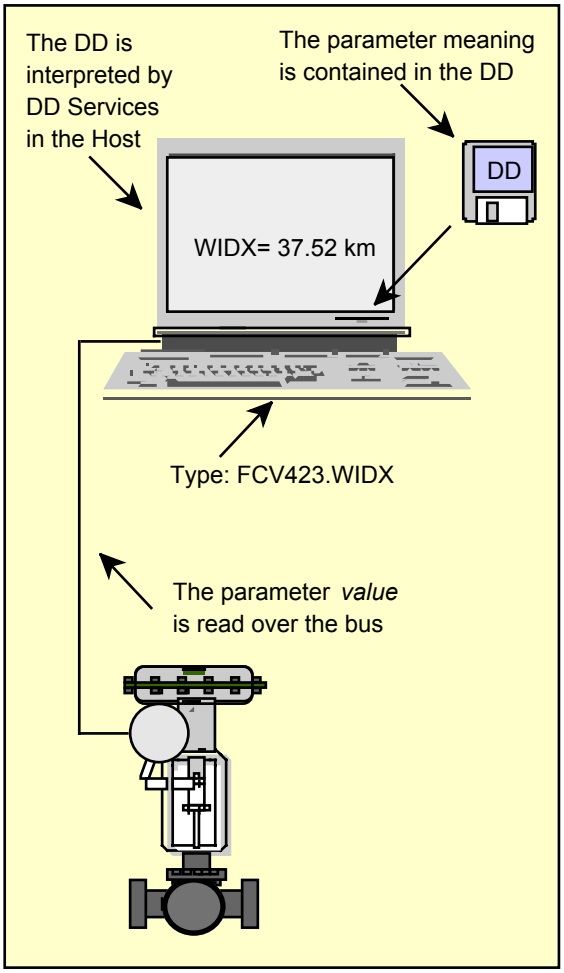

Figure 15. Custom Parameter Search.

is not necessary for the user to know the bus address of the device, only the tag number and parameter name. Custom programming is not required of the host system to facilitate access to the new, non-standard parameter. Other programs in the host can use the same access calls to exploit this feature in any way that the host designers perceive as useful.

It is easy to design-in other useful capabilities by the use of DDs. Valve suppliers are interested in total run time, travel distance, number of reversals, number of starts, number of trips, and

statistical analysis of these measurements. Transmitter suppliers consider multiple sensors, and performance enhancing filters, and all suppliers are interested in improved diagnostic capabilities. Each supplier is free to develop such features, and yet remain interoperable because with DD technology, these manufacturer specific parameters are as easily accessed as are standard parameters.

The DD for manufacturer specific parameters, like the example WIDX, are merely extensions of the standard DD and are otherwise handled in the same way as standard parameters.

To be Fieldbus conformant, a host system must, directly or indirectly, incorporate DD Services (or an equivalent technology). In addition, every registered field device must have a registered DD. Each device's DD is supplied to the host system. Device DD libraries are available from the Foundation on CD-ROM and on-line, and individual DDs are provided by the device supplier.

b) Menus and Methods

Another (optional) feature of DD technology is the ability to incorporate small programs within a DD. An example application would be to provide an interactive dialog session which guides an operator or technician through the calibration procedure for the device. Such programs are called *methods*. A device's DD may contain one or more methods, each of which is initiated from a *menu*. An operator simply selects the desired method from the menu and initiates it, special programming is not required.

c) Device Programs

Two additional capabilities which are specified by Fieldbus, though optional, are *program download* and *program invocation*. The download capability provides a mechanism wherein field devices may receive an upgrade to the device firmware remotely over the bus. This may be useful for version upgrades or other changes in functionality. Program invocation services permit a host to control the execution of a program within the device itself. This can meet a variety of needs, but the primary application is to perform complex internal diagnostics.

Configuration

The procedure for putting an individual device in the desired state for use is called *device configuration*, or by some manufacturers, *commissioning*. It includes setting all default parameter values such as alarm limits, mode, filter values, controller tuning parameters, set points, and more. The procedure for setting up the communication connections between multiple devices on a bus segment is called *bus configuration*, or *commissioning*. This includes providing each device with the data needed to establish the required links between function blocks, setting the communication schedules, and more.

Devices may be configured on-line with the devices on a live bus segment, using an on-line configuration tool. On-line configurators communicate with individual devices over the bus, reading information about the device's address, what function blocks the device has, the addresses of its function blocks, function block execution time, and literally hundreds of other parameters, all required to make systems such as the examples described earlier, work as described. The configurator will also use DDs to obtain the information needed for non-standard parameters.

There are also off-line configurators that perform the same function, but without a live bus. The individual device information is read by the configurator from special files designed to provide the exact same information as would be obtained from the device itself. In order that devices from multiple manufacturers can be configured with a single configuration tool, Foundation Fieldbus specification FF-103 defines the format and design of these files, which are called the Common File Format (CFF).

On-line configuration is performed with an on-line configurator using the actual devices and the DDs for those devices. Off-line configuration is performed with an off-line configurator and the DDs and CFFs for the devices. If configuration is done off-line, the configuration must at some point be loaded into the devices. This could be done during system integration, or device installation.

Host consoles generally provide one or both methods of configuration. There are also PC-based configuration tools available from various suppliers.

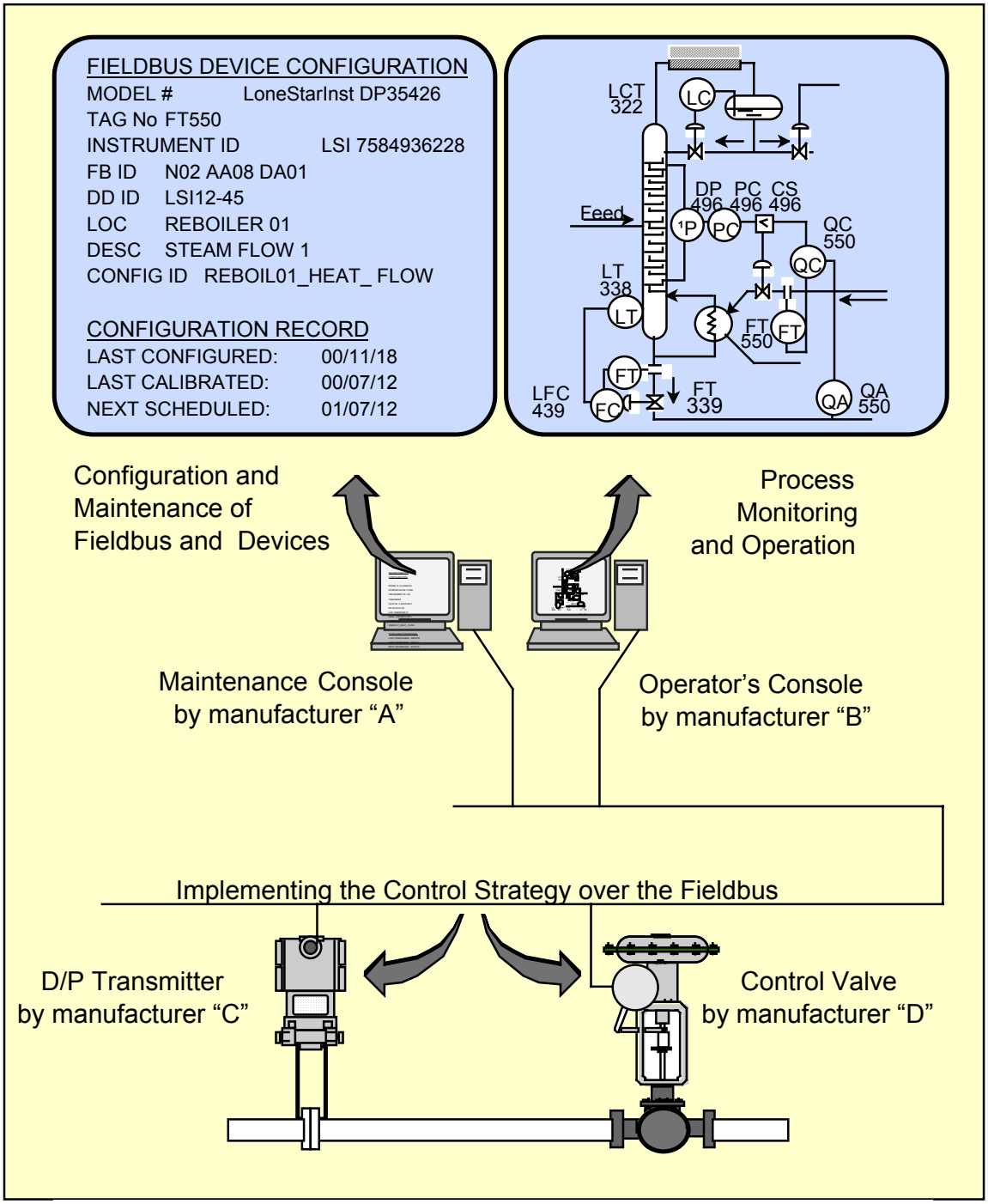

Figure 16. System Illustrating Multiple Aspects of Interoperability.

Viewpoint from the Console

Figure 16 shows several important characteristics of a Fieldbus control system. It illustrates a maintenance console from manufacturer "A" which displays data from the transmitter of manufacturer "C". This is data permanently stored in the resource block of the transmitter and is readily accessible by any conformant host device.

The Process Monitor from manufacturer "B" shows a display of the operating unit. Process values, alarms, and other data are all dynamically included on this display. The data are obtained over the bus from devices of manufacturers "C", "D", and others (not shown).

Constantly changing, dynamic data may be acquired as individual points, or more efficiently as data blocks. Relatively constant, static data may be acquired only when a change has occurred. Fieldbus communications are designed to efficiently provide the currently required display data with a minimum of bus traffic. The following sub-sections on Trends and Views, and a later section on communications, will elaborate on how this is achieved.

Figure 16 illustrates three essential aspects of interoperability. 1) Devices from different manufacturers must communicate with each other. This is accomplished with a standard communications protocol and enforced with conformance testing, 2) Manufacturer specific (custom) parameters must be accessible by devices from other manufacturers. This is accomplished with DD technology, as shown in the previous example. 3) The control strategy is open, and implemented among multiple devices on the network. This is accomplished with an open, published specification for the function block application, enforced with interoperability testing, and illustrated in the previous examples of continuous and batch control.

d) Access Privileges

Each Fieldbus function block contains a parameter called GRANT_DENY. This is a bit string parameter which is set by the user of the Fieldbus system. Host devices are programmed to interpret this bit string so that specific consoles have or do not have access to various parameters within the given block. This provides a flexible method for end users to control the use of various consoles on the system.

e) Trends

Fieldbus provides several features which facilitate systems such as shown in Figure 16. One of these is called *Trend*. A trend provides a short term historical record of parameter values sampled at some multiple of the block execution rate. The trend is stored within the field device and may be efficiently accessed by host consoles or temporary maintenance tools. Trends may also be automatically reported using a Report Distribution[8] when a new set of data is collected. A total of the last 16 values and the associated status are maintained by the trend. The time of the last sample is also maintained.

Trends are configured by the user and any function block input or output parameter may be trended. This includes analog and discrete parameters and bit strings. Trends provide an efficient way to update the display screens on system consoles such as shown in Figure 16.

f) Views

Another method of efficiently providing update information to console screens is through the use of *views*. While trends transmit recent historical data on selected individual parameters, views transmit the current information for groups of parameters. Four specific parameters groups have been defined in the Foundation specifications for the first four views, other views may be defined by users.

Views 1 and 3 contain all the dynamic parameters in a function block. Dynamic parameters are defined as those whose value is determined by the execution of the block. The dynamic parameters of an AI block, for example, include; the process variable (PV), the block output (OUT), a parameter that describes block errors, an alarm summary parameter, and a parameter giving the actual target, permitted and normal modes[9] of the block. There are a total of 12 dynamic parameters in an AI block.

Views 2 and 4 contain all the static parameters in a function block. Static parameters are generally configured or written to the device over the bus

[8] Report Distribution is one of three communication relationships used in Fieldbus and is discussed in a later section.

[9] Mode is discussed later in section; *e) Mode*.

and normally change infrequently. Examples of static parameters include process alarm limits, scaling data, option selections, filter values, controller tuning parameters, etc.. There are a total of 24 static parameters in an AI block.

All four defined views contain a parameter called Static Revision (ST_REV). The value of this parameter is incremented if, and only if, any static parameter is changed.

Views are not automatically transmitted, but must be read (using standard services) by the device wanting the information. It is thus up to the designer of the display screens to decide how much and how often views need to be read to support the desired displays. The mechanism just described relieves the host of unnecessarily reading hundreds of static parameters on a typical bus segment, without missing changes when they do occur.

A change in the static revision parameter also is called an "event", which is treated the same as an alarm as described in the following section.

g) Alarms, Alerts and Events

Anyone familiar with process control applications will recognize the term *process alarm*. These represent limits on the process variable (PV) which, if exceeded, require that a human operator be alerted to the situation. The AI, DI and PID blocks in Fieldbus all have process alarms. There are other conditions that can occur in a control system which should also result in alerting a human operator. In Fieldbus, these conditions are called either alarms or *events*; both of which are referred to as *alerts*. Regardless of the cause, alerts are all handled in a standardized way, though there are differences in some of the details.

Referring to Figure 17, the bottom of the diagram shows the three causes for the generation of alerts. They are block errors, process alarms, and static revisions. These alarm/event conditions all have a common set of data, as shown in the next block up in the diagram. For example, all will be time stamped at the time the alert is first detected. All will contain information about the state of the alarm condition, and the acknowledgment status.

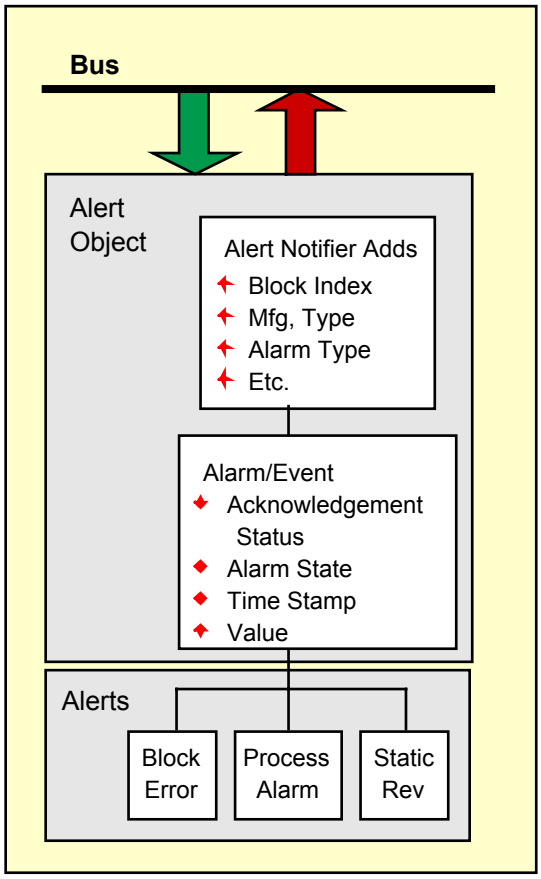

Figure 17. Alert Handling Mechanism.

A program running within the function block application, called the *Alert Notifier*, examines the alarm and event parameters and, if an alert state exists, builds a more complete set of data including address information, manufacturer type and several other details. This is stored in a data set called the *alert object.* The Alert Notifier also builds a report message, called the *event notification message.* Event notifications are broadcast on the bus at the earliest opportunity short of interfering with the requirements for control communications.

The alert notifier alleviates the need for higher level devices to poll each field control device to determine if an alert condition exists. It also relieves the function block from the overhead of event notification processing, so that execution schedules are not affected when an alert occurs.

An event notification is a comprehensive message describing the cause of the alert, a time stamp of when the alert was first detected, an indication of priority, a listing of block index, manufacturer type and other details.

In the case of multiple active alerts, the highest priority will be broadcast first. In the case of equal priority, notifications will be sent in order of age.

An alert results when either an alarm state is entered, or when it is left. An event notification is sent to the Host in both cases. The alert notifier requires confirmation from an interface device that the event notifications has actually been received. The complete sequence for a normal process alarm is illustrated in Figure 18.

The example in Figure 18 starts with an initial condition of no alarm being active. Step 2 indicates the process has entered an alarm state. Steps 3, 4 and 5 have been described in the previous paragraphs. At step 5, after the event notification has been sent, the alert notifier waits for confirmation that a host has received the message. If confirmation is not received after a period corresponding to the value in the parameter CONFIRM_TIME, the message will be sent again.

The alert notifier will continue re-sending on this schedule until confirmation is received. Step 9 indicates that the process condition has returned to normal, or at least within the alarm limits. This results in a repetition of the notification procedure, and another requirement for confirmation.

Step 15 introduces a requirement not previously mentioned; acknowledgment by a human operator. Although this is shown as the last step in the sequence, it can occur any at time the operator becomes aware of the alarm. Most likely, soon after step 6. All confirmations and the acknowledgment must occur to get back to the Alarm Clear state.

The example sequence in Figure 18 assumes that while the alarm is active, priorities do not change, the block does not go to the O/S mode, and the alarm is not disabled. For a comprehensive description of the alert processing state machine, refer to Part 1, Section 4.6 of the specifications.

	ACTIVITY	ALERT STATE
1	Conditions normal	Alarm clear, reported & ack
2	PV exceeds limits	
3		Alarm active, unreported
4	Alert notifier builds message & sends	
5		Alarm active, message sent
6	Alarm confirmed by host application	
7		Alarm active & reported
8	Time passes...	
9	PV returns to normal	
10		Alarm clear, unreported
11	Alert notifier builds message & sends	
12		Alarm clear, message sent
13	Alarm clear confirmed by host application	
14		Alarm clear, reported, not ack
15	Alarm acknowledged by human operator	
16		Alarm clear, reported & ack

Figure 18. Alarm Handling.

For all function blocks having process alarms, there is a parameter that provides an alarm summary. This contains the current alert status, plus an indication of which alarms have or have

not been reported; acknowledged, and/or disabled (and a method of disabling). It is updated upon every execution of the block. A second parameter sets an option to determine whether event notifications will be automatically acknowledged or require operator action.

There are multiple levels of process alarms. If the PV exceeds any of these, the corresponding alarm parameter, will become active. Included in that parameter is the value that exceeded the alarm limit. There is also a corresponding parameter that will contain a user assigned priority from 0 to 15. A priority of eight or higher is regarded as *critical*. Priority less than eight is *advisory*. If the priority is zero, the alarm is suppressed. If the priority is one, the alarm will be set, but a report (event notification) will not be generated.

In the previous section we discussed the communication and content of Views and how static parameters, which seldom change, may be automatically transmitted only when a change has occurred. A Static Revision is called an *event*, and is indicated by the UPDATE_EVT parameter. Included in the parameter is a value that represents the current revision level. This makes it easy for a Host application to detect a change and provide automatic acknowledgment.
An event is also an alert, and is processed like any other alert, see Figure 17. The event has a fixed priority of two.

After process alarms and events, the third source of alerts arises from function block behavior. If a condition develops that could interfere with the proper execution of a function block, it most likely warrants attention from a human operator. In Fieldbus, there are 16 possible block errors. These include such conditions as; configuration errors, memory failures, link errors, etc. Block alarms also have a fixed priority of two. Included in the block alarm parameter is a value that indicates which block error is the cause of the alarm.

The occurrence of block errors and the subsequent generation of an alert does not automatically interrupt control or otherwise intervene in loop behavior. The purpose of an alert is generally to warn an operator so that appropriate action can be taken. It is possible, however, that the event that caused a block error could itself disrupt control.

This will certainly lead to an event notification as described above. Depending on the problem, it is also likely that the mode of the affected block and others will transition to manual or out-of-service. But this is a result of the mode machine, not the alert handling procedure. Alerts do not directly affect block execution or system behavior. To review the specifications, see Function Block Part 1, Section 4.4.3.6 .

h) Mode

All function blocks have a mode parameter, which determines which inputs the block will operate on, and which outputs the block will generate from its internal algorithm. All blocks must support Out of Service (O/S) and, to be useful, at least one additional mode. Figure 19 gives a table of modes required of the most basic five Fieldbus function blocks.

In the O/S mode, the block is executing but it is not calculating an output. The last column in Figure 19 describes how the output of each function block is generated, depending on the block's mode. Local Override (LO) is a mode caused by conditions within the device. Specifically, a fault-state condition or an interlock condition. In LO, the block is calculating an output based on information other than the input value. Initialize Manual (IMan) is a manual mode used to initialize a block's output value during a transition to a control condition.

The appearance of the Cas mode for the DO and AO blocks may surprise readers new to Fieldbus. These are both output blocks and the perspective is that they are controlling a final control element, such as a valve. The valve position is viewed as a local process variable (PV), and the desired value for that PV is indicated by a set point (SP). The DO and AO blocks are consequently treated as the secondary or subordinate controller in a cascade control structure, thus the normal mode is Cas.

The FOUNDATION fieldbus specification describes four mode categories; target, actual, permitted and normal. The *target* mode is the desired mode, typically Cas for an output block and Cas or Auto for a control block. The target mode may be set by a human operator or by an application. The *actual* mode is the one being used by the block at any given moment. For example, the target mode of the temperature PID controller in the cascade control example (see Figure 9) would be Auto. When the upstream (temperature) transmitter failed, the PV signal carried a status of bad, which caused the controller to transition to Man. However the target mode would remained Auto, and when the fault condition clears, the controller will transition back to Auto.

Permitted modes are those that can be used as target modes. For example, the AO block supports five modes. Because of a specific application requirement, a user may wish to prevent use of, for example, the Auto mode for this block. Accordingly, the user can configure the block to permit only RCas, Cas, OS and Man as target modes.

Normal is the mode that should be expected in normal operation. It is defined and read by an interface device for operator information, but is not used by the block algorithm.

After calibration at the factory, the target mode of a block must be O/S for shipping. Target mode is a non-volatile parameter so upon power-up the target mode will always be what it was at power-down. After being configured for an application, the target mode should be set to whatever is appropriate for the application. On subsequent power-up, the target mode will remain unless specifically changed by an operator or an application. Figure 19 also indicates the usual target modes for the five basic blocks.

Mode	DI	DO	AI	AO	PID	Output is Generated By:
O/S	□	□	□	□	□	Last calculated value with status of bad
LO		□		□	□	DO & AO: fault state or external event, PID: tracking
IMan		□		□	□	Internal initialization
Man	□	□	□	□	□	Directly written by human
Auto	■	□	■	□	□	Algorithm processing input(s)
Cas		■		■	■	Algorithm processing input(s)
RCas		□		□	□	Algorithm processing input(s)
ROut					□	Output directly written by a supervisory computer

□ or ■ Indicates required mode
■ Indicates usual target mode

Figure 19. Function Block Mode chart.

Communications

Most of the fieldbus behavior and characteristics which we have been discussing are provided by the Function Block Application. However, for the application to perform as specified, it must have the support of a set of communication services which are specified in: FF-801, FF-822, FF-870, FF-875, and FF-880. These, as is the case for the Function Block Application, are open specifications maintained and published by the Fieldbus Foundation. The most fundamental features of the communication stack will be described here.

Communication between function blocks for control purposes was described earlier as being just in time (for function block consumption).
Control is not the only communication requirement. The Link Active Scheduler (LAS) is responsible for controlling all communications. The governing mechanism to achieve this control is called a *delegated token*. To communicate on the bus, a device must have received a special message called a token.

Token passing is a widely used method for controlling communications on a bus. In some designs the token circulates from device to device, in other cases it may shuttle between a master and various slaves. Techniques for determining token hold time also vary, but token passing protocols are often criticized for not being able to assure that transfers of time critical data occur when they should. For continuous control, data transfers at precise points in time are essential, a characteristic sometimes called *determinism*.

The Fieldbus tokens do not grant unlimited communication rights. Tokens delegate very specific instructions of either *what* is to be communicated, or *how long* communication may take place. In either case, the right of a device to use the network is defined and limited.

The LAS operates with three priorities. The first is to assure that communication for control occurs at precisely scheduled points in time; this was discussed in the steam heater example. The second priority provides for bus maintenance. This consists primarily of periodically distributing time to devices on the network for synchronization of device clocks, and of determining what devices are presently on the network (because devices can be dynamically added or removed). The third priority is to allow devices to communicate to each other for whatever purpose may have arisen, sending Event Notifications, for example.

The basic algorithm used by the LAS is shown in Figure 20. The logic shown is an implementation of the priorities described above. The LAS sends tokens with one of three basic instructions which are: Compel Data (CD), Probe Node (PN) and Pass Token (PT). The LAS also broadcasts a limited set of messages, such as Time Distribution (TD), which do not authorize the receiver to respond. When it is time for control data to be sent, the LAS will send a CD to the appropriate device and to the specific block and buffer within that block. This instruction requires immediate publication of the data and permits no other response. This is a very short, time bounded message.

During bus maintenance the LAS will broadcast a TD, which is concurrently read by all devices. Each device, under its own local control, will compare its internal clock reading with the new time value. For H1 devices, clocks will be reset to maintain accuracy within 1 millisecond. (A good design feature will also cause the device to make a small hardware adjustment which improves accuracy on every reset occurrence.)

The Fieldbus specification provides a maximum of 256 addresses on an individual bus segment. Sixteen of these are restricted to special functions such as group addressing, and will not be considered here. Four more are for temporary devices, such as “hand held” maintenance tools. Another four are reserved as default addresses, discussed further below. This leaves 232 addresses available for field devices. The actual number required is normally much less, and the number to be used in a given application is configurable.

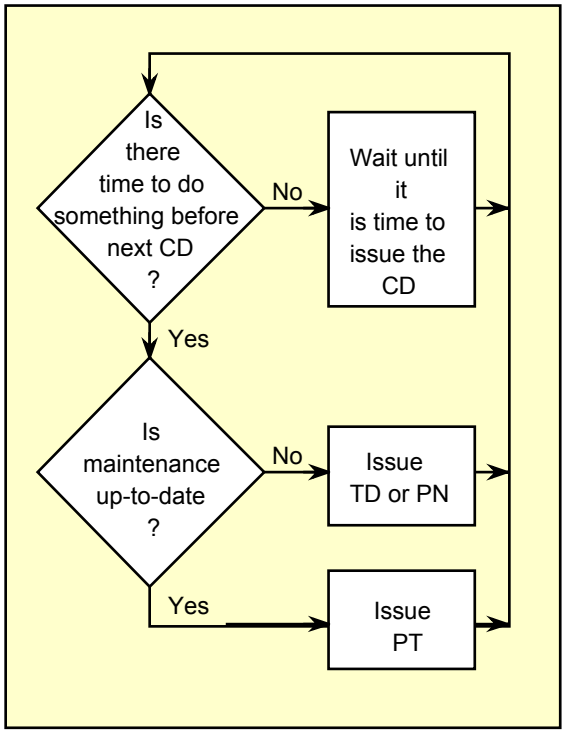

Figure 20. Algorithm for LAS.

When a device is configured for use on the network, it will be assigned an address which is retained in non-volatile memory (NVM). Hardwired addressing is *not* permitted.
The LAS maintains a list, called the *live list,* of all active devices on the network. It does this by sequentially sending a PN to every valid (configured), unused address on the network. When a newly attached device receives a PN, it

is required to return a short, defined message called a Probe Response (PR). The LAS will then initiate an identification sequence to determine the device ID, Tag number, and other information about the new device.

If the new device has not been configured with an address, it will use its own intelligence to select one of the default addresses mentioned above. The LAS will probe the default addresses and proceed with the same identification sequence when a device is found. If the new device has been configured with an address that duplicates an address already in use, it will not get a PN, since that address is for a device already on the live list. At some point, however, it will receive a Pass Token (PT) intended for the original addressee. The new device will detect a miss-match of certain other information in the PT and will re-assign itself to a default address. This is one reason why hard wired device addresses are not permitted.

Each device on the live list will periodically receive a PT. If the device does not return a response to the LAS, the device will be removed from the live list after three failed attempts. The LAS will then broadcast a Live List Update to be read and used by any redundant LAS devices on the network.

The LAS uses the CD, TD, and PN to cause specific actions to occur under control of the LAS itself. In sending a PT to a device, it is giving the device an opportunity to communicate messages under control of the device itself. With the PT comes a value called the *maximum hold time.* So, although the LAS delegates control of *what* the device may communicate, it retains control of *how long* the device can use the bus. The LAS thus maintains determinism, and can assure that the precise schedule required for control data is maintained.

Upon receiving the PT, a device may use as much or as little of its maximum hold time as it needs, but not more. If it has no need, it will return the PT immediately and the LAS can use the time for something else. If the device needs more time, it will return the PT at the end of the maximum hold time with a request for more time. The LAS can grant additional PTs to the device, based on a time usage algorithm not discussed here.

Devices use the bus time granted by a PT to report alarms and other Event Notifications, exchange Read/Write messages with humans, communicate diagnostic information and answer requests from operators or other devices. While the PT grants the device control of what to communicate, the LAS can influence what the device chooses by providing a priority within the PT. For example, to indicate that alarms are to be reported before answering an operator's request for the device model number.

The communications between fieldbus devices all fall into three classifications. The method used in transmitting data for control purposes is called Publisher/Subscriber. In this model, the data to be sent is buffered, meaning the most current data will be transmitted and any older data is overwritten. Data required for control must be transmitted at precise intervals and so transmission is scheduled. There is no requirement (or opportunity) for the receiver to confirm that data is correctly received. If a data point is missed, the value is immediately stale and the system will need to rely on the previous value until a future value is transmitted. Finally, the data is broadcast on the network from a single source to any device needing the data, one-to-many.

The second category was illustrated in the earlier example of a thermocouple failure where a block alarm was generated. This is called a Report Distribution and provides somewhat different features. The data to be sent is queued rather than buffered. This means that if a series of values or messages are created, none are discarded. They will each be sent in turn as time is available. It is important that such messages to be transmitted with urgency. This is best accomplished by sending them at the earliest opportunity after the occurrence of the underlying event, rather than on a scheduled basis. It is also important that the message is correctly received. This is assured by requiring that the appropriate recipient confirm receipt by sending a return confirmation message. Otherwise, the original transmission will be periodically re-transmitted. Reports are initiated by the sender and are transmitted as a one-to-many message similarly to Publisher/Subscriber.

The third classification of communications is illustrated by an operator at a host device changing the value of a parameter in a field

device, such as a set point or tuning parameter. This model is called Client/Server. The values or messages are queued, so each will be transmitted in its turn. They are transmitted in the unscheduled time during a macrocycle. Re-transmission by the sending device will be repeated until a confirmation message is received from the recipient. This exchange is between one specific device and another, thus it is a one-to-one communication relationship.

The three communication relationships are summarized in Figure 21. Regardless of the communication type, all transmitted messages contain a 16-bit frame control field similar to a cyclic redundancy check. For continuous H1 communications this results in a theoretical error rate of less than one undetected error in 21 years.

Communication Relationship	Publisher/Subscriber	Report Distribution	Client/Server
Characteristics	Buffered, network initiated, scheduled, unconfirmed, one-to-many.	Queued, user initiated, unscheduled, confirmed, one-to-many.	Queued, user initiated, unscheduled, confirmed, one-to-one.
Example Uses	Continuous, real-time control.	Process alarms, block alarms, trends, and other events.	Operator access such as set point and tuning changes, alarm management, access display views, remote diagnostics.

Figure 21. Characteristics and Example Uses of Communication Relationships.

High Speed Ethernet (HSE)

The preceding descriptions and examples have focussed on the characteristics of H1 Fieldbus behavior. As explained in the Background section at the beginning of this Primer, a higher speed physical layer specification was always intended for selected process applications and for factory (discrete parts) automation. The original high speed solution, called H2, was to consist of the identical protocol and function block application running on different media at either 1 Mbit or 2.5 Mbit per second.

In March of 1998, the Foundation Board of Directors re-directed the high speed solution, and terminated further work on H2. The new approach would be based on Ethernet and was to utilize, as much as possible, commercially available, off-the-shelf technology.

The new solution is called *HSE*, for High Speed Ethernet, and is designed to integrate multiple protocols, including multiple H1 Foundation fieldbus segments as well as foreign protocols. The connectivity capabilities of HSE are described in the following section.

a) What HSE Does

The network in Figure 22 shows a Host System operating on an HSE bus segment labeled Segment A. It is shown communicating through an Ethernet switch to two H1 segments, a second HSE segment, and to a segment running a foreign protocol such as Modbus.

Any of the devices connected to the switch may attempt communication to any other device and it is the function of the switch to provide the correct routing and to negotiate transmission without collisions.

The connecting mechanism between HSE and H1 segments is called a Linking Device (LD). A typical LD will serve multiple H1 segments, though for simplicity only one segment per LD is shown in Figure 22. The connection between HSE and a foreign protocol is through an I/O gateway. The capabilities of the network shown are as follows.

A ⇔ B and A ⇔ C The HSE host interacts with a standard H1 device through a Linking Device. In this situation the HSE host is able to

configure, diagnose, and publish and subscribe data to or from the H1 device.

A ⇔ D The HSE host interacts with an HSE device. In this situation the HSE host is able to configure, diagnose, and publish and subscribe data to or from the HSE device.

E ⇔ B This is a situation where a foreign device wishes to subscribe to an H1 resident device or vice versa. The foreign device may be another network, a device connected via telecommunications services, or a directly connected end-system.

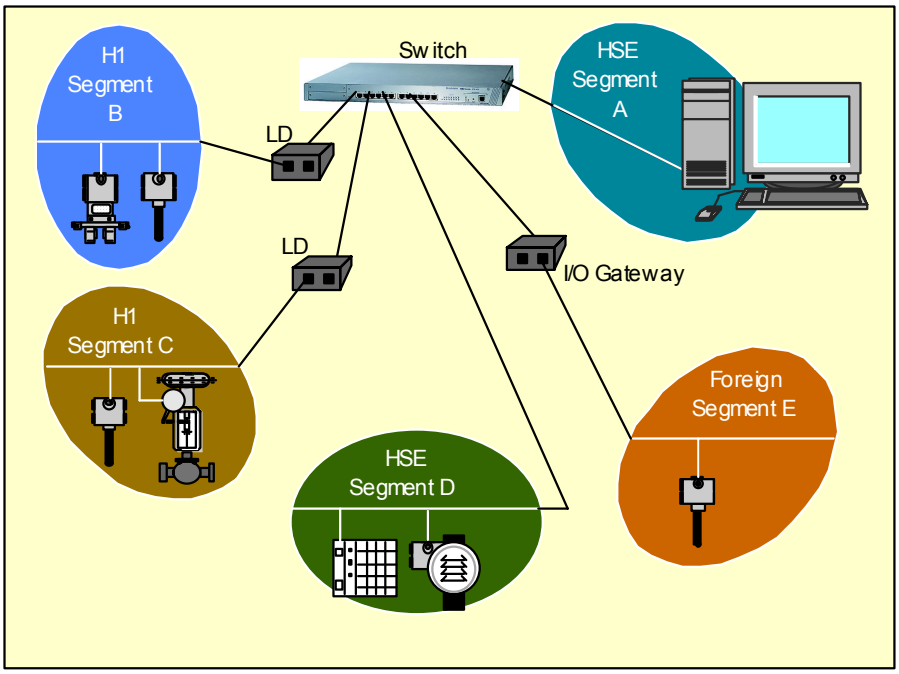

Figure 22. HSE/H1 Network.

B ⇔ C In this situation, the interaction is between two H1 devices on two distinct H1 bus segments. The segments are connected to the Ethernet with Linking Devices. Communications between devices on Segments B and C are functionally equivalent to communications between two H1 devices on the same bus segment.

A ⇔ E This connection defines the relationship between a foreign device and the Foundation fieldbus application environment. The connection is made with a device called an *I/O gateway*. As defined in the HSE specifications, the foreign device is seen as a publisher to an HSE resident subscriber. Host Device A can treat the data stream from the I/O gateway in the same manner as it treats the data streams from devices on segments A, B and C.

b) H1/HSE Interconnections

Ethernet specifies a physical layer and a data link layer only. By adopting Ethernet, the HSE solution has the entire range of standard, off-the-shelf hardware components available for network implementations. However, to be useful, higher level protocol layers must be added. Standard Information Technology (IT) and internet applications almost universally utilize Transport Control Protocol and Internet Protocol (TCP/IP), in addition to the Ethernet data link. More will be said about the protocol in the following section.

Messages sent on an Ethernet are bounded by a series of data fields called a *frame*. The combination of a message and frame is called an *Ethernet packet*. Typically, a similarly encoded TCP/IP packet will be inserted in the message field of the Ethernet packet. The basic construction is shown in Figure 23.

Fieldbus uses a similar data structure where messages are bounded by addressing and other data items. That which corresponds to a packet in Ethernet, is called a Protocol Data Unit (PDU) in Fieldbus.

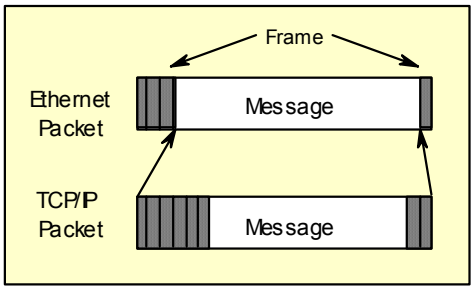

Figure 23. Typical Internet Packet.

Now consider a communication between two H1 devices over an intervening HSE segment, as illustrated by B ⇔ C in Figure 22. The easiest method might be for the LD, upon receiving a communication from an H1 device, to simply insert the entire H1 PDU into the message part of the TCP/IP packet. Then the LD on the destination H1 segment, upon receiving the Ethernet packet, would merely strip away the Ethernet frame and send the H1 PDU on to the receiving H1 bus segment. This technique is called *tunneling*, and is commonly used in mixed protocol networks.

The solution developed for HSE is somewhat more complex, and is dramatically more efficient than tunneling. As illustrated in Figure 24, most of the Fieldbus PDU is inserted into the data field of a TCP/IP message field. However the

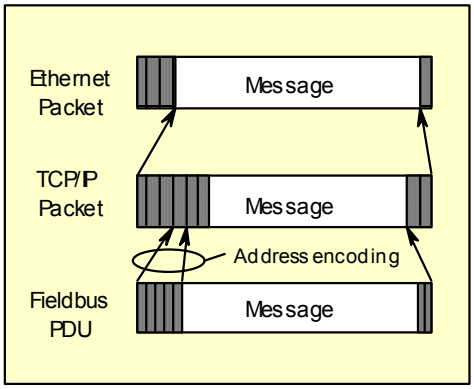

Figure 24. Nested Messages in HSE.

Fieldbus address is encoded as a unique TCP/IP address. The entire TCP/IP packet is then inserted into the message field of the Ethernet packet.

Because of the HSE encoding scheme, networks having multiple LDs can locate and transfer messages to the correct destination much more quickly, with far less extraneous bus traffic, than would occur with tunneling. Perhaps even more important, every H1 device (and every HSE device for that matter) has a unique TCP/IP address and can be directly accessed over standard IT and internet networks.

Scheduled communication between H1 devices is still controlled by the associated H1 Linkmasters, so determinism and other H1 characteristics are maintained across the network. The message size in an Ethernet packet is variable, but is limited to a specified maximum to prevent one device from holding the bus hostage.

c) Basic Details of the HSE Specification

The HSE specification relies, wherever possible, on previously existing specifications. Use of existing Ethernet technology makes available reliable, low cost hardware for HSE implementations. The specifications for this physical layer, and the Ethernet data link layer are maintained by the Institute of Electrical and Electronic Engineers (IEEE).

Also used are established internet protocols which are maintained by the Internet Architecture Board, and are widely taught in digital communications curricula. These include:

- TCP Transport Control Protocol
- UDP Unit Datagram Protocol
- IP Internet Protocol

Existing Fieldbus specifications, which have been widely tested in H1 applications and are already maintained by the Fieldbus Foundation, were used where applicable. These include:

- FMS Fieldbus Message Specification
- SM System Management
- FBAP Function Block Application Process

Finally, where necessary, new specifications were developed and tested to provide a complete

high speed communications *and* control solution. The new technology includes:

- FDA Field Device Access (Agent)
- HSE SM System Management for HSE

The various components of the HSE solution can be described in terms of the OSI model, as shown in Figure 25. It can be seen that Layers 1 and 2 are standard Ethernet. Layers 3 and 4 are covered by the TCP/IP suite of protocols. Layers 5 and 6 are not used.

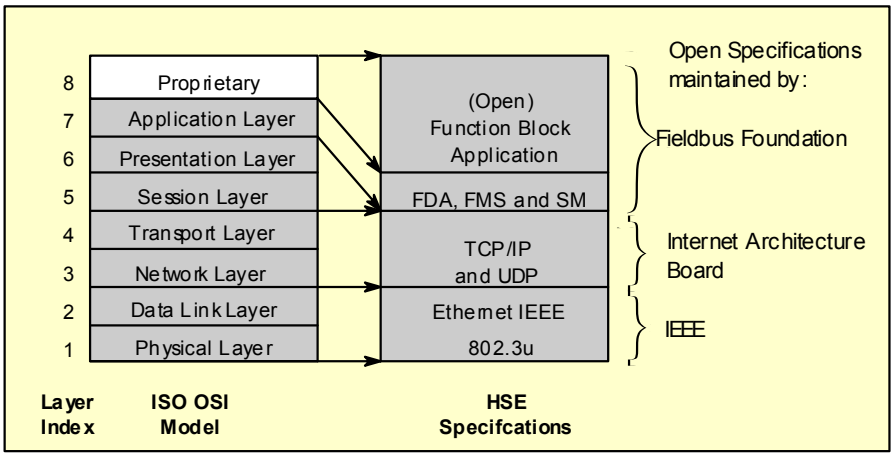

Figure 25. OSI Model and HSE Protocols.

The FDA Agent allows System Management (SM) and Fieldbus Message Specification (FMS) services used by the H1 devices to be conveyed over the Ethernet using standard Unit Datagram Protocol (UDP) and Transport Control Protocol (TCP). This allows HSE devices to communicate to H1 devices that are connected via a Linking Device.

The FDA Agent is also used by the local Function Block Application Process (FBAP) in a HSE device. Thus, the FDA Agent enables remote applications to access HSE devices and/or H1 devices through a common interface.

The HSE SM kernel ensures that system level functions in each device are coordinated. These functions include system time, addition and removal of devices from the network, and function block scheduling.

System Management assures that the clocks of HSE devices, including LDs, are synchronized. Normally, each Linking Device will serve as the primary LAS for all H1 segements connected to that LD, thus all H1 device clocks will be synchronized with system time. Local time is used to time stamp events so that event messages from devices may be correlated across the system. Local time is also used to schedule the execution of the local function blocks.

d) Redundancy

HSE provides for various levels of redundancy up to and including complete device and media redundancy. HSE fault tolerance is achieved by operational transparency i.e., the redundancy operations are not visible to the HSE applications. This is necessary because HSE applications are required to coexist with standard Information Technology (IT) applications.

The HSE LAN Redundancy Entity (LRE) coordinates the redundancy function. Each HSE device periodically transmits a diagnostic message representing its view of the network to the other HSE devices on both Ethernet interfaces. Each device uses the diagnostic messages to maintain a Network Status Table (NST), which is used for fault detection and transmission port selection. There is no central "Redundancy Manager". Instead, each device determines its behavior in response to faults it detects.

e) HSE Summary

HSE is designed to integrate multiple protocols, including multiple H1 Fieldbus segments and foreign protocols, as illustrated in the network in Figure 22.

A key addressing concept employed by HSE is the use of the Internet Protocol (IP), rather than tunneling. HSE uses the Dynamic Host Configuration Protocol (DHCP), Internet Protocol (IP), as well as System Management functionality to assign addresses. Each device will receive an IP address from the DHCP server. H1 addresses are mapped to IP addresses by the Linking Device. When a device is accessed via the HSE network, the IP address is used. Linking Devices sort HSE packets and send the appropriate H1 messages to the H1 devices.

The key concept used by HSE in its redundancy specification is that each device chooses the best communication path. The status of the network is shared among all (HSE) devices. The shared network status allows devices to independently select ports for communication. The connectivity through all ports allows devices to avoid bad or busy ports.

All devices periodically transmit diagnostic messages. The messages contain the view of the network health and status as seen by the device. The devices use the diagnostic messages to build a Network Status Table for port selection. Diagnostic messages include information such as sequence numbers, device number, number of ports, port used for this message, status of all ports, and current selected transmission port.

While the HSE name comes from High Speed Ethernet, the HSE specifications permit the use of 10 Mbit/sec Ethernet (also called 10 Base-T), 100Mbit/sec Ethernet (also 100 Base-T or Fast Ethernet) 1Gbit/sec Ethernet (also called Gigabit Ethernet) or future standards including optical links. The networks may be either switched or un-switched. Despite this intended flexibility, the only tested and demonstrated implementations, as of this writing, are *Fast Ethernet* using *switched networks.*

For furhter information on HSE, readers are referred to the *Foundation Fieldbus System Architecture (H1+HSE)* specification number FF-581 FS 1.1.

Conclusion

The specifications that define Foundation fieldbus are owned and maintained by the Fieldbus Foundation, and are promoted as Foundation fieldbus.

The solution distributes functionality across the network and makes maximum use of intelligence in the individual field devices. It provides a deterministic protocol for real-time control. It defines a standardized, object-oriented, function block model for application software and a unique Device Description technology to achieve multi-manufacturer device and host interoperability without custom programming.

Foundation fieldbus represents an extensive combination of technology and organizational infrastructure. It provides a tested, open specification for interoperable devices which are designed to support diagnostics, monitoring and control in mission critical applications. But the technology goes well beyond written specifications. It includes multiple sources for development and configuration tools, communication stacks and applications software. There are automated test systems and documented test procedures for testing stack conformance and device interoperability

The Fieldbus Foundation provides an established infrastructure which supports the test programs, maintenance and improvements in the specification, maintains and supplies key software utilities, and provides an organization where member companies can cooperate on technology development and exchange.

Industrial process and manufacturing automation applications share many common requirements. Both can benefit from devices having multi-manufacturer interoperability, both need realtime scheduled and unscheduled bus access, they must deal with physical resource constraints, and both need increasing device autonomy. There is a common need for monitoring and control systems where subsystems can report error and operational health conditions, and perform remote calibration, diagnostics and program execution.

There is a need for a standardized communication interface to the physical resources of a device, that does not in turn

require that the physical resources themselves be standardized. Similarly, there is a need for a standardized, modular application that accommodates the addition of unforeseen functionality, without the need for special tools and custom programming.

By meeting all of the above requirements, Foundation fieldbus has the potential for dramatic reductions in overall system cost. These savings arise from the technology and the architecture of fieldbus itself, and from the cooperation among suppliers, system integrator and end users.

APPENDIX I
EXCERPTS FROM THE IEC/ISA 1987 DRAFT REPORT

These excerpts from the draft report illustrate the committee's focus, and provide an underlying understanding of the direction that foundation fieldbus has followed.

Draft Report Excerpts

"3.1.4. Communication Integrity - For safe operation, the system should include error detection. A value of 20 year mean time between undetected transmission errors would assure reliable operation in the typical plant environment...." [A discussion of noise sources and RFI/EMI issues was included.]

"3.1.5. Connection/Disconnection of Nodes - The bus should be capable of continued operation while a connected device (node) is being connected or disconnected…"

"3.2.6. Block Transfer - The protocol should provide a mechanism to transfer large blocks of data without delaying the communication of process data …"

"3.2.7. Redundancy - The following describes H2 applications only. The protocol should support full end to end redundancy of the fieldbus system as an option.…"

"3.2.12. Field Device Status Information - This function would add provision in the communication protocol and in the messaging capability to include a condensed indication of field device status with its current data or in the response to an output command. The condensed indication should include levels of priority.…"

"3.2.13. Addressability - It should be possible for each device or each process variable from being addressed at the user level by a unique identifier such as a tag name. For example, a human interface device could seek the location of a specific field device by a unique tag name."

"4.5. Commissioning Data - It is important to have the correct device installed in each service during construction. The work that is currently done in loop checkout could be greatly simplified if each field device could contain the device's tag name, serial number, range , etc. in readable , non-volatile digital memory. As soon as the device is connected to the control system, the system could then interrogate the information to confirm that communication is established with the correct device with the correct operating characteristics. The standard protocol should include provision for accessing this data."

"4.9. Event Relative Time Stamping - The communications protocol should allow field devices to time stamp events with a resolution of at least 1 millisecond for the H1 applications of the fieldbus, [and] 0.1 milliseconds for the H2 applications of the fieldbus.…"

"4.18. Access Security - The standard should provide for access security. Multiple access levels and the ability to change the access rights should be provided."

"4.21. Addition of Remote Control - There will be an incentive to implement a conformance class that supports functions required to implement automatic control in the transmitter, the valve positioner, or the junction box. The protocol should be able to support the required modes, the time-out gates that may be needed, the anti-windup indicators, etc."

"4.23. Maintenance Information Capture - Some implementations of the fieldbus standard may have an incentive to include the capture of certain data associated with the field device itself. It is assumed that this function may only need the inclusion of appropriated parameter names in the highest level of the protocol. Two examples of this type of function are detailed here." [The examples describe a maintenance record and a database to build an audit trail for all changes made to the device.]