1—Basics of PROFIBUS Operation

To help you quickly get a feel for the capabilities of PROFIBUS, its salient features are summarized in Table 1-1. Note that these features are somewhat generalized – network speed is shown as 9.6kbit/s (lower limit) to 12,000kbit/s (upper limit), although the network speed may vary depending on the medium or technology used. For example, MBP-IS, the medium used for PROFIBUS PA, operates at a single baud rate of 31.25kbit/s. *An interesting feature of PROFIBUS is that the protocol is the same no matter what transmission medium or technology is used.* Thus, PROFIBUS DP (utilizing RS-485 copper, fiber optics, infrared, and so on) and PROFIBUS PA (utilizing MBP-IS), use the same protocol and the devices can exchange cyclic I/O data with a standard DP master.

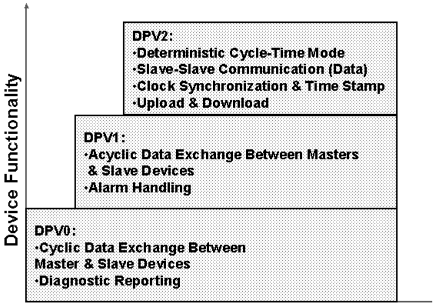
FMS will not be covered in this guide since its primary function of peer-to-peer (master-master) communication has been mostly replaced in recent years by Ethernet-based protocols.

Table 1-1 – PROFIBUS Features

Communication Methods	Master/Slave, Multi-Master/Slave	
	Publisher/Subscriber	
Network Speed	9.6kbit/s – 12,000kbit/s	
Data Transfer Size	Up to 244 Bytes	
Transmission Media/Technologies: Maximum No. Nodes	RS-485 STP Copper: 126	
	Fiber Optic: 126	
	IR: 126	
	RF: 126	
	Slip Ring: 126	
	MBP-IS: Depends on Power Budget	
Maximum Distance	RS-485 STP Copper (Segment/With 9 Repeaters)	9.6kbit/s: 1,000m/10,000m
		12,000kbit/s: 100m/1,000m
	Fiber Optic (Between Fiber Optic Repeaters)	Plastic: 50m
		Multi-Mode Glass: 400m
		Single-Mode Glass: 15km
	IR & RF	Varies With Vendor Product
	MBP-IS	31.25kbit/s: 1.9km Max. Depending on Cable Type
Diagnostics	Standard: 6 Bytes	
	Detailed: Up to 238 Bytes Total	Device-Related
		Module-Related
		Channel-Related

PROFIBUS is an open, vendor-independent protocol that became part of the international standard IEC 61158 in 2000. Though the protocol is mature, it certainly is not static! Over time, it has been extended into new application areas by working groups of employees from companies that have similar products and target application areas. These extensions have always been developed under the requirement that they ensure "backward compatibility." This pocket guide is not the place to discuss the details of these extensions and will concentrate primarily on the basic DPV0 operations; however, Figure 1-1 lists the extensions that have been standardized in the past few years.

Figure 1-1 – PROFIBUS DP Extensions


Device Functionality
DPV2:
·Deterministic Cycle-Time Mode
·Slave-Slave Communication (Data)
·Clock Synchronization & Time Stamp
·Upload & Download
DPV1:
·Acyclic Data Exchange Between Masters & Slave Devices
·Alarm Handling
DPV0:
·Cyclic Data Exchange Between Master & Slave Devices
·Diagnostic Reporting

As Figure 1-1 shows, DPV0 is the foundation for PROFIBUS and was the first version after FMS. DPV0 came from optimizations to FMS, the original PROFIBUS protocol, to support fast I/O data exchange. PROFIBUS DPV1 added extensions that allowed run-time reading/writing of parameters for more sophisticated devices such as intelligent drives, for example, and PROFIBUS PA field instruments, such as valve positioners, pressure transmitters, and so on. DPV2 added extensions primarily so that motion-control applications can be performed directly across PROFIBUS rather than requiring a secondary motion-control bus.

This first chapter will give you a brief overview of the cyclic I/O data exchange (DPV0) operation of a PROFIBUS network. It will not deal with *how* you set up the network to operate. Chapter 2 will discuss that topic. Here, we will cover four basic aspects of a PROFIBUS DP network:

- Master/Slave Concept: A general overview of device interactions.
- Device and System Startup: Just how does the master get all those devices and the system into "operational" mode for control?
- Cyclic I/O Data Exchange: What the network normally does when it is humming along at up to 12,000kbit/s.
- Device Diagnostic Reporting: What diagnostics can a device report and what does a device do when it detects a diagnostic condition, for example, a wire break on an output point?

1.1 Master/Slave Concept

PROFIBUS DP is a network that is made up of two types of devices connected to the bus: master devices and slave devices. It is a bi-directional network, meaning that one device, a master, sends a request to a slave, and the slave responds to that request. Thus, bus contention is not a problem because only one master can control the bus at any time, and a slave device must respond immediately to a request from a master. Since a request from a master to a slave device is heard by all devices attached to the bus, some mechanism must exist for a slave device to recognize that a message is designated for it and then respond to the sender. Hence, each device on a PROFIBUS network must have an assigned address. For specifying the address, most devices have either rotary switches (decimal or hexadecimal) or DIP switches. Some few devices require that their address be set across the bus using a configuration tool. This concept will be discussed in the next chapter.

The PROFIBUS protocol supports addresses from 0 to 127. However, addresses 126 and 127 have special uses (discussed in Chapter 2) and may not be assigned to operational devices. Address 0 has become something of a default address that vendors assign to network configuration and/or programming tools attached to the bus. Thus, the addresses that may be used in practice for operational devices – for example, PLCs, I/O nodes, drives, encoders, and the like – are 1 to 125.

1.2 Device and System Startup

The user specifies which slave devices the master should find on the bus as well as what information is to be transferred from the master to each slave during this startup phase. (Relax…this is much easier than you might think!) All of the information that the master must know to start up the bus comes from a configuration database file that is generated by a PROFIBUS configuration tool.

Each vendor of PROFIBUS master devices offers a configuration tool for generating the database file for their masters. However, once one has learned how to use any of these tools, it is generally quite easy to transfer this knowledge to another tool because all PROFIBUS configuration tools must share certain common functionality. A configuration tool for cyclic I/O operation must be able to do the following:

- process GSD (device description) files and maintain a hardware catalog of devices to be configured on the bus
- allow the PROFIBUS device address to be specified
- allow the specification of the input and output data to be transferred between master and slave
- allow certain startup parameters to be selected in order to activate specific operating modes or features of the device
- allow selection of the system baud rate
- generate the database file so it can be used by the master

At the same time a vendor develops a slave device, it must develop a device description (GSD) file. This file completely describes the PROFIBUS functionality of the device – for example, baud rates supported, possible input/

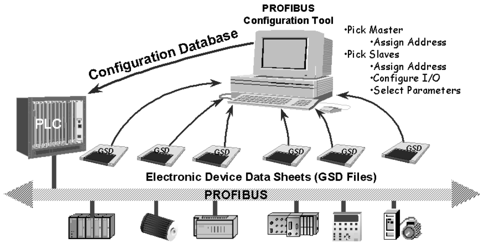
output data configurations, startup parameter choices, and so on. These GSD files can typically be downloaded via the Internet either from www.profibus.com or from an individual vendor's web site. Once you "install" the GSD file for a device into the PROFIBUS configuration tool, it will appear in the tool's hardware catalog, which enables it to be configured for bus operation.

The installation process varies for different configuration tools, although it is extremely simple in any case. In some tools, installation consists of nothing more than copying the GSD file into a "gsd subdirectory," while in others one simply "imports" the new GSD file into the hardware catalog by selecting that option from a menu.

Once all the appropriate GSD files are installed into the configuration tool, you can define a bus configuration. This is a straightforward process. You first pick the appropriate master from the master device list in the hardware catalog and assign a PROFIBUS address. Not surprisingly, you will generally see only master devices from a specific vendor within that vendor's configuration tool! This is nothing to be concerned about. We are really only concerned that the tool process the standard GSD files for slave devices, and it is required to do this. You repeat the following steps until the entire bus configuration has been described: Select a slave device, assign the PROFIBUS address, specify the I/O to be exchanged, and select the appropriate parameters for the desired operation of the device. You then save this bus configuration file and generate the configuration database.

You can now load this configuration database file into the master device. The mechanism for loading the database file into the master may vary with each vendor. The most common mechanism is to download the file into the master via a serial port. Some configuration tools allow you to load the file into a flash memory card that is then inserted into the master for download. Some older systems required an EPROM to be programmed and then inserted into the master for download. No matter what mechanism is used, after download, the master has the information necessary to start up all the devices in its configuration. This information is stored in retentive memory. Figure 1-2 depicts the functionality of a PROFIBUS configuration tool.

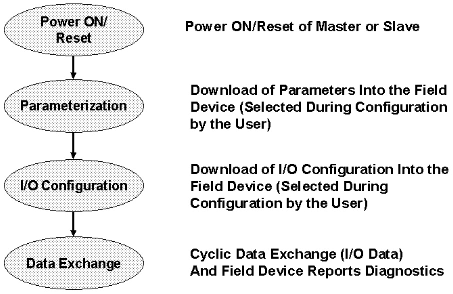
Figure 1-2 – PROFIBUS Configuration Tool Functionality



The master must now determine if the devices at the assigned addresses contained within the configuration database are physically on the bus and initialize them for "operational" or "data-exchange " mode. To get the devices into this mode a PROFIBUS master goes through a well-defined sequence of interactions with each of the slave devices in its bus configuration.

Figure 1-3 shows the steps in the startup sequence for a slave device. For instance, if the master device experiences a power loss, when it powers back up it uses the configuration database in retentive memory to go through the startup sequence with each device in its configuration to get all devices back into operational mode. If a slave device fails and must be replaced, the master recognizes that a replacement device of the same type and with the same PROFIBUS address has been attached to the bus. When it does, it goes through this same startup sequence and automatically brings the device into operational mode. Voilá…Plug & Work!

Figure 1-3 – Slave Startup Sequence


Power ON/ Reset
Power ON/Reset of Master or Slave
Parameterization
Download of Parameters Into the Field Device (Selected During Configuration by the User)
I/O Configuration
Download of I/O Configuration Into the Field Device (Selected During Configuration by the User)
Data Exchange
Cyclic Data Exchange (I/O Data) And Field Device Reports Diagnostics

1.3 Cyclic I/O Data Exchange

After the bus system has been “started up”, the normal interaction between a master and each of its assigned slaves is to exchange I/O data, as shown in Figure 1-4. The master, a PLC with a PROFIBUS interface, for example, sends output data to a slave device in its configuration. The addressed slave immediately responds with its input data. *It is important to grasp this concept of output data sent from the master to the slave and input data returned from the slave to the master.* This “directional” attribute of the I/O is identical to I/O that is hardwired directly to backplane I/O in a PLC rack. It typically maps into the input and output areas of PLC memory, as shown in Figure 1-5, and can generally be accessed by the PLC logic program in exactly the same way as backplane I/O. This cyclic (repeated) I/O data exchange takes place asynchronously to the control logic scan and is repeated as quickly as possible. *Data exchange takes place every cycle for every slave in a master’s configuration.* At the most commonly used baud rate of 1,500kbit/s, data-exchange cycles are normally repeated many times during a single control-logic scan.

Figure 1-4 – Master/Slave Data Exchange

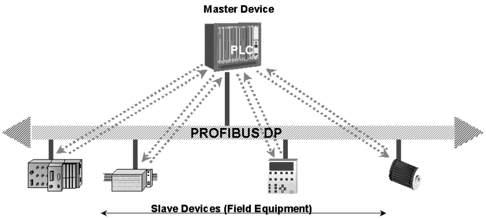
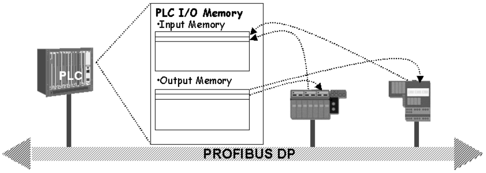
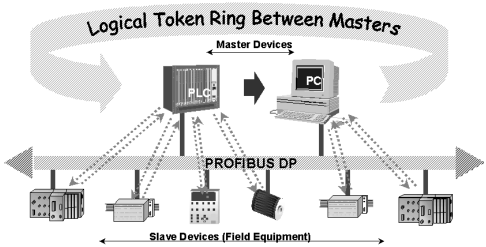


Figure 1-5 – Bus I/O Maps Into PLC Memory



Although 85 percent or more of PROFIBUS installations are single-master systems, multi-master systems, as illustrated in Figure 1-6, exist and work quite well. In such a system, each master is given control of the bus for a short time and during this time it exchanges I/O data with each of its assigned slaves. It then passes control to the next master on the bus, via a short message called a “token”, and that master exchanges I/O data with each of its slaves. *Only the master holding the token is allowed to initiate communication to its slaves*. Once the last master in the “logical token ring” has gone through its data-exchange cycle, it passes control back to the first master, and the overall operation starts again.

Figure 1-6 – Multi-Master/Slave Interactions


Logical Token Ring Between Masters
Master Devices
PLC
PC
PROFIBUS DP
Slave Devices (Field Equipment)

1.4 Device Diagnostic Reporting

The PROFIBUS protocol offers quite extensive diagnostic capabilities that device vendors can design into their products. PROFIBUS offers the capability to diagnose an operations problem all the way down to, for example, an overvoltage on an analog input or a broken wire on an output.

During a data-exchange cycle, a PROFIBUS slave device can indicate to the master that it has detected a diagnostic condition. In the next data-exchange cycle, the master fetches the diagnostic information from the slave.

A device can report diagnostic information in four different formats: standard diagnostics, device-related diagnostics, module-related diagnostics, and channel-related diagnostics. Sound complicated? It really isn't!

Any PROFIBUS master must save any diagnostic data from a slave in order for your control program to access it. Each master does it in a slightly different way, so you need to familiarize yourself with your particular master.

The *standard diagnostics* (six bytes) that every slave device is required to report contain information that is generally related to startup problems. For example, if the I/O configuration that was set up in the configuration tool does not match what the slave expects, it will report a "configuration fault." If you have configured a slave device in your configuration file, but the slave actually found on the bus at that address is different, the device will report a "parameterization fault." The six standard diagnostic bytes are used to report faults that are common across all slave devices.

A vendor can use the *device-related diagnostics* format to report information that may be specific to the particular device or application area, and that cannot be reported using the standard module-related or channel-related diagnostic formats. The format of this type of diagnostic information is defined by the vendor–its detailed structure is not covered in the PROFIBUS standard. Therefore, you must check the device's documentation to determine the exact format.

Module-related diagnostics are used to report diagnostics for a modular slave – that is, one that consists of an "intelligent" head module plus plug-in modules. This format gives the head module the capability to report that a particular plug-in module has a diagnostic. It does not tell us what the diagnostic is – just that a particular module has a problem. The format for module-related diagnostic information is defined in the PROFIBUS standard. This means that once a logic block is constructed to decode this type of diagnostic information, the decode logic will work for any device from any vendor that reports module-related diagnostics.

The last type of diagnostic block is used to report *channel-related diagnostics*. A device can use this format to report that an individual channel of a specific module has a problem – for example, short circuit, wire break, overvoltage, and so on. This makes it very easy to diagnose the problem right down to the wire level! The format for this type of diagnostic information is also defined in the PROFIBUS standard. The exact formats of these different diagnostic information blocks will be presented in Appendix A.

1.5 Fail-Safe Operation

Fail-safe, or fail-to-known-state, operation is an optional feature available for implementation in a slave device. This feature allows you to specify the states of slave outputs in the case of a bus failure, such as a master failure, a break in the bus, and so on. On such an occurrence, non-fail-safe devices usually clear their outputs to zero, while fail-safe devices set their outputs to states that you define during the configuration phase.

The procedure for specifying the output states is covered in Chapter 2, Section 2.3.2, describing device parameterization.

1.6 Optional Device Functionality

A device vendor has the option of including some additional PROFIBUS functionality in a device. These optional features include allowing the setting of the device address across the bus, supporting the output SYNC feature, and supporting the input FREEZE feature. If these optional features are supported, the vendor places keywords indicating so in the GSD file. For example, if a device supports the setting of its address across the bus, its GSD file will include an entry **Set_Slave_Add_supp = 1.**

There is a very rudimentary motion control capability, to *SYNC*hronize outputs, built into the standard DPV0 protocol. This SYNC functionality allows, for example, a group of drives on a conveyor to be ramped up to speed in a synchronous fashion. A SYNC operation can be initiated via an instruction in your control program (if the master supports it). This forces all the affected devices to immediately process the last output data sent from the master – for instance, a speed and direction command. The master sends this SYNC command out between I/O cycles. While giving the drives time to ramp up, your control program can write output data – for example, the next speed and direction step in the ramp-up/ramp-down – to the drives in the group over the next few cycles, The drives ignore the data until your control program causes another SYNC command to be sent between I/O cycles. When the ramp operation is complete, your control program can cause an UNSYNC command to be sent to put the drives in the group back into the normal mode of processing output data as it is received.

There is also a capability for your control program to cause a group of devices to FREEZE their input images that are being sent back to the master. This provides the capability to, for example, take a snapshot at a given time of several analog inputs distributed over several devices. Perhaps you are doing some calculations in your control program involving these analog input values, and you need all the values to be "frozen" at the same instant. A FREEZE operation takes place when an instruction in your control program initiates it, and these operations can be initiated repeatedly from your program. The devices continue to read the field inputs but do not update the image being sent back to the master except on a FREEZE command, which is sent by the master between I/O cycles. If you want the devices to be put back into the normal mode of continuously updating the inputs being sent back to the master, your program can initiate an UNFREEZE command.

In this chapter, we have attempted to give you a quick, general overview of the capabilities and features of PROFIBUS DPV0. Now let's get into the details of configuring the network and getting it up and running!