Acromag®

BusWorks™ 900PB Series
ProfiBus/RS485 Network I/O Modules

Technical Reference

INTRODUCTION TO PROFIBUS DP

ACROMAG INCORPORATED
30765 South Wixom Road
P.O. BOX 437
Wixom, MI 48393-7037 U.S.A.

Tel: (248) 624-1541
Fax: (248) 624-9234

8500-698-A02M000

TABLE OF CONTENTS

INTRODUCTION TO PROFIBUS DP

This information is provided as a service to our customers and to others interested in learning more about Profibus. Acromag assumes no responsibility for any errors that may occur in this document, and makes no commitment to update or keep this information current.

Be sure to visit Acromag on the web at **www.acromag.com.**

The following information describes the operation of ProfiBus DP as it relates to Acromag Series 900PB DP slave I/O modules. For more detailed information on ProfiBus, you may refer to the ProfiBus Trade Organization at the PTO website www.profibus.com.

Acromag manufactures a line of I/O modules that support Profibus DP over RS485. Feel free to visit our website at www.acromag.com to obtain the latest information about these and other Acromag products.

ABOUT PROFIBUS

Acromag Series 900PB modules utilize the popular ProfiBus DP FieldBus communication format. ProfiBus was created in 1989 by the German government in cooperation with several manufacturers of automation equipment. It is a messaging format specifically designed for high-speed serial I/O in factory and building automation applications. It is an open standard and is recognized as the fastest FieldBus in operation today. It is based on RS485 and the European EN50170 Electrical Specification. The DP suffix refers to “Decentralized Periphery”, which is used to describe distributed I/O devices connected via a fast serial data link with a central controller. To contrast, a programmable logic controller (PLC) normally has its input/output channels arranged centrally. By introducing a network bus between the main controller (master) and its I/O channels (slaves), we have *decentralized* the I/O.

ProfiBus is based on universal international standards and oriented to the OSI (Open System Interconnection) reference model per international standard ISO 7498. In this model, every layer handles precisely defined tasks. Layer 1 of this model is the physical layer and defines the physical transmission characteristics. Layer 2 is the data link layer and defines the bus access protocol. Layer 7 is the application layer and defines the application functions. ProfiBus DP uses only layers 1 & 2 of this model, plus the user interface. Layers 3 to 7 are not used.

A ProfiBus system uses a bus *master* to poll *slave* devices distributed in multi-drop fashion on an RS485 serial bus. A ProfiBus slave is any peripheral device (I/O transducer, valve, network drive, or other measuring device) which processes information and sends its output to the master. The slave forms a “passive station” on the network since it does not have bus access rights, and can only acknowledge received messages, or send response messages to the master upon request. It is important to note that all ProfiBus slaves have the same priority, and all network communication originates from the master. Acromag I/O modules form intelligent *slave* devices.

Acromag modules implement the ProfiBus protocol via an industry-standard SPC3 ASIC from Siemens. This ASIC acts like a RAM or UART chip to the internal microcontroller and completely handles the requirements of the protocol standard. The ASIC will transfer network data to and from the microcontroller and automatically provide the response to the bus according to the ProfiBus specification.

A ProfiBus master forms an “active station” on the network. ProfiBus DP defines two classes of masters. A class 1 master handles the normal communication or exchange of data with the slaves assigned to it. A class 2 master is a special device primarily used for commissioning slaves and for diagnostic purposes. Some masters may support both class 1 and class 2 functionality. Master-to-master communication is normally not permitted in Profibus, except in order to grant bus access rights to another master via the exchange of a token. However, master-to-master communication between two mono-master systems can be facilitated using a DP-DP gateway. Note that the exchange of bus access rights via this “token ring” only applies between masters on the bus.

A class 1 master device is normally a central programmable controller (PLC), or a PC running special software. The class 1 master sets the baud rate and the slave’s auto-detect this rate. The class 1 master handles the data exchange with the slaves assigned to it, and acts as the main controller for the exchange of I/O information with its distributed slaves, cyclically retrieving user I/O data according to a defined message cycle. A master can communicate actively with its assigned slaves, but only passively (upon request) with another class 2 master device.

The class 2 master is usually a configuration device, perhaps a laptop or programming console, and is provided for commissioning, maintenance, or diagnostic purposes. It acts like a “supervisory” master in that it can actively communicate with class 1 masters and their slaves, in addition to its own slaves, but usually only for the purpose of configuration, problem diagnosis, and data/parameter exchange. That is, class 2 masters may only briefly take over control of a slave. All exchanges between a class 2 master and class 1 master originate with the class 2 master.

ProfiBus DP normally operates using a cyclic transfer of data between master(s) and slave(s) on an RS485 network. That is, an assigned master periodically requests (polls) each node (slave) on the network. All data communication exchanges between a master and slave originate from the master device. Each slave device is assigned to one master and only that master may write output data to that slave. Other masters may read information from any slave, but can only write output data to their own assigned slaves.

Masters can address individual slaves, a defined group of slaves (multi-cast), or can broadcast a telegram to all connected slaves. Slaves will return a response to all telegrams addressed to them individually, but do not respond to broadcast or multi-cast telegrams from a master device. ProfiBus sends Broadcast and Multi-Cast messages as global control telegrams using address 127 and an optional group number for a targeted group of slaves.

Because ProfiBus uses a cyclic (periodic) polling mechanism between masters and slaves, it is also *deterministic*. That is, the behavior of a ProfiBus system can be reliably predicted over time. In fact, ProfiBus was designed to guarantee a deterministic response. To contrast, CAN and Ethernet are event-driven bus systems and consequently form non-deterministic systems.

The length (and timing) of the I/O data to be transferred from a single slave to a master is predefined in the slave's device data base or *GSD file*. The GSD files of each device connected via the network (slaves and class 1 masters only) are compiled into a master parameter record which contains parameterization and configuration data, an address allocation list, and the bus parameters for all connected stations. A master uses this information to set up communication with each slave during startup.

After a master receives its master parameter record, it is ready to begin exchanging data with its slaves. During startup, after a system reset, or upon return to power, a master will attempt to re-establish contact with all the slaves assigned to it before assuming the cyclic exchange of I/O data. Each slave must already have a unique valid address from 0-125 in order to communicate with the master. Any slave that has a default address of 126 will await the Set_Slave_Address command from a class 2 master before it can be parameterized. In attempting to establish communication, the master starts with the lowest address slave and ends with the highest address slave. A master will send parameterization and configuration telegrams to all of its assigned slaves (a slave may only be write-accessed by its assigned master, the master that parameterized and configured it during startup). The parameterization and configuration telegrams ensure that the functionality and configuration of a slave is known to the master. If an additional slave is added to the network bus and is not already accounted for in the master record, a new master record must be generated and a new configuration performed so that the master is informed of the status of the new device.

ProfiBus DP most often uses a single class 1 master device (*mono-master*), cyclically polling many distributed slaves. However, ProfiBus also allows for acyclic communication between class 2 masters and slaves, making more than one active station or master possible. A class 1 master will automatically detect the presence of a new active station connected to the network bus (a class 2 master). When the class 1 master completes its polling cycle, it will pass a "token" to the class 2 master granting it temporary access to the bus. Deterministic behavior is maintained because the class 2 master can only use the time allotted to it via the *gap time* specified. Although, mono-master operation is generally recommended, it is not mandatory. That is, a ProfiBus system may have more than one class 1 master, but master to master communication is not permitted, except for the granting of bus access rights via token exchange.

To illustrate the idea of communication between masters in a ProfiBus DP system, a class 1 master cyclically exchanges data with all of the slaves assigned to it, one at a time, according to its list of assigned slaves taken from the master record. At the end of this data cycle, additional time (gap time) is allotted to provide for acyclic communication between a class 2 master and the same slaves. During this time, the class 1 master will pass a token to the class 2 master granting it bus access rights. The class 2 master which currently holds the token has the opportunity to exchange data with all the slaves within a specific period of time called the *token half-time* or *token hold-time* (T_H). The class 2 master may then proceed to read data or diagnostic information from any of the slaves, and then at the completion of its cycle, it will pass the token back to the class 1 master.

Since there usually is not enough time during the gap to complete a full data exchange, this process of data retrieval by the class 2 master may continue over several cycles. At the end of record transfer, the class 2 master will clear the connection. Note however, that the class 2 master may only establish communication with the slaves during the gap time.

As stated earlier, it is possible for a class 2 master to temporarily take over control of a DP slave. During this time, the DP slave will stop its normal data exchange with its class 1 master. The class 1 master recognizes this and will proceed to cyclically request diagnostics from the slave, checking the Master Address field for as long as another valid address is present. After the class 2 master finishes its communication with the slave, it sets the Master Address field of the slave to invalid (255). This causes the class 1 master to attempt to regain control of the slave and it will reparameterize and reconfigure the slave before resuming data exchange with it.

Key Concepts

ProfiBus DP –

- Open standard based on EN 50170.
- Fastest Fieldbus standard to date with data rates up to 12MB.
- Plug & play operation.
- Up to 244 bytes of input/output data per message.
- Up to 126 stations may connect to the bus.
- Up to 32 stations per bus segment.

Class 1 Master –

- Central controller that exchanges I/O data with connected slaves.
- Determines the baud rate (slaves auto-detect this rate).
- Manages the token transfer between masters. Detects another master during the gap time.

Class 2 Master –

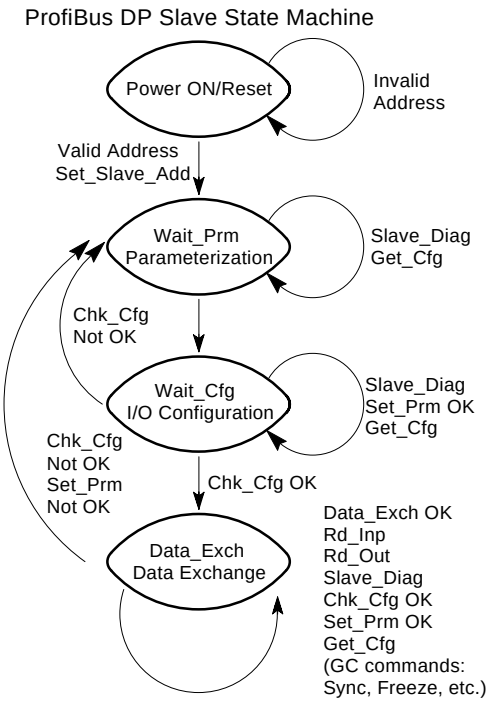
- Diagnostic, configuration, or startup tool.
- Can only control one slave at a time.
- Does not have write-access to the slave.
- Does not have a GSD file.

Slave -

- A passive station which can only respond per a master request and acknowledge messages. A slave has no bus control rights.
- The GSD file defines the slave for the master.

PROFIBUS DP SLAVE STATE MACHINE

The following state machine helps illustrates how ProfiBus DP operates with respect to the slaves.



Note the four main states: Power ON/Reset, Parameterization, I/O Configuration, and Data Exchange.

The master uses the following general telegram sequence during startup:

1. Request Diagnostics.
2. *Change Station Address (optional service, Class 2 Master only).*
3. Parameterize the Slaves.
4. Configure the Slaves.
5. Request Diagnostics again before data exchange to ensure that system startup was OK.
6. Data exchange.
7. *Global Control (optional).*

Power ON/Reset State

The power on/reset state is the initial state following power up for the DP slave. In this state, the slave may receive a telegram from a class 2 master to change its station address. A slave will be held in this state if it does not have a valid address from 0-125. After completion of its power-on initialization routine and if the slave has a valid station address, the slave will proceed to the Wait for Parameterization state.

Parameterization State

In this state, the DP slave awaits the parameterization telegram from the master which identifies the slave's master and the mode the slave is to operate in. A slave in this state will reject all other telegrams except a request telegram for diagnostics or configuration. After its parameters have been set, the slave will proceed to the I/O Configuration State.

I/O Configuration State

In this state, the slave awaits a configuration telegram that specifies the number of input and output bytes that are to be exchanged in each data telegram cycle with the slave. The configuration telegram also causes the slave to check the configuration which was sent against the stored configuration. A slave in this state will accept a request telegram for diagnostics or configuration, or a set parameters telegram.

Data Exchange State

After parameterization and configuration have been accomplished, the slave cyclically exchanges I/O data with the master. This is a cyclic transfer of I/O data and possible diagnostic information.

Fail Safe Operation

A ProfiBus master runs in two modes: Operate and Clear. With respect to the master of a ProfiBus DP system, the term fail-safe simply refers to whether the class 1 master sends 0 length data, or data set to 0, when it is in Clear Mode. With respect to a DP slave device, the term fail-safe refers to whether the slave will process output telegrams with zero length data, or not. Whether the combined master/slave system is considered fail-safe depends on the actions taken by the slaves if the master fails, or if the master switches to Clear Mode. Ideally, the failure of a master should not cause errors in any of its slaves and the slave outputs should go to a predictable (defined) state. Using a fail-safe mode, the slave outputs can automatically switch to a fail-safe state in the event of master failure, or when the master switches to Clear Mode.

A slave may assume a fail-safe state if its watchdog time expires without having received a message from its assigned master. Normally this timer is reset every time the master talks to the slave. If this time expires, this means the master has not communicated with the slave recently, and the slave is not being controlled. The slave will then leave the data exchange mode and its outputs will go to a pre-defined state (either their reset state, or another user-defined state). This state is usually set via user-defined parameters of the parameterization telegram and its GSD file, or sometimes via hardware switches on the slave. Some slaves may provide parameters or switches that also allow the slave outputs to retain their last state, but this is not considered fail-safe.

A slave may also assume a fail-safe state if its master switches from Operate Mode to Clear Mode. With normal operation in Data_Exchange mode, a class 1 master is in Operate Mode and cyclically exchanges I/O data with its assigned slaves. The class 1 master may use a global control telegram to inform the slaves that it is switching from Operate Mode to Clear Mode. A master may elect to switch to Clear Mode while it is bringing slaves online and not all slaves have been parameterized and configured yet. It may also switch to Clear Mode as a result of a run/stop switch on the master. In the Clear State, the master may attempt to parameterize and configure the remaining slaves assigned to it in an effort to reinitiate data exchange, while it continues data exchange with the other slaves (they will be receiving output data of 0, or output data of zero length). Operate mode does not resume until all slaves are online and exchanging data, or until the master is told to resume operation via a run/stop switch or under program control. Further, some masters may go to Clear Mode if a slave is disabled, rather than continue to control a partial system (this response may be specified as a parameter in the master's GSD file and parameterization telegram, via a mechanical switch, or as part of its master program).

When a master switches to Clear Mode, it sends a global control telegram to all slaves with the first data byte (octet 1) = 2 and the second data byte (octet 2) = 0. In the next data cycle, the master sends data telegrams to all stations with either the output data equal to 0, or the output data length equal to 0 (i.e. only the telegram header and no data). If the slave GSD file contains “Fail_Safe = 0”, the master sends output telegrams with the data set to 0 in Clear Mode. However, if the slave GSD file contains “Fail_Safe = 1” (supports Fail-Safe Mode), the master will send output telegrams with a data length of 0 in Clear Mode. . Slaves that do not support fail-safe mode do not process data telegrams with no data. Some older masters do not make this distinction of fail-safe mode and will send data telegrams with the output data set to 0 in Clear Mode. This will force all slave outputs to go to 0 in Clear Mode and this may not be desirable for some critical control applications.

By the master sending output telegrams with no data in Clear Mode, a fail-safe slave has the option of either setting all outputs to 0, retaining the last output state (though this response is not fail-safe), or going to a pre-defined default “fail-safe” state. This state may be defined in the GSD file and included in the user-parameters portion of the parameterization telegram, or it may be set via switches at the slave. The slave will stay in this state until it receives a global control broadcast telegram from the master telling it the master is returning to Operate Mode and it receives an output telegram with the correct output data length, whereupon it updates its outputs normally as part of data exchange mode.

To summarize, fail-safe mode in ProfiBus DP simply refers to whether the master sends output data messages of zero length in Clear Mode or not, and whether a slave is able to process output messages of zero length. The actions taken by the slave in response to these messages is optional and specific to the slave implementation. Note that a “fail-safe” slave may not actually act in a fail-safe manner (for example, it may retain the last state prior to Clear Mode). In any case, for PTO compliance, a fail-safe slave must at least give the option of clearing the outputs if the master fails or switches to Clear Mode.

Watchdog

A slave may assume a defined state if its watchdog time expires without having received a message from its assigned master. Normally this timer is reset by the slave every time the master talks to the slave. If this time expires, this means the master has not communicated with the slave recently, and the slave is not being controlled. A communication error is detected by the slave and reported with a diagnostic telegram (see function codes). The slave will then leave the data exchange mode and its outputs will go to a predefined state (either the reset/clear state, or a user-defined state) and await reparameterization and reconfiguration by the master. The timeout state is clear by default, or it may be set via user-defined parameters of the parameterization telegram and GSD file, or sometimes via hardware switches on the slave. Some slaves may provide parameters or switches that also allow the slave outputs to retain their last state, but this is not considered fail-safe.

During parameterization, a master sets up the communication and monitoring times for the slave including the watchdog time (T_{WD} set by Watchdog Factors in DU bytes 2 & 3 of the Set_Prm Telegram). The slave ASIC implements a watchdog function where the slave will monitor the bus communications with the master over time, and in the event of master failure (timeout), the slave outputs go to a defined state. If the watchdog timer is not retriggered by the slave station via bus communication with the master within the time specified, then the slave will set its outputs to a defined state and return to the Wait_Prm state (Wait For Parameterization). In setting watchdog time T_{WD}, you need to consider the bus cycle time, plus a safety factor for repeated telegrams (usually 25% minimum).

Before we examine slave operation in detail, we need to get a little background information on a device's GSD file and how the software is used to build a ProfiBus system.

GSD FILES

The GSD file is an electronic device data sheet or device data base file that identifies the ProfiBus device. All ProfiBus devices (class 1 masters and slaves) have their own GSD files. The GSD file is the fundamental building block for the master parameter record. Use of the GSD file by a ProfiBus configuration tool permits plug & play interoperability among different devices from different manufacturers. This file does not reside within the device itself, but usually on a separate disk/drive. It is an ASCII text file that contains device-specific data, such as, vendor identification information, supported baud rates, supported message length, number of input/output data, meaning of diagnostic messages, timing information, plus options and features supported, data formats, and available I/O signals. For modular ProfiBus systems, a GSD file may contain several configurations (one for each I/O module), one of which will be found valid during startup.

A GSD file is named by combining a vendor name identifier with the device's ident_ number. For example, "ACRO06F3.GSD" is used for the Acromag 981PB-1012 device. The suffix ".GSD" denotes a language independent GSD file. A ".GSE" file would specify an English file, ".GSF" for French, ".GSS" for Spanish, ".GSI" for Italian, and ".GSG" for German.

The GSD file begins with the specifier "#Profibus_DP". In the body of the file, the parameters are specified as parameters of a keyword (as in "keyword = parameter", see below). Comment lines begin with a semicolon. Case is not significant and the sequence of parameters is not important. Lines are limited to 80 characters, but may be continued by placing a backslash character "\" at the end of the line to be continued. A GSD file is divided into sections as follows:

GSD FILES

GSD General Specifications	Keyword	Keyword
This section contains information on vendor and device names, hardware and software revisions, ident_number, supported baud rates, reaction time intervals at supported baud rates for monitoring times, and optional signal support at the bus connector.	Vendor_name	187.5_supp
	Model_Name	500_supp
	Revision	1.5M_supp
	Ident_Number	MaxTsdr_9.6
	Protocol_Ident	MaxTsdr_19.2
	Station_type	MaxTsdr_93.75
	FMS_Support	MaxTsdr_187.5
	Hardware_Release	MaxTsdr_500
	Software_Release	MaxTsdr_1.5M
	9.6_supp	Redundancy
	19.2_supp	Repeater_Ctrl_Sig
	93.75_supp	24V_Pins

GSD Slave Specifications	Keyword	Keyword
This section contains all slave-related specifications, such as the number and type of I/O channels, specification of diagnostic text, auto-baud support, alternate mode support and information on available modules with modular devices.	Freeze_Mode_supp	Max_Input_Len
	Sync_Mode_supp	Max_Output_Len
	Auto_Baud_supp	Max_Data_Len
	Set_Slave_Add_supp	Unit_Diag_Bit
	User_Prm_Data_Len	Diag_Text
	User_Prm_Data	Unit_Diag_Area
	Min_Slave_Intervall	Module
	Modular_Station	Channel_Diag
	Max_Module	

Master Specifications (Master Devices Only): This section contains all master-related parameters, such as: the maximum number of slaves that can be connected, or upload/download options. This section is not present for slave devices and not covered here.

The GSD files of all connected devices are compiled together to form the master parameter record. The master parameter record contains the parameterization and configuration data taken from the all the GSD files, and includes an address allocation list, plus the bus parameters for all the connected slaves. During startup, a master will use this information to set up communication with each of its assigned slaves prior to exchanging actual I/O data with them.

The ProfiBus Trade Organization offers an easy to use, menu-driven editor which can be used to prepare GSD files. This GSD-Editor also contains a GSD-Checker which guarantees the conformance of the GSD file to the ProfiBus standard. The format of GSD files is precisely specified in the EN50170 standard and described in ProfiBus Guideline 2.041.

The ProfiBus Trade Organization also maintains a library of GSD files for all certified slave devices. You can access these files via the internet at http://www.profibus.com.

To give you an idea of what a GSD file might look like, consider the following partial GSD file taken from Acromag Model 983PB-2012.

```
#Profibus_DP
GSD_Revision          = 1               ; Version of the GSD file
Vendor_Name           = “Acromag, Inc.” ; Vendor name
Model_Name            = “983PB-2012”    ; Product name
Revision              = “A”
Ident_Number          = 0x06F1          ; Ident Number
Protocol_Ident        = 0               ; ProfiBus DP Only (1-DP/FMS)
Station_Type          = 0               ; Type of device (Slave)
Hardware_Release      = “A”             ; Hardware version of the device
Software_Release      = “A”             ; Software version of the device
;
9.6_supp              = 1               ; 9600bps Supported
19.2_supp             = 1
93.75_supp            = 1
187.5_supp            = 1
500_supp              = 1
1.5M_supp             = 1
3M_supp               = 1
6M_supp               = 1
12M_supp              = 1
MaxTsdr_9.6           = 60              ; Maximum response time
MaxTsdr_19.2          = 60              ; at different baud rates.
MaxTsdr_93.75         = 60
MaxTsdr_187.5         = 60
MaxTsdr_500           = 100
MaxTsdr_1.5M          = 150
MaxTsdr_3M            = 250
MaxTsdr_6M            = 450
MaxTsdr_12M           = 800
;
Redundancy            = 0               ; Redundancy Not Supported
Repeater_Ctrl_Sig     = 2               ; Includes RTS Support w/TTL
Implementation_Type   = “SPC3”          ; Uses Siemens SPC3 ASIC
24V_Pins              = 0               ; Does Not Include 24V
Fail_Safe             = 1               ; Supports Fail-Safe Mode
Freeze_Mode_supp      = 1               ; Supports FREEZE
Sync_Mode_supp        = 1               ; Supports SYNC
Auto_Baud_supp        = 1               ; Includes Auto Baud Detection
Set_Slave_Add_supp    = 1               ; Addr can be set via ProfiBus
User_Prm_Data_Len     = 3
User_Prm_Data         = 0x00,0x00,0x00  ; Module Specific Parameters
; 00H = Set outputs to 0
; 01H = Maintain Last Output Values
; 02H = Set output to user-defined values in bytes 1 and 2
; Byte 1 is the lower byte of user-defined output data which is outputs 0 to 7
; Byte 2 is the upper byte of user-defined output data which is outputs 8 to 11
;
```

```
Slave_Family            = 3
Min_Slave_Interval      = 1             ; Min_Slave_Interval is 100us
Modular_Station         = 0             ; 0-compact, 1-modular
Max_Diag_Data_Len       = 6             ; No User Diagnostics are Sent
; I/O Byte
Module                  = "12 CH DIG I/O:xxxx1198 76543210" 0x11,0x21
EndModule
;
```

REQUIRED SOFTWARE

Leading PLC manufacturers offer configuration software for their products that make it easy to generate the master parameter record for their devices. This software is generally master-specific and depends on the design of the master device. In order to setup a ProfiBus system, you will need a software configuration tool, such as Allen Bradley's Plug and Play software, or the Siemens COM ProfiBus package. This software configures the active stations and tells them what devices are present on the bus and how much data it needs to exchange with them.

Recall that the master parameter record contains all the required data for the bus system, including an address allocation list and the bus parameters of the connected slaves. The address allocation list assigns each remote I/O byte a unique address in the I/O space of the master's I/O space.

As an alternative method to PLC specific software, Siemens offers an easy to use configuration tool called COM PROFIBUS that can be run on any PC (offline version). An online version of COM PROFIBUS requires an interface card.

The configuration software utilizes the GSD files (device master data) of the connected slaves to create the master parameter record. This record is typically transferred or downloaded to the class 1 master from floppy disk, dual port RAM, or Flash EPROM. Typically, if you need to add another slave to an existing system, you simply upload the current master parameter record, add the new parameterization data for the slave (imported from the GSD file), then download the new master record to the master again. Since an active bus station automatically detects a new active bus station, you can then perform a reset, and the bus system will reconfigure itself.

In addition to the slave-specific data which the configuration tool gathers from the GSD file, you may also have to provide the following bus and protocol-related data via the configuration tool:

- The Protocol Used (DP, FMS, or mixed network).
- The transmission baud rate (the tool checks to determine whether all stations or slaves actually support this speed).
- The GAP factor (number of bus passes after which an additional active station is searched for). A master automatically detects the connection of other masters via a periodic search driven by this factor.
- The Highest Station Address (HSA).
- The Watchdog Time.
- Within a PLC, the CPU type and type of addressing used.

Most configuration tools also allow you to permanently store user parameters and configuration bytes which the master will automatically send to the slave during system startup.

TYPES OF TRANSMISSION

ProfiBus DP uses two types of transmission services in sending message telegrams that are defined in Layer 2 (The Data Link Layer) of the ISO/OSI model and summarized below:

SRD (Send and Request Data with acknowledge)

With SRD, data is sent and received in one telegram cycle. That is, the master sends output data to the slave and receives input data from the slave in its response (if applicable) within a specified period of time. The important thing to remember about this service, is that a master may send output data to a slave and request input data from the slave in its response, all in a single telegram cycle. This is the transmission service most often used in ProfiBus DP that makes the data exchange very efficient for mixed I/O devices.

SDN (Send Data with No acknowledge)

This service is used when a message must be sent simultaneously to a group of slaves (multi-cast), or all slaves (broadcast). Slaves do not respond or acknowledge broadcast or multi-cast messages.

A third type of transmission service used in ProfiBus FMS is SDA (Send Data with Acknowledge), with data sent to a master or slave and a short acknowledgement sent in response. This is not used in ProfiBus DP and is not covered here.

PROFIBUS DP DATA CHARACTER FORMAT

All ProfiBus characters are comprised of 11 bits (1 start bit + 8 data bits + 1 even parity bit + 1 stop bit). ProfiBus DP exchanges data in NRZ code (Non Return to Zero). That is, the signal form of binary “0” or “1” does not change during the duration of the bit. If nothing is being transmitted, the idle state potential on the line is “1”. A start bit causes the line to go to “0”.

ProfiBus NRZ-Coded Character Frame (Even Parity)

Start	D0	D1	D2	D3	D4	D5	D6	D7	Parity	Stop
“0”	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	**even**	**“1”**
←	*LSB*	←	←	←	←	←	←	*MSB*	←	←

This character frame applies to all data/character bytes, including the telegram header bytes. When messages are transmitted on ProfiBus serial networks, each character or data byte is sent in the order of least significant bit (lsb) to most significant bit (msb), as shown above. For word transfer (more than 1 byte), the high byte is transmitted first, followed by the low byte (Big-Endian/Motorola format).

ProfiBus Data Error Checking

ProfiBus networks employ the *even parity* method of data error checking which controls how the parity bit of a data frame is set.

With even parity checking, the number of 1 bits in the data portion of each character frame is counted. Each character contains 8 bits. The parity bit will then be set to a 0 or 1, as required in order to result in an even total number of 1 bits. For example, if a character frame contains the following eight data bits: 1100 0011, then since the total number of 1 bits is 4 (already an even number), the frame's parity bit will be set to 0 for even parity.

When a message is transmitted, the parity bit is calculated and applied to the frame of each character transmitted. The receiving device counts the quantity of 1 bits in the data portion of the frame and sets an error flag if the count differs from that sent. As such, parity checking can only detect an error if an odd number of bits are picked up, or dropped off, from a character frame during transmission. For example, with even parity, if two 1 bits are dropped from a character, the result is still an even count of 1 bits and no parity error will be detected.

PROFIBUS TELEGRAM (MESSAGE) STRUCTURE

A ProfiBus telegram may contain up to 256 bytes--up to 244 bytes of data per node per message, plus 11 bytes of overhead. This overhead is referred to as the *Telegram Header*. All telegram headers are 11 bytes, except for Data_Exchange telegrams which have 9 bytes of header information (the DSAP and SSAP bytes are dropped). Note that 12 bytes is a lot of overhead for a single message and this makes ProfiBus less efficient for small amounts of data. However, since up to 244 bytes of data may occur per message, and since the output data is sent and the input data received in a single telegram cycle, this makes ProfiBus more efficient when large amounts of data must be transferred. Note that an idle state of at least 33Tbits (sync-time in bit time) must be present before every request telegram to be sent, and all data is transferred without gaps between individual characters. All data exchanges between a Master and Slave are handled in the telegram header using Service Access Points (SAP's). ProfiBus DP uses SAP's 54 to 62, plus the default SAP (Data_Exchange).

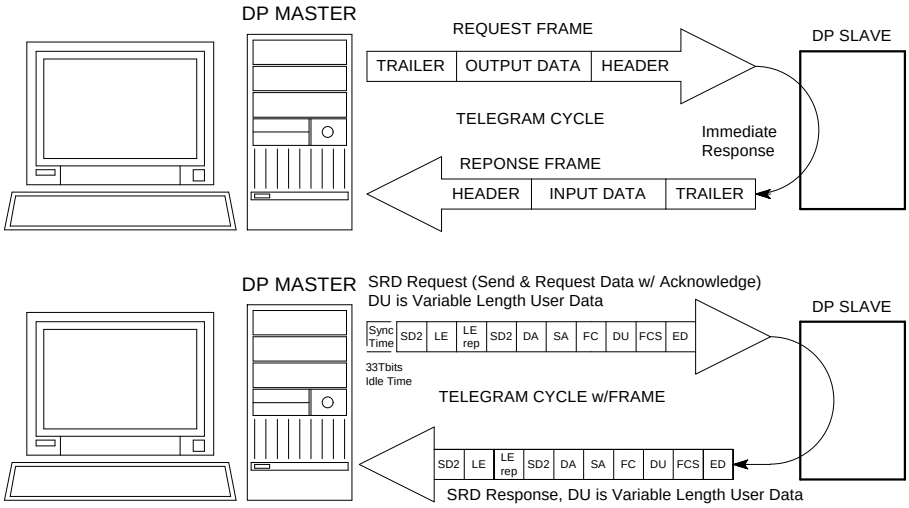
Telegram Header Data and Frame

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
1b	1b	1b	1b	1b	1b	1b	1b	1b	var	1b	1b

ProfiBus DP Telegram Header Abbreviations and Frame Bytes

SD	1 byte	Start Delimiter (used to distinguish telegram format).
LE	1 byte	Net Data Length (DU) + DA + SA + FC + DSAP + SSAP.
LEr	1 byte	Length *repeated.*
DA	1 byte	Destination Address– Where this message goes to.
SA	1 byte	Source Address – Where this message came from. The address of the sending station.
FC	1 byte	Function Code (FC=Type/Priority of this message). Used to identify the type of telegram, such as request, acknowledgement, or response telegrams (FC=13 signals diagnostic data). See below.
DSAP	1 byte	Destination Service Access Point (COM port of receiver). The destination station uses this to determine which service is to be executed.
SSAP	1 byte	Source Service Access Point (COM port of sender).
DU	1 to 32b (or 1-244b)	Data Units/ Net Data from 1 to 244 bytes.
FCS	1 byte	Frame Checking Sequence (ASIC addition of the bytes within the specified length).
ED	1 byte	End Delimiter (always 16H).

The following picture depicts the telegram sequence between a class 1 master and DP slave. The paragraphs that follow describe each telegram frame byte in greater detail.



Start Delimiter (SD)

The Start Delimiter identifies the beginning of a telegram and its general format. ProfiBus DP uses four types of Start Delimiters (SD) for request and response telegrams, plus a fifth response for a short acknowledgement as shown below. Note that the short acknowledge response does not use a start delimiter. Also, a telegram response does not have to use the same Start Delimiter as the request telegram.

Telegram Format	Value	Data Field Length
SD1	10H	0 bytes (No Data Field)
SD2	68H	1 to 32 bytes (or up to 244)
SD3	A2H	8 bytes fixed.
SD4	DCH	0 bytes (No Data Field)
SC	E5H	0 bytes (No Data Field), Short Acknowledge.

Start Delimiter SD1 (SD=10H) = **Request_FDL_Status**

Telegram with fixed information section and no data field…

SD1	DA	SA	FC	FCS	ED
10H	xx	Xx	x	x	16H

An active station sends this telegram to look for new active stations on the bus after the GAP time has expired.

Start Delimiter SD2 (SD=68H) = *Telegram w/ variable DU.*

Telegram with variable information section and data field length…

SD2	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	X	x	68H	xx	xx	x	3CH	3EH	x..x	X	16H
				←----------*VAR LENGTH*----------→							

Data telegram with variable data length. Used in SRD service (Send and Request Data with acknowledge) which allows output data to be sent and input data to be received in one telegram cycle.

Start Delimiter SD3 (SD=A2H) = *Telegram w/ fixed DU.*

Telegram with fixed information section and data field length…

SD3	DA	SA	FC	DU	FCS	ED
A2H	xx	xx	x	x..x	x	16H

This delimiter is used for data telegrams with fixed data length (the DU data is always 8 bytes long).

Start Delimiter SD4 (SD=DCH) = *Token Telegram*

Master-to-master token telegram…

SD4	DA	SA	ED
DCH	xx	xx	16H

This delimiter is used between 2 active bus stations to grant bus access rights.

No Start Delimiter

No Start Delimiter - Short Acknowledgement Telegram…

SC	The short acknowledgement frame SC is a 1 byte message that can be used to positively acknowledge an SDA request (ProfiBus FMS only), or negatively acknowledge an SRD request.
E5H	

Length of Telegram (LE & LEr)

This byte specifies the length of a telegram with variable data length (i.e. SD2 Telegrams) from the DA byte to the end DU byte (range is DU+5b to 249). Note that the length of the DU is generally limited to 32 bytes, but the standard allows for lengths up to 244 bytes. LE is repeated in the LEr field for redundant data protection.

Destination Address & Source Address (DA & SA, 00H..7FH)

The master device addresses a specific slave device by placing the 8-bit slave address in the DA address field of the telegram (Destination Address). It includes its own address in the SA address field (Source Address). Valid addresses are from 0-127 (00H..7FH). Address 126 is reserved for commissioning purposes and may not be used to exchange user data. Address 127 is reserved as the broadcast address, which all slave devices on a network recognize. When the slave responds, it will place its own address in the source address field of its response to let the master know which slave is responding, and it will place its assigned master's address in the destination address field of its respone. Recall that a slave does not issue a response to broadcast messages (address 127).

Note that the inclusion of a DSAP or SSAP entry in a request or response telegram is identified by setting the highest bit in the address byte of the DA (Destination Address) or SA (Source Address), respectively. This may look like an address greater than 127, but only the lower 7 bits of the DA and SA contain the actual address.

Function Code or Frame Control (FC)

The Function Code (FC) or Frame Control field specifies the type of telegram (request, response, acknowledgement), type of station (passive or active/slave or master), priority, and telegram acknowledgement (successful or unsuccessful) as follows:

ProfiBus DP Function Codes for Request Telegrams

FC Code	Function (The MSB in FC = 1)
4	SDN low (Send Data with No acknowledge)
6	SDN high (Send Data with No acknowledge)
7	Reserved/Request Diagnostic Data
9	Request FDL Status With Reply
12	SRD low (Send and Request Data with acknowledge)
13	SRD high (Send and Request Data with acknowledge)
14	Request ID With Reply
15	Request LSAP Status With Reply

ProfiBus DP Function Codes for Acknowledgement Telegrams

FC Code	Function (The MSB in FC = 0)
0	ACK Positive
1	ACK Negative (FDL/FMA1/2 user error UE, interface error)
2	ACK Negative (No resource/memory space for Send Data (RR).
3	ACK Negative (No service activated (RS), SAP not activated).
8	Response FDL/FMA ½ Data low and Send Data OK)
9	ACK Negative (No response FDL/FMA1/2 Data & Send Data OK).
10	Response FDL Data High and Send Data OK.
12	Response FDL Data Low, No resource for Send Data.
13	Response FDL Data High Resource For Send Data.

Service Access Points (SSAP & DSAP)

Data exchanges are handled in the telegram header using Service Access Points (SAP's). The SAP tells what data is to be transmitted or which function is to be performed. Only telegrams that include data fields use DSAP & SSAP bytes (i.e. SD2 & SD3 telegrams). Recall that SRD transmission combines an output message with an input response in a single telegram cycle. The telegram header contains an SSAP (Source Service Access Point) and/or DSAP (Destination Service Access Point) that indicates the service(s) to be executed. One exception is the cyclical Data_Exchange telegram which is performed with the default SAP (SSAP or DSAP is not provided in its header). Additionally, some telegrams may only provide a DSAP or SSAP, but not both.

The inclusion of a DSAP or SSAP entry in a request or response telegram is identified by setting the highest bit in the address byte of the DA (Destination Address) or SA (Source Address), respectively. Based on the detected SAP's, each station is able to recognize which data has been requested and which response data is to be supplied. ProfiBus DP uses SAP's 54 to 62 listed below, plus the default SAP.

SAP	SERVICE
Default SAP=0	Cyclical Data Exchange (Write_Read_Data)
SAP54	*Master-to-Master SAP (M-M Communication)*
SAP55	Change Station Address (Set_Slave_Add)
SAP56	Read Inputs (Rd_Inp)
SAP57	Read Outputs (Rd_Outp)
SAP58	Control Commands to a DP Slave (Global_Control)
SAP59	Read Configuration Data (Get_Cfg)
SAP60	Read Diagnostic Data (Slave_Diagnosis)
SAP61	Send Parameterization Data (Set_Prm)
SAP62	Check Configuration Data (Chk_Cfg)

SAP55 is optional and may be disabled if the slave does not provide non-volatile storage memory for the station address. Note that SAP's 56, 57, and 58 are not enabled until the DP slave assumes the Data_Exchange state. SAP's 59, 60, 61, and 62 are always enabled.

Note that the DSAP & SSAP entries in a request telegram are also included in the response telegram, where DA + SA + DSAP + SSAP in the response message corresponds to SA + DA + SSAP + DSAP in the request telegram (content position flips).

Data Unit (DU)

This field contains the data for the station at DA (request data), or the data for the station at SA (response data). DU is generally limited to 32 bytes, but the standard allows for lengths up to 244 bytes (assuming 11 bytes of header information for 255 bytes total).

Frame Check Sequence (FCS)

This field contains the Frame Check Sequence or telegram checksum (00H..FFH). It is simply the sum of the ASCII bytes of information from DA to DU modulus 256. Checksum = (DA + SA + FC + DU) mod 256. This is simply the bytes added together and divided by FFH (255). This is an integrated function that is normally performed by the ProfiBus ASIC.

End Delimiter (ED)

This byte identifies the end of a ProfiBus telegram and has a fixed value of 16H.

DP COMMAND FUNCTIONS

The following functions are implemented in DP slaves and class 1 masters (see SAP descriptions above). There are only 8 mandatory slave functions, plus the optional Set_Slave_Add function (the slave address can usually be set via external switches). All commands noted below are optional for class 2 master devices.

Functions	DP Slave	Class 1 Master
Data_Exchange	√	√
Rd_Inp	√	
Rd_Outp	√	
Slave_Diag	√	√
Set_Prm	√	√
Chk_Cfg	√	√
Get_Cfg	√	
Global_Control	√	√
Set_Slave_Address	√ **(Optional)**	
Get_Master_Diag		√
Start_Seq		√ (Optional)
Download		√ (Optional)
Upload		√ (Optional)
End_Seq		√ (Optional)
Act_Para_Brct		√ (Optional)
Act_Para		√ (Optional)

Function Status

The corresponding function acknowledgements will contain a status parameter that may be tested to verify the success of a function request. Possible status values are shown in the following table:

Status Value	Description
OK	Acknowledgement positive.
IV	Invalid parameters in request.
NO	Service in this state not possible.
DS	Local FDL/PHY entity is not in logical token ring or disconnected from line.
NA	Negative Acknowledge – No reaction from remote station.
RS	Service or remote-address at remote-LSAP or remote-LSAT not activated: Remote station is no DP-Station, Remote station is not ready for these functions, Remote station is associated with another requestor, Optional service not available.
RR	Resources of the remote-FDL entity not sufficient or not available.
UE	Remote-DDLM/FDL interface error.
NR	No response data.
TO	Function timeout expired.
FE	Format error in request frame.
RE	Format error in response frame.
LE	Data block length too large (Upload/Download)
NI	Function not implemented.
EA	Area too large (Upload/Download)

Function Status

Status Value	Description
AD	Access denied.
IP	Invalid parameter.
SC	Sequence conflict.
SE	Sequence error.
NE	Area Non-existent.
DI	Data incomplete.
NC	Master parameter set not compatible.

Refer to EN 50170 for more information on function status codes.

OPERATING STATES AND APPLICABLE FUNCTIONS

Recall that a ProfiBus DP slave must pass through 3 other states prior to the Data Exchange state: Power On/Reset, Parameterization, and I/O Configuration. Each DP slave state and the related function telegrams are described in the following paragraphs. Refer to EN 50170 Volume 2 for a more detailed explanation.

Initial Power ON/Reset

In the initial state following power up, the slave initializes itself and automatically detects the correct baud rate for communication. If a valid address from 0-125 has been set, the slave will pass to the parameterization state. However, if the slave's address is set to 126 (the default commissioning address), then the slave will await a "Set_Slave_Address" telegram from a class 2 master to change its station address before proceeding to the parameterization state. A class 2 master can use SAP 55 (Set_Slave_Add) in a telegram header to change the address of a slave following power-up, but only if the slave's Set_Slave_Address lock bit is clear and it has the default address of 126. Recall that stations whose address cannot be set externally will have a default address of 126 and this address can only be changed with a class 2 master device.

The address must be set to a valid address from 0-125 in order for the slave to pass to the parameterization state.

So that the slave address does not have to be reassigned after power-up, this address is normally stored in non-volatile memory and uploaded upon initialization, or it may be loaded from switches on the unit. If there are several similar stations whose address cannot be set externally and have a default address of 126, be sure to connect them to the network one at a time in order to set their addresses.

The important point about initialization is that the slave address must be set to a valid address from 0-125 (either through external switches or via the Set_Slave_Address command from a class 2 master) in order for the slave to pass from the initialization state to the parameterization state.

From the factory, Acromag modules have a default address of 126. However, address 126 may not be used for data exchange, as it is reserved for the purpose of commissioning. Acromag modules have external switches for setting the slave address, but also support address changes via the Set_Slave_Address command when the external switches are preset to 126 and the internal EEPROM address is also 126. If the unit is instead powered-up with the external address switches set from 0-125, the Set_Slave_Address request is refused with an RS error message. If the unit is powered-up with the external address switches set to 126, then the unit will retrieve its address from the internal EEPROM memory.

The address stored in EEPROM can only be modified via the Set_Slave_Address command. Additional changes to the internal EEPROM address can also be locked out. If the internal EEPROM address is also set to 126 (and the external switches are at 126), then the unit will await the Set_Slave_Address command to change its EEPROM address to a valid address from 0-125, before proceeding to the parameterization state. Note that if the address lock bit has been set (locked), then further changes via Set_Slave_Address are effectively locked out. In this case, you may set the external switches to 255 (FFH) and re-power the unit, this will clear the lock bit and restore the internal EEPROM address to 126.

Set_Slave_Add Telegram (SAP 55)

Before commissioning, a station must have already been assigned a unique station address from 0-125. This is usually accomplished via hardware switches on the device. In the event that the device does not provide switches for setting its address (or its address switches are set to 126), the address can be set via this bus command from a class 2 master.

The Set_Slave_Add telegram is used by a class 2 master to change a slave's address following power-up when its address cannot otherwise be set via hardware switches (and it has a default address of 126). This command also allows further address changes over the bus to be locked out. Note that this telegram transmits the Ident_Number for security reasons. If the ident_number does not match that of the targeted slave, the address will not be changed. Additionally, if the lock bit has already been set, then again, the address will not be changed.

DP slaves that have not been assigned an address typically have a default address of 126, which is reserved for commissioning via class 2 master devices. No class 1 master is allowed to use address 126 to communicate. Further, only one slave with address 126 is permitted to be connected to the network at one time and this address may not be used to exchange I/O data.

With respect to Acromag modules which support address changes via external switches and optionally via this command, the Set_Slave_Address command is used to modify the internal (EEPROM) address setting. The internal address setting will determine the slave address if the unit is powered-up with the external switches set to 126. Otherwise, if the external switches are set from 0-125, then the switch settings determine the slave address, not the value stored in EEPROM. However, in order for the Set_Slave_Address command to change the slave address, both the internal EEPROM address and external switches must be set to address 126 prior to power-up. The Set-Slave_Address command will be rejected with the RS error code for any other conditions.

Set_Slave_Add Telegram

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	09H	09H	68H	XX	XX	X	37H (55)	3EH (62)	X..	X	16H

DU Byte 1 (New_Slave_Add, 0-125)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

DU Byte 2 (Ident_Number for security check, High Byte)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

DU Byte 3 (Ident_Number for security check, Low Byte)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

The identification number (Ident_Number) is used to establish a reference between a slave and its corresponding GSD file. This number is assigned by the ProfiBus Trade Organization and cannot be changed. It is used here for security purposes--if the ident_number does not match that of the slave, the address will not be changed. Note that only class 2 masters do not require an ident_number.

DU Byte 4 (LOCK - Enable/Disable Further Address Changes)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
00H = False/Unlocked, Change of address possible later. 01H = True/Lock, No further address changes possible[1].							

[1]To unlock an Acromag module for address changes, set the external switches to 255 (FFH) prior to applying power and this will clear the lock bit and restore the internal EEPROM setting to 126. Next, simply set the switches as desired from 0-125, or to 126 if you intend to use the Set-Slave_Address command, and re-power the unit.

The parameter Status can be tested to indicate if the Set_Slave_Add request frame was sent successfully and accepted by the slave. However, the acknowledgement does not indicate whether the new values were actually accepted by the slave. A master can check the correct execution of this function by using Slave_Diag with the new DP-slave address and then testing Status for the possible values of: OK, DS, NA, RS, RR, UE, and RE.

After completion of the power-on initialization routine and after the slave's address has been set to a valid address from 0-125, the slave will proceed to the Wait_Prm state (Wait for Parameterization).

Parameterization

After the slave completes its power-on initialization routine and its address has been set to a valid address from 0-125, the slave will proceed to await the Parameterization Telegram (Set_Prm) from the master which serves to identify the master for the slave and specify the slave's operating mode. The slave address must already be set to a valid address from 0-125 for a master to perform parameterization (Set_Prm). If the slave address is instead set to 126, then it will await the Set Slave Address command before proceeding. While awaiting the parameterization telegram, a slave will reject all other telegrams except Slave_Diag or Get_Cfg.

Set_Prm Telegram – SAP 61

This function is used to set the parameters of a slave at startup, after a restart or system reset, and at any time except within data exchange mode. At a minimum, the parameterization telegram contains 7 bytes of specific information required by the ProfiBus standard. This includes: Response Monitoring/Watchdog Time T_{WD}, T_{SDR} time for Master/Slave timing, Freeze/Sync Mode Enable/Disable, Lock/Unlock Slave for this Master, Group Assignment, Master Address, and Identification Number. Additionally, it may also contain other user-related parameters in bytes 8 to 32, or up to 244.

Set_Prm Parameterization Telegram With Header

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	X	X	X	8x	8x	X	3DH (61)	3EH (62)	X..	X	16H

Note that with SRD transmission, the DSAP specifies the Set_Prm function (61) request, while the SSAP requests that the Chk_Cfg function (62) follows (return response).

Set_Prm Parameterization Telegram DU Byte 1 – Station_status

7	6	5	4	3	2	1	0
Lock_Req	Unlock_Req	Sync_Req	Freeze_Req	WD_On	*Reserved –Set to 0*		
WD_Fact_1							

WD_On (Watchdog On): Set this bit to 1 to activate the watchdog control. Refer to the Watchdog section for more information.

Freeze_Req (Freeze Mode Request): If this bit is set, the slave will operate in the Freeze Mode as soon as the Global_Control function is received. If a slave does not support the Freeze control, it sets the Diag.Not_Supported bit within the diagnostic information.

Sync_Req (Sync Mode Request): If this bit is set, the slave will operate in Sync Mode as soon as the Global_Control function is received. If a slave does not support the Sync control, it sets the Diag.Not_Supported bit within the diagnostic information.

Unlock_Req (Unlock Request): See Table below.

Lock_Req (Lock Request): See Table below.

Lock Bit 7	Unlock Bit 6	Description
0	0	The Min T_{SDR} parameter may be changed. All other parameters remain unchanged.
0	1	DP Slave is unlocked/released for other masters.
1	0	DP Slave locked for other masters. All parameters are accepted and can be carried over (except min T_{SDR} = 0).
1	1	DP Slave is unlocked/released for other masters.

Set_Prm Telegram DU Byte 2 – WD_Fact_1, Range 1 to 255

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
WD_Fact_1							

Set_Prm Telegram DU Byte 3 – WD_Fact_2, Range 1 to 255

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
WD_Fact_2							

The watchdog is switched on or off via the WD_On bit of DU byte 1. Bytes 2 & 3 are factors used for setting the watchdog control (T_{WD}) time. The watchdog time is calculated between 10ms and 650 seconds as follows: T_{WD} =10ms * WD_Fact_1 * WD_Fact_2. The watchdog control causes the slave outputs to go to a failsafe state if the master fails to communicate with the slave before this time expires.

Set_Prm Telegram DU Byte 4 – Min T_{SDR} , Range 11 to 255 Tbit

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Min T_{SDR}: Minimum time in Tbit (bit time) after which a slave is allowed to answer. This value must be less than Max T_{SDR}. 11 Tbits are specified permanently in the standard.							

Byte 4 sets the minimum T_{SDR} time (in bit time, 11-255) a slave will wait before it is allowed to send a response to the master. If 00H is specified, the previous or default value is used.

Set_Prm Telegram DU Byte 5 (Ident_Number, High Byte)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

Set_Prm Telegram DU Byte 6 (Ident_Number, Low Byte)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

The Ident_Number of the slave is transmitted for security purposes. The Set_Prm parameters will not be accepted if this number does not match that of the slave. However, the min T_{SDR} can still be set if the Ident_number doesn't match and both the Lock_Req and Unlock_Req bits are set to 0.

Set_Prm Telegram DU Byte 7 (Group_Ident)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Group_Ident							

ProfiBus DP supports multi-cast messaging via a Global_Control telegram function directed to a specific group of connected slaves identified via this group number. Each bit represents a unique group. Note that Group_Ident is only accepted if the Lock_Req bit is also set.

Set_Prm Telegram DU Bytes 8 to 32 or 244 (Optional, User_Prm_Data)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Spec_User_Prm_Byte 8 (SPC3 ASIC related)							
0	0	0	0	0	WD_ Base	Dis_ Stopbit	Dis_ Startbit

In general, these bytes are user-defined for specific DP Slave device/module related parameters. They can be used to transmit startup information or for adjusting values or levels and generally take the place of DIP switches. Their meaning and range are application specific. Because the Acromag modules use the SPC3 ASIC, DU Byte 8 is defined as follows, and the remaining bytes are user-defined.

Set_Prm Telegram DU Byte 8: Spec_User_Prm_Byte (SPC3 ASIC)

Bit	Name	Description	Default Status
0	Dis_Startbit	Used to disable start bit monitoring in the receiver (1=disabled).	Dis_Startbit = 1 (Disabled)
1	Dis_Stopbit	Used to disable stop bit monitoring in the receiver (1=disabled).	Dis_Stopbit = 0 (Enabled)
2	WD_Base	Used to specify the time base with which the watchdog is pulsed (0=10ms, 1=1ms)	WD_Base = 0 (Time base is 10ms)
3-7	*Reserved*	Not Used – Set to 0.	0

Parameterization Telegram DU Bytes 9-244 (Optional User_Prm_Data)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
User_Prm_Data							

The first 7 bytes of Set_Prm are evaluated by the slave's ASIC (without user-prm_data) and in accordance with the standard, or the first 8 bytes (with Spec_user_prm_data). For Acromag slaves, the eighth byte is used for SPC3 related characteristics. The remaining bytes are available to the application.

The response of a slave to the parameterization telegram is "E5H" (a short acknowledge). After its parameters are set, the slave proceeds to the I/O Configuration State.

The parameter Status can be tested to indicate the success or failure of the parameterization telegram with possible values of: OK, DS, NA, RS, RR, UE, and RE.

I/O Configuration

After parameterization (Set_Prm), the slave awaits the configuration telegram (Chk_Cfg). This telegram specifies the number of input and output bytes that are to be exchanged in each telegram cycle with the slave. The configuration telegram also causes the slave to check the configuration which was sent against the stored configuration. A slave awaiting Chk_Cfg will only accept the Set_Prm, Slave_Diag, or Get_Cfg telegrams.

Chk_Cfg Telegram – SAP 62

The Chk_Cfg configuration telegram causes a slave to check the configuration which was sent, against the stored configuration. If the slave detects a conflict when it compares the sent information with the entries originating from the GSD file, it will report the incorrect configuration to the master when asked for diagnostics later and will not proceed to exchange data with the master. A slave will acknowledge a configuration telegram with "E5H" (short acknowledge).

A ProfiBus master can scan the configuration of the input/output data with the Read_Cfg telegram, and then configure the slave with Chk_Cfg. The slave response must contain a configuration with which the slave can actually boot.

Chk_Cfg Configuration Telegram With Header

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	X	X	X	8x	8x	X	62 (3EH)	62 (3E)	X..	X	16H

Configuration Telegram DU Byte 1

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

…

Configuration Telegram DU Byte x

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X

Format of DU Bytes 1-X

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
0=Consistency of Byte/Word 1=Consistency Entire Length	0=Byte 1=Word	Input/Output: 00=Special Format 01=Input 10=Output 11=Input and Output		Length of Data: 0000 = 1 Byte/Word 1111 = 16 Bytes/Word			

Get_Cfg Telegram – SAP 59

Note that the Read Configuration Data (Get_Cfg) telegram is accepted by a slave in any state and allows a master to scan the actual configuration of the slave (Real_Cfg_Data).

The parameter Real_Cfg_Data contains the configuration data as a string of 1 to 32 bytes (optionally up to 244 bytes) that have the same format as the identifiers of Chk_Cfg noted above.

The parameter Status can be tested to indicate the success or failure of this function with possible values of: OK, DS, NA, RS, UE, NR, and RE.

Diag_Data Telegram (Diagnostics Request) – SAP 60

If a diagnostic message becomes necessary during data exchange, the DP slave will signal this to the master by sending its response with high priority (see Function Code). Then in the next bus cycle, the master will send a diagnostic request telegram to the slave instead of the normal data exchange telegram. Further, any master (not just the assigned master) can request diagnostic data from any slave at any time.

The Diag_Data telegram is used by the master to request diagnostic information from the slave. During startup, a master typically requests diagnostic data before sending the parameterization telegram, and then again after configuration, before it assumes the data exchange mode with the slave.

The master evaluates the diagnostic information to determine if the parameterization and configuration info is correct. If no further diagnostic service is required, the master proceeds to exchange data with the slave.

Diag_Data Request Diagnostics Telegram With Header

SD2	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	FCS	ED
68H	X	X	68H	8x	8x	X	3CH (60)	3EH (62)	X	16H

Request Diagnostics Response Telegram With Header

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	X	X	68H	8x	8x	X	3EH (62)	3CH (60)	X..	X	16H

The diagnostic information of a DP slave consists of 6 bytes of standard diagnostic information, plus any user-diagnostic information (slave specific). A set bit (1) in a position means the linked definition has occurred. The parameter Status can be tested to indicate the success or failure of the Diag_Data function with possible values of: OK, DS, NA, RS, UE, NR, & RE.

Diag_Data Response Telegram DU Byte 1 – Station_status_1

Bit	DIAGNOSTIC
0	**Diag.Station_Non_Existent:** Set to 1 by the master if slave cannot be reached over the line. Slave sets this bit to 0.
1	**Diag.Station_Not_Ready:** Set by slave if slave is not ready for data transfer.
2	**Diag.Cfg_Fault:** Set by slave if it detects a mismatch in config data.
3	**Diag.Ext_Diag:** Set by slave to indicate a diagnostic entry is in the slave-specific diagnostic area (see below).
4	**Diag.Not_Supported**: Set by slave if requested function/service is not supported.
5	**Diag.Invalid_Slave_Response:** Slave sets this bit to 0. Set to 1 by the master if it receives an implausible response from the slave.
6	**Diag.Prm_Fault:** Set by slave if last parameter frame was faulty (wrong parameterization, bad length, bad ident_number, etc.).
7	**Diag.Master_Lock:** Set by a class 1 master to indicate slave has been parameterized by another master (if address in DU byte 4 is not 255 and differs from its own address). Set to 0 by slave.

Diag_Data Response Telegram DU Byte 2 – Station_status_2

Bit	DIAGNOSTIC
0	**Diag.Prm_Req**: Set by a slave if it needs to be parameterized and cleared once parameterization is complete.
1	**Diag.Stat_Diag:** Static diagnostics. Slave sets this bit to cause the master to retrieve diagnostic information until this bit is cleared (the slave sets it if it's not able to provide user data).
2	*Slave sets this bit to 1.*
3	**Diag.WD_ON:** Set by slave to indicate Watchdog is active.
4	**Diag.Freeze_Mode:** Set by slave after it has received the Freeze control command.
5	**Diag.Sync_Mode:** Set by slave after it has received a Sync command.
6	*Reserved.*
7	**Diag.Deactivated:** Set by the master if slave has been marked inactive within the slave parameter set and is removed from cyclic processing. Slave sets this bit to 0.

Diag_Data Response Telegram DU Byte 3 – Station_status_3

Bit	DIAGNOSTIC
0-6	*Reserved.*
7	**Diag.Ext_Diag_Overflow:** Set if there is more diagnostic information than specified in Ext_Diag_Data. For example, slave sets if slave has more diagnostics than it can enter into its send buffer. Set by master if slave sends more diagnostic information than it can enter into its diagnostic buffer.

Diag_Data Response Telegram DU Byte 4 (Para Master Address)

Bit	DIAGNOSTIC
0-7	**Diag.Master_Add:** The master's address that parameterized this slave is entered here. If no master has parameterized this slave, then the DP Slave inserts 255 here (FF without parameterization).

Diag_Data Response Telegram DU Byte 5 - Ident_Number High Byte

Bit	DIAGNOSTIC
0-7	Manufacturer Identification Number High byte for ID & verification

Diag_Data Response Telegram DU Byte 6 - Ident_Number Low Byte

Bit	DIAGNOSTIC
0-7	Manufacturer Identification Number Low byte for ID & verification.

DU Bytes 7-32 (or optionally up to 244 bytes) contain DP-Slave specific diagnostic information structured in blocks according to format type: device-related, identifier-related, and channel-related.

Diag_Data Response Telegram DU Byte 7-X – Ext_Diag_Data

Bit	DIAGNOSTIC
0-7	Ext_Diag_Data (see formats below, refer to your model specifications for extended diagnostic data).

Ext_Diag_Data - Application Specific Diagnostics

The master can store user-related diagnostics in the following three different formats: *device related*, *identifier related*, and *channel related*.

Device-Related Diagnostic

This information is device-related and can be coded in any form. It can be used to indicate general diagnostic information such as: over-temperature, under-voltage, over-voltage, etc.

Device Related Diagnostic Byte 1 of X – Header Byte

7	6	5	4	3	2	1	0
0	0	X	X	X	X	X	X
Bits 7,6 = 00		Bits 5-0 = Block Length in bytes including header byte 2 to header byte 63.					

Device Related Diagnostic Byte 2 of X – Diagnostic Field Byte(s)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Diagnostic coding is device-specific and can be coded as desired.							

Device-Related Diagnostic Example:

Byte	7	6	5	4	3	2	1	0	
1	**0**	**0**	0	0	0	1	0	1	Device Related Diagnostic, 5 bytes (Header + 4bytes)
2	X	X	X	X	X	X	X	X	*Code As Desired*
3	X	X	X	X	X	X	X	X	*Code As Desired*
4	X	X	X	X	X	X	X	X	*Code As Desired*
5	X	X	X	X	X	X	X	X	*Code As Desired*

Identifier-Related Diagnostic (For Modular Systems)

This diagnostic structure is based on a modular system, where each module has one identifier (configuration byte). Thus, a defective or faulty module can be easily detected and no additional description necessary. One bit is reserved for every identifier byte specified in the configuration (e.g. 0 x 10 for 1-byte input). A set bit in a bit position means that this I/O diagnostic is pending. Bits not configured are set to 0 and bytes are padded to the byte limits. When modular systems with one identifier byte per module are involved, you can indicate diagnostics by specific module. One bit per module indicates the diagnosis.

Identifier-Related Diagnostic Byte 1 of X – Header Byte

7	6	5	4	3	2	1	0
0	**1**	*X*	*X*	*X*	*X*	*X*	*X*
Bits 7,6 = 01		Bits 5-0 = Block Length in bytes including header byte 2 to header byte 63.					

Identifier-Related Diagnostic Byte 2 of X – Bit Structure

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
A set bit in a bit position means that the corresponding identifier byte has the diagnostic..							

Identifier-Related Diagnostic Example:

Byte	7	6	5	4	3	2	1	0	
1	**0**	**1**	0	0	0	1	0	1	Identifier Related Diagnostic, 5 bytes (Header + 4bytes)
2	0	0	0	0	0	0	0	1	*Identifier Number 0 (Module 1) Has Diagnostic*
3	0	0	1	0	0	0	0	0	*Identifier Number 13 (Module 14) Has Diagnostic*
4	0	0	0	0	0	0	1	0	*Identifier Number 18 (Module 19) Has Diagnostic.*
5	0	0	0	0	1	0	0	0	*Identifier Number 28 (Module 29) Has Diagnostic.*

Channel-Related Diagnostic

This diagnostic structure is used for pre-defined failure types with specific identifiers used to identify the specific faults. Fault definitions can be defined per module and per channel. Additional device-specific definitions are also possible. In this block beginning at DU byte 7, the diagnosed channels and diagnostic reasons are entered in sequence, one at a time, with 3 bytes per diagnostic entry according to the following format:

Channel Related Diagnostic Byte 1 of 3 – Identifier Number

7	6	5	4	3	2	1	0
1	**0**	X	X	X	X	X	X
Bits 7,6 = 10		Bits 5-0 = Identifier Number, 0 to 63					

Channel Related Diagnostic Byte 2 of 3 – Channel Number

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Bits 7,6 = Input/ Output Coding: 00=Reserved 01=Input 10=Output 11=Input/Output		Bits 5-0 = Channel Number, 0 to 63					

Channel Related Diagnostic Byte 3 of 3 – Error Type

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Channel Type: 000 = Reserved 001 = Bit 010 = 2 Bit 011 = 4 Bit 100 = Byte 101 = Word 110 = 2 Words 111 = Reserved			Error Type: 0 = Reserved 1 = Short Circuit 2 = Under-voltage 3 = Over-voltage 4 = Overload 5 = Over-temperature 6 = Line/wire Break 7 = Upper Limit Value Exceeded 8 = Lower Limit Value Exceeded 9 = Error 10..15 = Reserved 16..31 = Manufacturer Specific/Device Related				

If a slave transmits more diagnostic data than the master is able to process in its diagnostic buffer, the master sets the Diag.Ext_Diag_Overflow bit. If there is more diagnostic information pending at the slave than can be transmitted, the slave is only allowed to truncate at the block limits of the device, identifier, or channel related diagnostic. Further, if the block length field of the device or identifier related diagnostic contains a non-zero length, this marks a complete (not truncated) diagnostic block. For the sake of efficiency, the slave is allowed to transmit a Diag_Data field of fixed length (the unused bytes following Ext_Diag_Data are filled with zero at the slave and/or class 1 master).

The parameter Status can be tested to indicate the success or failure of the Diag_Data function with possible values of: OK, DS, NA, RS, UE, NR, and RE.

The following bits are taken from the diagnostic information bytes above and treated as follows:

PRM_REQ (From DU Byte 2)

If this flag is set, the slave has to be parameterized. The application has detected a state requiring a new startup with appropriate reparameterization and reconfiguration. Following this diagnostic, the master performs a startup with specified reparameterization and configuration.

An example where this type of diagnostic might occur is if a modular ProfiBus system has been expanded (i.e. a module was added).

There are three diagnostic bits that can be driven by the application: STAT_DIAG, EXT_DIAG, and EXT_DIAG_OVERFLOW.

STAT_DIAG (From DU Byte 2)

Because of a state in the application, the slave cannot provide valid data. After receiving this diagnostic, the master continues to request only diagnostic information from this slave until the slave resets this bit. If the normal ProfiBus DP state is data exchange, data communication can be resumed immediately after the static diagnosis bit is cleared. An example of where this diagnostic may be encountered is for a slave whose output driver voltage supply has failed.

EXT_DIAG (From DU Byte 1)

The slave uses the EXT_DIAG=1 bit to signal that user-specific diagnostic data is present in the user diagnostics area. EXT_DIAG=1 causes a diagnostic telegram to be sent to the ProfiBus master. After the cause of the diagnostic message has been corrected (the applicable bit combination in the user-related diagnostic data is 0), the EXT_DIAG bit must also be reset (set to 0) and this is necessary for certification.

If this bit is set, a diagnostic entry must be present in the user-related diagnostic area. If this bit is clear, then the standard status information of the diagnostic area is involved and this is handled with a lower priority. When the EXT_DIAG bit is cleared (0), the data must be considered as status information from the viewpoint of the system and this data is not treated as diagnostic data by the master.

EXT_DIAG_OVERFLOW (From DU Byte 3)

The slave sets this bit when more diagnostic data is available than will fit into the area provided for diagnostic data. For example, more channel diagnostic data may be present than the sending buffer or the receiving buffer of the master can hold.

Data Exchange State

After parameterization and configuration have been accomplished, the master can start exchanging cyclical I/O data with the slaves. The following services are available in data exchange mode: Read_Inp (read the inputs of any slave), Read_Outp (read the outputs of any slave), and Data_Exchange (send and receive data to the slave parameterized and configured by the master). A slave will automatically check the transferred output data, respond with the input data, and generate a message if it detects a discrepancy.

The Data_Exchange function refers to the cyclic transfer of I/O data and possible diagnostic information between slaves and their class 1 masters. Recall that ProfiBus may use SRD transmission (Send and Request Data with Acknowledge) that allows it to send output data and receive input data in a single message/response cycle. With Data_Exchange, the number of inputs and outputs has already been defined in the configuration data at the system startup. In Data_Exchange mode, the master cyclically sends the output data to the slave and receives input data (if present) in return. If the slave is purely an output device (no input data to return), it responds with "E5H" in its response data field (a short acknowledge). Unlike every other telegram which has 11 bytes of header information, the Data_Exchange telegram has only 9 bytes of header information as it uses the default SAP (implied), with the DSAP & SSAP bytes are dropped from the telegram header. In Data_Exchange mode, the slave will allow the reconfiguration of I/O data to occur (Chk_Cfg), but will not permit reparameterization.

Data_Exchange Send Output or Receive Input Data Telegram

SD	LE	LEr	SD	DA	SA	FC	DU	FCS	ED
68H	X	X	68H	8x	8x	X	X..	X	16H

Recall that with data exchange, the telegram header has only nine bytes since the default SAP is used and the DSAP & SSAP bytes are dropped from the telegram header. This is indicated by the most significant bit of the DA & SA bytes which are clear (0). If the MSB is set to 1 in the DA & SA bytes, this indicates a DSAP & SSAP follows in the telegram header. The DU contains from 1 to 244 bytes of user data to be transferred (output data and/or input data).

In Data_Exchange mode, any master can read the I/O data of any slave at any time using the "Read_Inputs" and "Read_Outputs" telegrams. These telegrams have the same structure as the cyclic Data_Exchange telegram, but include the DSAP and SSAP bytes. For these telegrams, the MSB of the DA & SA bytes will be set to "1" to indicate that a DSAP & SSAP byte follows in the telegram header.

Read_Inp Telegram – SAP 56

The master can use this telegram to *asynchronously* read the input data (Inp_Data) of any slave in Data_Exchange mode.

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	FCS	ED
68H	05H	05H	68H	8x	8x	X	38H (56)	3EH (62)	X	16H

The response telegram format is the same as noted above, but with the DSAP/SSAP mirrored (swapped) and the DU bytes embedded.

Read_Outp Telegram – SAP 57

The master can use the Read Outputs telegram to *asynchronously* read the output data (Outp_Data) of any slave in Data_Exchange mode.

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	FCS	ED
68H	05H	05H	68H	8x	8x	X	39H (57)	3EH (62)	X	16H

The response telegram format is the same as noted above, but with the DSAP/SSAP mirrored (swapped) and the DU bytes embedded.

Global_Control Service – SAP 58

ProfiBus DP uses the Global Control function to send special commands addressed to a single slave, a specific group of slaves (multi-cast), or to all slaves at once (broadcast). ProfiBus sends broadcast and multi-cast messages as global control telegrams using address 127 (an optional group number is included for a select group with multi-cast).

Using SDN transmission (Send Data with No acknowledge), a class 1 master will use the Global_Control service to inform the slaves of his mode (Operating or Clear Mode), or to send commands such as sync, unsync, freeze, unfreeze, and clear data to a group of slaves, typically for synchronization purposes. Note that a slave will only accept this command from the same master that parameterized and configured it. There is no response returned to an SDN telegram.

Global_Control Telegram

SD	LE	LEr	SD	DA	SA	FC	DSAP	SSAP	DU	FCS	ED
68H	X	X	68H	8x	8x	X	3EH (62)	3AH (58)	X..	X	16H

DU Byte 1 (Control_Command To Be Executed)

7	6	5	4	3	2	1	0
Res.	*Res.*	Sync	Unsync	Freeze	Unfreeze	Clear_Data All outputs cleared.	*Res.*
0	0	00=No Function 01=Deactivated 10=Activated 11=Deactivated		00=No Function 01=Deactivated 10=Activated 11=Deactivated		0=Do Not Clear Outp 1=Clear Outputs	0

Sync: The output states transferred in Data_Exchange are delivered and frozen. The output data which follows is held until the next Sync command or Unsync command.

Unsync: This control cancels the sync command.

Freeze: This causes the states of the inputs to be read and frozen until the next Freeze command or Unfreeze. Slaves must ensure that following a freeze command, the last frozen values of the inputs must be transferred in the next data exchange cycle.

Unfreeze: This control cancels the freeze command.

Clear_Data: A class 1 master may use a global control telegram to inform the slaves that it is switching from Operate Mode to Clear Mode. This bit is set in Clear Mode (02 00H), and cleared in Operate Mode (00 00H). A slave will respond by either clearing its outputs, or it may optionally assume a user-defined state with its master in Clear Mode. Please refer to Failsafe Operation for more information on the use of this bit.

DU Byte 2 (Group_Select or Group_Ident Number)

7	6	5	4	3	2	1	0
X	X	X	X	X	X	X	X
Group_Select: 0-255, Number must match the Group_Ident number of the parameterization data (multi-cast). If set to 0, all slaves are addressed (broadcast).							

Sync & Freeze are optional slave services and may not be supported by some slaves. A master uses a Freeze telegram to make a slave or group of slaves freeze their inputs in the current state. A Sync telegram causes the output data currently available to be transferred to the outputs, to be frozen. An Unsync & Unfreeze command will cancel this state. The Clear Data service allows the outputs to be switched to a defined state if an error occurs. The response parameter Status can be tested to indicate whether transmission of the request frame was successful or not, with possible values of: OK, DS, NO, and IV.

Use of Freeze

In closed-loop control systems and for the synchronization of drives, etc., it is sometimes necessary to have a precise time image of the process inputs of a group. Freeze is used to accomplish this as follows:

The master sends a Freeze command to the selected group *(A global control telegram goes to the selected group at a precise time).* This causes all addressed slaves to freeze their inputs. During the next Data Exchange cycle, the slaves transfer the frozen inputs of the group to the master.

After the master receives this data, the master sends an Unfreeze command to the group and the bus system returns to the normal data exchange mode again, and all input changes are transferred during each data cycle.

Use of Sync/Unsync

For the time-controlled operation of output devices which belong to a group, sync and unsync are used as follows:

After the data is frozen with a Freeze command and has been processed by the master, the master reacts by sending a sync command to the slave group to obtain the outputs. During the next data cycle, the master supplies the slave group with the data to be output, then concludes this cycle with an unsync command in the following data exchange cycle. Unsync causes the slaves to transfer their outputs at a precise time.

BUS TIMING

Unlike CAN and Ethernet which are event-driven busses, ProfiBus was designed to guarantee a deterministic response.. The *determinism* of a system refers to the ability to precisely predict the behavior of the system over time.

ProfiBus uses a polling mechanism between master and slave. The time it takes a slave to respond to a message from the master is the *reaction time*. Even if a ProfiBus system receives many I/O signal changes at some point in time, there is no change in reaction time. Further, even if another master (class 2) is used to perform diagnostics on a slave device while it is communicating with its class 1 master, the reaction time for the system will remain the same. This is because the class 2 master used to perform diagnostics will not be allowed to use more time than the configured gap time within the bus cycle.

Because ProfiBus is deterministic, we can calculate a reliable system reaction time. But before we get into the details of calculating bus cycle times, we must define a few terms as follows:

Bit-Time: To help simplify timing calculations, it is convenient to normalize the time units with respect to the baud rate by using units of Bit-Time (Tbit). One bit-time is the time it takes to transmit one bit and is the reciprocal of the transmission rate (baud rate). For example:

1 Tbit (Bit Time) at 12MB = 1/12000000bps = 83ns/bit

Sync-Time (T_{SYN}): The synchronization time is the minimum time a station must remain in the idle state before it can accept another request. For ProfiBus DP, an idle state of 33Tbits (bit-time) must be present before every request telegram and this is called the sync-time.

Slave Reaction Time (T_{SDR}): The reaction time is the time it takes a slave to respond to a message. This time is often expressed as a minimum value (min T_{SDR}), or maximum value (max T_{SDR}). Min T_{SDR} is set within the parameterization telegram during startup. Max T_{SDR} varies with the transmission rate and is specified at the supported baud rates within the device GSD file. For ProfiBus DP, this value may range from a minimum of 11Tbits (min T_{SDR} default) to a maximum of 255Tbits.

Initiator Delay Time (T_{SDI}): T_{SDI} refers to the station delay of the *initiator* of a request or token frame (the master).

Initiator Idle Time (T_{ID1}): After receiving the last character of a telegram, the initiator must wait this amount of time until it sends the next telegram. The idle time (Tid1) is the time between transmission of the last bit of a frame (no acknowledge) and the transmission of the first bit of the next frame. It is at least the sync time (T_{SYN}), plus some safety margin (Tsm), but is also calculated as the maximum of these three values: T_{SYN} + Tsm, min T_{SDR}, or T_{SDI} (station delay of telegram initiator). The addition of safety margin (Tsm) is very important at high baud rates.

Minimum Slave Interval: The minimum slave interval is the minimum time that must expire between two slave polling cycles in which a slave can exchange data with the master. To permit the slave station to be able to respond during every data cycle, it controls the bus cycle with this parameter. It is defined in the slave's GSD file via the parameter Min_Slave_Interval, which is specified as a 16 bit factor of 100us (Min_Slave_Interval = 1 is 100us). On some older equipment, the ProfiBus link was implemented in software (as opposed to within the slave ASIC) and a typical value was about 2ms. On newer equipment with modern ASIC's, values down to 100us can be achieved.

Calculating System Reaction Time

A simplified calculation of system reaction time for a ProfiBus DP system is derived from the following parameters:

- T_{SDR} (Station Reaction Time).
- The Transmission (Baud) Rate.
- The Net Data Length specified.
- Min_Slave_Interval (min time between two slave polling cycles).

Example: One master and 5 slaves are connected via ProfiBus DP. Ten bytes of output data and 20 bytes of input data are to be transferred per slave at 12Mbps. Each slave utilizes an SPC3 ASIC. Calculate the relative bus cycle time for this network.

Let T_{MC} = Time of 1 telegram cycle
(request telegram + T_{SDR} + slave response).
Let T_{BC} = Time of 1 bus cycle (the sum of all telegram cycles).

Given:
T_{SYN} = 33 TBits (Bus idle time or ProfiBus Sync-Time)
T_{ID1} = 75 TBits (SPC3 bus idle time, at 1.5MB T_{id1} = 36 TBit).
T_{SDR} = 30 TBits typical for baud rates ≥ 1.5MB (SPC3 ASIC).
Min_Slave_Interval = 1 (100us, from slave GSD file).

Calculate:
1 Tbit (Bit Time) at 12MB = 1/12000000bps = 83ns/bit

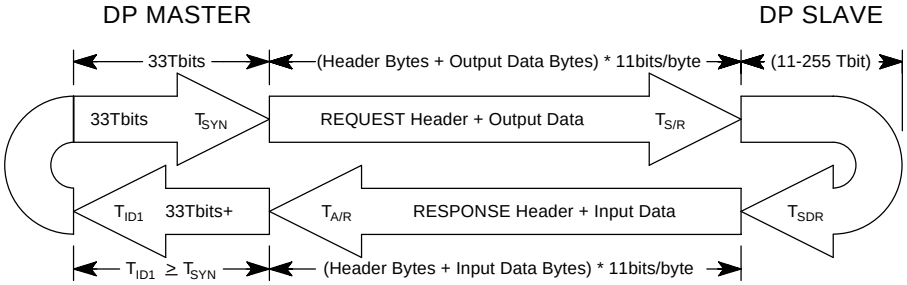
In data exchange mode, a telegram header consists of only 9 character bytes. If we include the bits of the character frame, there are 11 bits for every character byte (Start Bit + 8bits/char + Stop Bit + Parity). Since only 1 master is present, we can ignore the token hold time of token telegrams. Thus, the basic time required by one telegram cycle (not including data) is obtained by adding the relevant bus times and the time to transmit the telegram header as follows:

T_{MC} (in TBits) = 2 * 9(header byte length) * 11bits/byte + T_{SDR} + T_{SYN} + T_{id1}
T_{MC} = 198 bits + 30 bits + 33 bits + 75 bits = 336 Tbits
T_{MC} (us) = 336Tbits * 83ns/Tbit = 28us

Thus, 28us is the basic time required by the telegram header including the bus times without accounting for the data. For our example, we must include the data (10 bytes Output + 20 bytes Input). Recall that the SRD service (Send and Request Data with acknowledge) will send data to the outputs and receive data from the inputs in one telegram cycle. The time for a single telegram cycle with this data included is:

T_{MC} = [336Tbits] + amount of net data
= 336 + [10bytes Output + 20bytes Input]*(11bits/byte) = 666Tbits
T_{MC} = [28us] + 330Tbits*83ns/bit
T_{MC} = [28us] + 27.39us = 55.39us/slave.

To simplify this calculation, you can assume that a basic transfer time of 28us plus approximately 1us per DU data byte (actually 0.83us/byte) is required to complete a telegram cycle. The following figure gives an overview of the dominant bus times in a telegram cycle (assuming no interference or repetitions).



TELEGRAM CYCLE WITH RELEVANT BUS TIMES

Timing of 1 Message Cycle = $T_{MC} = ((T_{S/R} + T_{SDR} + T_{A/B})*T_{TD}) + T_{ID}$

Note that the slave has a Min_Slave_Interval of 100us and this dominates the bus timing for one telegram cycle. However, the Min_Slave_Interval is 100us between two polling cycles at the same station. If you have at least 3 stations present, then the actual transmission time at 12MB will become the determining time factor for the bus cycle rather than the Min_Slave_Interval. Refer to the EN50170 standard for a more detailed calculation of transmission time.

HIGH SPEED REQUIREMENTS

Please note the following for communication at baud rates greater than 1.5Mbps:

- The ProfiBus connector has built in series inductors on the data lines for operation at the higher baud rates. This is one more reason that you should only use approved ProfiBus connectors.
- For operation at 12Mbps, a minimum cable length of 1M is required between stations.