Modbus RTU
REF 542plus

Technical Reference

Contents

Copyrights

Trademarks

Guarantee

Please inquire about the terms of guarantee from your nearest ABB representative.

1. Introduction

1.1. This manual

The Multifunction Protection and Switchgear Control Unit REF 542plus handle several data for protection, control and monitoring of the switch panel, as well as for the GIS panels type ZX as also for the AIS panels ZS1. All of the data are available on the internal main controller serial port. The application of Modbus RTU board within the Multifunction Protection and Switchgear Control Unit REF 542plus allows one or two control systems to be connected. In case of two control systems it is possible to send and to receive the same command and monitoring data using Modbus RTU communication protocol. The ASCII mode is not available.

1.2. Use of symbols

This publication includes the following icons that point out safety-related conditions or other important information:

The electrical warning icon indicates the presence of a hazard which could result in electrical shock.

The warning icon indicates the presence of a hazard which could result in personal injury.

The caution icon indicates important information or warning related to the concept discussed in the text. It might indicate the presence of a hazard which could result in corruption of software or damage to equipment or property.

The information icon alerts the reader to relevant facts and conditions.

Although warning hazards are related to personal injury, it should be understood that operation of damaged equipment could, under certain operational conditions, result in degraded process performance leading to personal injury or death. Therefore, comply fully with all warning and caution notices.

1.3. Intended audience

This manual is intended for operators and engineers to support normal use of as well as configuration of the product.

1.4. Product documentation

Name of the Manual	Document ID
Application Notes	1MRS755870
CAN Manual	1VTA100189-Rev 1, en
Configuration Manual	1MRS755871
iButton Programmer User Manual	1MRS755863
Manual Part 3, Installation and Commission	1 VTA100004
Manual Part 4, Communication	1VTA100005
Motor Protection with ATEX Certification, Manual	1MRS755862
Operator's Manual	1MRS755869
Protection Manual	1MRS755860
Technical Catalogue	1MRS755859
Technical Reference Modbus RTU	1MRS755868
Web Manual, Installation	1MRS755865
Web Manual, Operation	1MRS755864

Other related documents

[1] Gould, "Modbus protocol reference guide", January 1985

[2] ABB Network Partner "SPA bus Communication Protocol V 2.5 − Technical Description", 1996

[3] "Reading of fault recorder data from RE.316 and RE.216 via SPA bus V1.4", Draft ABB

1.5. Revision history

Version	Date	History
1VTA100065	01.06.2004	First release
A	28.02.2006	Document updated • language • layout
B	30.09.2006	Document updated

2. Safety information

Dangerous voltages can occur on the connectors, even though the auxiliary voltage has been disconnected.

Non-observance can result in death, personal injury or substantial property damage.

Only a competent electrician is allowed to carry out the electrical installation.

National and local electrical safety regulations must always be followed.

The frame of the device has to be carefully earthed.

The device contains components which are sensitive to electrostatic discharge. Unnecessary touching of electronic components must therefore be avoided.

3. Main features

Main features of the Multifunction Protection and Switchgear Control Unit REF 542plus communication by using Modbus RTU:

Description	**Features**
Type of transmission	Serial asynchronous
Protocol	Modbus RTU slave
Baud rate	300, 600, 1.200, 2.400, 4.800, 9.600, 19.200, 38.400, 76.800, 153.600 bauds
Data bits	11 (including start, stop, parity) start bit (1) data bits (7,8) stop bits (0,1,2)
Parity check	None, even, odd
First communication mode	
Data flow	Half duplex
Connections	RS485
Connectors	2 wires terminal block
Maximum distance	130 m
Second communication mode	
Data flow	Full duplex
Connections	Glass fiber optic
Connector	Standard ST
Maximum distance	2000 m
Number of channels	Up to two for each communication mode
Number of units to be connected on the same bus	Up to 32

4. Processed data

All the data listed in the SPA bus table in REF 542plus User Manuals can be processed by the Modbus RTU board.

- Measurement values
- Status of binary input and output
- Control command
- Reading and setting parameters values
- Reset Alarm
- Reading System parameters
- Reading of protection events from the Start / Trip page
- Reading fault recorder data
- Time synchronisation
- Time setting
- Re-programming the application software
- Configuration and program description

The communication between the Multifunction Protection and Switchgear Control Unit REF 542plus and the upper system level is based on master-slave procedures.

4.1. Event chronology

The event chronology is codified in the SPA bus table. The buffer is in position to record the last 100 events.

As the master unit sends out a request, the Multifunction Protection and Switchgear Control Unit REF 542plus transmits the stored events, marked by the absolute time (year-month-day-hour-second-millisecond). REF 542plus shows the number of stored events in a dedicated location, so that the master unit can read the event table (polling).

4.2. Fault recording

The 7 analog signals and the 32 binary signals can be recorded by the Multifunction Protection and Switchgear Control Unit REF 542plus. On request, the transmission to the master unit is possible as well. In the master unit, the file must be converted in COMTRADE format. Therefore, the corresponding conversion software, SMSCONVERT, will be provided.

The records of the system faults are stored in a buffer of maximal recording time of 5 s and the buffer can contain a maximum of 5 records, which means that the minimum recording time is 1 s. If required, the maximal recording time can also be set for an arbitrary number of records up to 5. The Multifunction Protection and Switchgear Control Unit REF 542plus is designed with a dedicated memory for storing recorded events, so that the master unit can read the records by polling.

5. Communication channels

The communication board is available in two hardware versions. The first is a version with a serial communication channel according to the RS485 Standard on twisted shielded pair. The second is a version with glass fiber optic connections type ST.

The two-channel version can be used for the connection between Switchgear Protection and Control Unit REF 542plus and two separates systems working independently (such as DCS and SCADA).

The two channels can work simultaneously and operate separately. In order to guarantee a high level of safety, the data is processed as described in the following table:

R/W	Access	Mode
Data readout	Always active on both channels.	Simultaneous readout
Events readout	Always active on both channels.	Simultaneous readout
Fault recording readout	Always active on both channels.	Simultaneous readout
Control writing	Active on a single channel after the opening of a control session.	After one of the two channels get the input to open a control session the same operation cannot be performed on the other channel. A further control session is made possible only after the conclusion of the previous session.
Parameter writing	Active on a single channel after the opening of a parameter session.	After one of the two channels get the input to open a parameter session the same operation cannot be performed on the other channel. A further parameter session is made possible only after the conclusion of the previous session.

Both the control and the parameter session can work independently; they can be activated simultaneously either on the same channel or separately on the two channels. The controls and the parameters can be accessed only after the opening of the relevant sessions. The sessions can be closed in two different ways: on request or automatically, should the opening time exceed the maximum preset time of 10 minutes. The session opening does not affect the local or remote functionality of REF 542plus.

6. Configuration mode

All the reading and writing activities performed by the Multifunction Protection and Switchgear Control Unit REF 542plus and the Modbus RTU communication systems are based on a memory map located in the communication board. A dedicated PC for the configuration contains the configuration tool to define this map. Connecting RS485 or the optical fibres communication channel with the serial channel of the PC configures the communication board. The ABB Modbus Configuration Tool is able to program all the units connected to the same communication bus as well as to work on one single unit.

While the memory map is unique for both channels the two channels on the communication boards can be configured independently on both sides:

R/W	Access	Mode
Configuration writing	Active on a single channel after the opening of a configuration session.	After one of the two channels get the input to open a configuration session, the same operation cannot be performed on the other channel. A further configuration session is made possible only after the conclusion of the previous session.

The communication board can be configured after opening of the relevant session. The opening of a session does not affect the local functionality of REF 542plus. Only the communication to the upper level control system is stopped.

The only limitation of the information flow is to change the status of the switching devices. In case of a direct connection between the PC and the communication bus, all of the communication activities of REF 542plus, which are not involved in the session, remain active.

The sessions can be closed in two different ways: on request or automatically, if the opening time exceeds the maximum preset time of 5 minutes.

7. System architecture

The different possible architectures can be seen in Fig. 7.-1 and Fig. 7.-2.

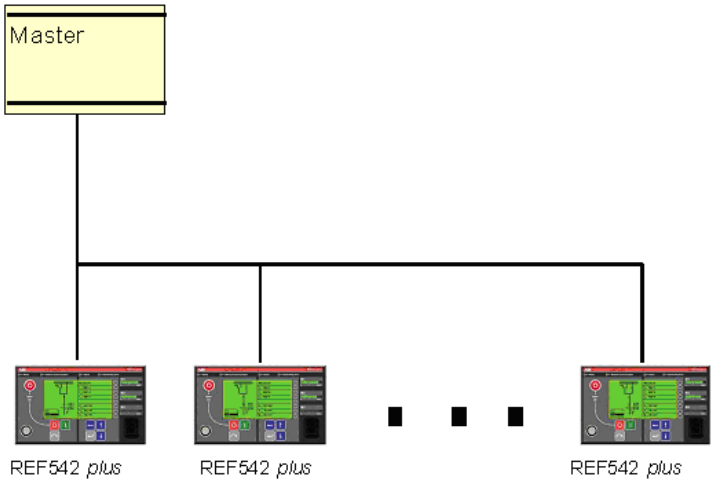


Fig. 7.-1 Single communication channel

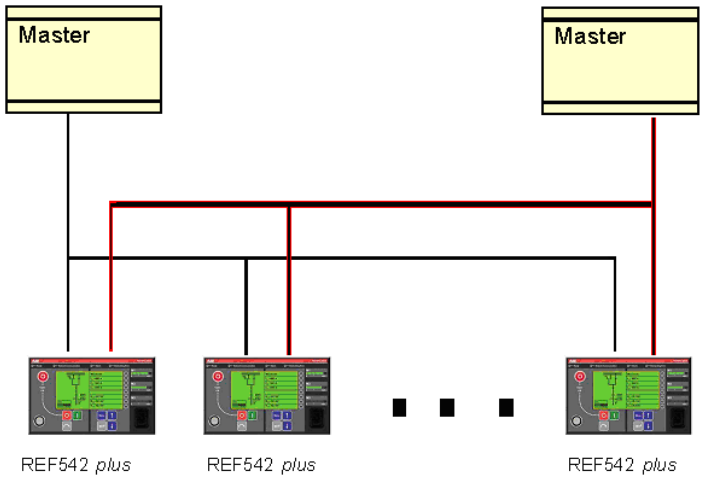


Fig. 7.-2 Double communication channel with two distinct and independent channels

8. Technical data of the communication ports

The communication board has two possible types of ports:

- Isolated RS-485 with two independent channels
- Fiber optic with two independent channels

The two different communication channels are selectable.

In the RS-485 version, the communication flows as half duplex channel for each port.

In the fibre optic version, the communication flows as full duplex channel for each port.

8.1. Optoisolated RS485

Technical data of the optoisolated RS485 interface is the following:

- Electrical isolation. 2000 V 50 Hz, 1 minute
- Interface signals: Data-A, Data-B (no ground reference)
- Connector: 2-pin, 5.08 mm spacing
- Cable: twisted pair (if shielded, connect the shield, one side only, on the earth screw of the case)
- Maximum baud rate: 153600 bit/s
- Maximum cable length: 130 m

8.2. Fibre Optic interface

The fibre optic communication interface can be used alternatively to the opto-isolated communication board with RS-485 interface. The Modbus RTU communication board is designed in such a way, that both plastic and a glass fibre Transmitter/Receiver can be mounted. The plastic and glass fibre interfaces have different performances as specified in the following example:

Glass Fiber interface	
Transmitter	HFBR 1412 (HP)
Receiver	HFBR 2412 (HP)
Connector	ST
Cable	Glass Fibre, 62.5/125 mm diameter
Maximum cable length	2000 m (it depends on the fibre attenuation per meter)
Wavelength	820 nm

Table 7.-2 (Continued)

Glass Fiber interface	
Transmitted power	-12 dBm (typical)
Receiver sensitivity	-24 dBm
Maximum baud rate	115000 bit/s

9. Modbus RTU protocol

On the communication board only the Modbus RTU mode is implemented [1]. The Modbus RTU protocol is implemented according to the following restrictions described in the following technical reference.

The Modbus functions are handled according to the following table:

Function	Meaning	Implementation	Notes
1	Read coil status	Full compliant	Based on dynamic DDEFs
2	Read input status	Full compliant	Based on dynamic DDEFs
3	Read holding registers	Full compliant	Based on dynamic DDEFs
4	Read input registers	Full compliant	Based on dynamic DDEFs
5	Force single coil	Full compliant	Based on dynamic DDEFs
6	Preset single register	Full compliant	Based on dynamic DDEFs
7	Read exception status	Full compliant	On static custom data
8	Loopback diagnostic test	Compliant	On static custom data
9	Program − 1	Not implemented	
10	Poll program complete - 1	Not implemented	
11	Fetch event counter communications	Not implemented	
12	Fetch communications event log	Not implemented	
13	Program − 2	Not implemented	
14	Poll program complete − 2	Not implemented	
15	Force multiple coils	Full compliant	Based on dynamic DDEFs
16	Preset multiple registers	Full compliant	Based on dynamic DDEFs

Table 7.-2 (Continued)

Function	Meaning	Implementation	Notes
17	Report slave Id	Compliant	On static custom data
18	Program − 3	Not implemented	
19	Reset communications link	Not implemented	
20	Read general reference	Compliant on custom data	On 12 custom files
21	Write general reference	Compliant on custom data	On 12 custom files
22 ... 64	Reserved for expanded functions	Not implemented	

Each Communication board serial channel may have a different Modbus Slave Address. These addresses are supplied to the communication board following the steps described in the following sections.

9.1. Modbus RTU addressing

The Modbus RTU protocol permits to read or write data by means of their addresses in the Modbus virtual address space.

Such an address space is partitioned according to the following data types:

Range	Length	Type	Access functions
1 ... 10000	1 bit data	COILS	1 (read) , 5 (write), 15 (multiple write)
10001 ... 20000	1 bit data	INPUT STATES	2 (read)
20001 ... 30000	Unused		
30001 ... 40000	16 bits (word)	INPUT REGISTERS[a]	4 (read)
40001 ... 50000	16 bits (word)	REGISTER OUTPUT[b]	3 (read), 6 (write), 16 (multiple write)

Table footnotes from previous page

[a] Device's variables that the control system can read but not modify.
[b] Device's variables that the control system can read and modify.

In order to permit a faster transmission of disturbance records, the Modbus communication board protocol reserves the Modbus range 39900 .. 39931 to the REF 542plus variable M31.

9.2. Modbus address configuration (Data Definitions)

As the Multifunction Protection and Switchgear Control Unit REF 542plus micro controller permits to access data only by means of SPA bus telegrams [1], it is necessary to define a mapping among Modbus addresses and SPA bus telegrams accepted by REF 542plus.

The mapping is carried out by means of configuration tables, composed by Data Definition records. Each Data Definition, abbreviated as DDEF is a 44-bytes record, with the following structure:

Field	Dimension (in bytes)	Meaning
DataAddress	2	0xFFFF = DDEF not valid 1..50000 = Modbus address
DataType	1	1 = AI (reading analog inputs) 2 = DI (reading digital inputs) 4 = AO (writing analog outputs) 8 = DO (writing digital outputs) 0x10 = WP (parameter "write protected") 0x20 = WC (command "write protected")
DataLength	1	1,2,4 : data dimension (in bytes) of each member in the DDEF
Polling	1	0 = Off 1 = LP (low priority) 2 = HP (high priority)
Reserved	1	
SPA-BUSTelegram	14	14 ASCII char including the ending character ('\0')
ScaleFactor	1	+3..-3 : scale factor: value multiplied by 10 (Scale Factor)

Table 7.-2 (Continued)

Field	Dimension (in bytes)	Meaning
Members	1	1..255 Number of REF542 elements involved in the DDEF
Name	20	The record identifier string (i.e., the name of the variable/s mapped in the record)
Class	2	The class of the variable mapped in the record, according to the following enumeration list: 0 = measure 1 = status 2 = command 3 = parameter 4 = reading/writing FUPLA variable 5 = event 6 = disturbance record 7 = information

9.3. DataAddress

It points out the starting address in the Modbus address space according to the classification described in the previous paragraph. The 0xFFFF address value means an unused DDEF.

As reading and writing operations on a given REF 542plus, that is REF 542plus variable, must have the same Modbus address, but different SPA bus telegrams (one for reading and one for writing). It is possible to have two DDEFs with the same DataAddress (or with range overloaded), but with different DataType direction.

9.4. DataType

It points out the type of each member and the data access direction:

AI each member is a readable analog variable[1)]; the member's value is required by means of the function 4, if it corresponds to a device input register (that is measurement, event, read-only Fupla variable, fault record) or by means of the function 3, if it corresponds to a device output register (that is parameter).

DI each member is a readable digital variable[2)]; the member's value is required by means of the function 2, if it corresponds to a device input status (that is digital input) or by means of the function 3, if it corresponds to a device coil (that is command, digital output).

1) The term "analog", in this context, means datum whose value is 1/2/4 bytes wide.
2) The term "digital", in this context, means datum whose value is 1 bit wide.

AO each member is a writable analog variable; the member's value is transferred by means of the function 6, if it corresponds to a single 2 byte's device output register (single writing) or by means of the function 16 in the other analog writing cases (multiple writing).

DO each member is a writable digital variable; the member's value is transferred by means of the function 5 in case of single device coil writing or by means of the function 15 in case of multiple device coil writing.

The DI, DO, AI and AO bits are mutually exclusive.

The WP bit can be used only with AO variables; when set to '1', it means that all the members involved in the DDEF (parameters) are to be modified only within a "writing parameter session".

The WC bit can be used only with DO variables; when set to '1', it means that all the members involved in the DDEF (commands) are modifiable only within a "writing command session".

9.5. DataLength

It indicates the dimension, in bytes, of each DDEF member

Referring to [3], the following rules are valid:

For US-type SPA bus variables (unsigned integer variables, format X{X{X{X}}}), DataLength must be 2. Each member is mapped into one Modbus register and is transmitted and received in a 16-bits integer format:

High byte	Low byte

For UL-type (unsigned long variables, format X{X{X{X}}}. X{X{X}}) and SL-type SPA bus variables (signed long variables, format {+} | - X{X{X{X}}}. X{X{X}}), DataLength should be 4. Each member is mapped into two Modbus registers. The communication board converts each member in a 4-bytes signed integer format before transmitting it. If a scale factor is defined for the members, such a factor is applied during the conversion. In the same way, the communication board receives members in the 4-bytes signed integer format. After reception, the communication board converts each member in the IEEE float format. If a scale factor is defined for the members, such a factor is applied during conversion. The transmission format is:

High word		Low word	
High byte	Low byte	High byte	Low byte

In order to use less Modbus addresses, UL/SL-type may be accessed with a 2-bytes DataLength. In this case, the communication board converts each member in a 2-bytes signed integer format before transmitting it. If a scale factor is defined for the members, such a factor is applied during conversion (possible truncations are not controlled). In the same way, the communication

board receives members in the 2-bytes signed integer format and converts each member in the IEEE float format. If a scale factor is defined for the members, such a factor is applied during conversion.

UL and SL-type may be mapped both on 1 and 2 Modbus registers, (DataLength 2 or 4) with and without the scale factor. In the old version, only 4 bytes DataLength were allowed.

For ST-type SPA bus object (string) or for not typed objects (that is variables M28 and M31 of disturbance records), DataLength must be 1 and the member field must be equal to the maximum length (in bytes) of the object. If the object length is odd, the member field must be extended to the next even value (word alignment). This way, the number of Modbus addresses used by the object is equal to the number of members divided by 2.

In each DDEF the total number of Modbus registers used is:

Data Length * Members / 2

9.6. Polling

It indicates the optional polling class to apply to all the variables involved in DDEF:

- HP (2) = high polling priority
- LP (1) = low polling priority
- OFF (0) = no polling

Polling variables are requested directly by the communication board to REF 542plus and stored inside the internal memory of the communication board. These requests are cyclic and asynchronous with regard to the control system requests. They allow the communication board to answer immediately to the control system, without waiting for interaction with REF 542plus.

The polling cycle follows the sequenceHP, HP, HP, LP, HP, HP, HP, LP, ……. (3 high priority polling and 1 low priority polling).

Only DDEFs with Data Type DI or AI can be defined as polled.

9.7. SPA bus telegram

It points out the SPA bus telegram allowing REF 542plus access to the variables mapped over the Modbus registers range.

The telegram format is equal to the “Header” format described in the chapter 3. Message Format of [3], unless the Slave number field which is omitted in the SPA bus telegram field.

The length of the SPA buss telegram is 14 characters, including the ending character (‘\0’). For a telegram whose length is less than 13 characters, the characters following the ending character (‘\0’) are not significant, but must always be inserted.

9.8. ScaleFactor

Indicates the optional scale factor (sf) to apply to the data exchanged with REF 542plus according to the following formulas:

MODBUSValue = REFValue * 10^{sf}

REFValue = MODBUSValue / 10^{sf}

The scale factor ranges between −3 and +3. If a DDEF has no scale factor, the field must be set to 0.

9.9. Members

Points out the number of homogeneous elements mapped on the Modbus area starting at DataAddress.

The total length of the Modbus area, in number of Modbus registers, is then (DataLength * Members) / 2.

Note that the SPA bus strings should be mapped on DDFEs with “DataLength = 1” and “Members = size of the string”.

9.10. Name

Name is a free string with a maximum of 20 characters identifying the record.

9.11. Class

Class is the data class owning the record according to the following list:

0 = measurement

1 = status

2 = command

3 = parameter

4 = reading/writing FUPLA variable

5 = event

6 = disturbance record

7 = information

DDEF example 1: (Applicable from Modbus firmware version 2.0) Referring to the list showed in [5], the following DDEF permits to read the actual currents L1,L2,L3 (variable IL123, SL-type, scale factor '/1000') from channel 1:			
	Value (in decimal)	dim.in bytes	Notes
DataAddress	30031	2	Start Modbus address = 30031
DataType	2 (= AI)	1	Reading "analog" data
DataLength	4	1	4 bytes per member
Polling	0 = OFF	1	No polling
Reserved		1	
SPABUS Telegram	"R1I1/3"	14	Read channel 1- I type from 1 to 3
ScaleFactor	3	1	/1000 scale factor to apply
Member	3	1	3 members; next valid MODBUS address is 30031 + (4*3) / 2 = 30037)
Name	"IL123"	20	
Class	0	2	Measurements
DDEF example 2: (Applicable from Modbus firmware version 2.0) The actual currents L1,L2,L3 may be addressed in a short form too: in this case only the integer part of the values is transferred			
	value (in decimal)	Dim in bytes	Notes
DataAddress	30031	2	Start Modbus address = 30031
DataType	2 (= AI)	1	Reading "analog" data
DataLength	2	1	2 bytes per member
Polling	0 = OFF	1	No polling
Reserved		1	

Table 7.-2 (Continued)

DDEF example 2: (Applicable from Modbus firmware version 2.0) The actual currents L1,L2,L3 may be addressed in a short form too: in this case only the integer part of the values is transferred			
SPABUSTelegram	“R1I1/3”	14	Same SPA Telegram
ScaleFactor	0	1	Decimal part is not required
Member	3	1	3 members; next valid Modbus address is 30031 + (2*3) / 2 = 30034)
Name	“IL123”	20	
Class	0	2	Measurements
DDEF example 3: (Applicable from Modbus firmware version 2.0) a DDEF corresponding to the REF542 software version string could be:			
	value(in decimal)	dim. in bytes	Notes
DataAddress	40001	2	Start Modbus address = 40001
DataType	2 (= AO)	1	Reading “analog” data
DataLength	1	1	1 byte per member
Polling	0 = OFF	1	No polling
Reserved		1	
SPABUSTelegram	“R0F2”	14	Read channel 0 − type F- element 2
ScaleFactor	0	1	No scale factor to apply
Member	22	1	String length = 21 bytes, + 1 byte to align to the word.
Name	“Software version”	20	
Class	7	2	Information

10. Data definition configuration

Starting from version V2.0 of the communication board the maximum number of DDEFs is 1000. All of them are 44 bytes wide (22 Modbus registers).

All the DDEFs are stored in the DDEF table, whose length is 22*1000 Modbus registers and is mapped into an extended memory file MODBUS (file 1) accessible by means of function 20 (reading) and function 21 (writing).

Within the file_1, the first DDEF has address 0; the second DDEF has address 22, and so on.

It is permitted, but not recommended, to modify a single field of a given DDEF.

The restrictions on reading and writing operations over DDEFs are described in the paragraph “Configuration” in Section 10.1. Modbus functions.

10.1. Modbus functions

10.1.1. Function 1 (read coil status)

Specification 3.1 of [1] is fully implemented, according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 1 message,

Field	Slave Adress	Function	CoilStart Number	Number Of Data
Dimension (in bytes)	1	1	2	2
Value	Xx	1	0 ... 9999[a]	1 ... 2000

[a)] Data of word (2 bytes) or long (4 bytes) type are transmitted in the “big-endian order”, that is “High_word − Low_word” and, inside a single word, “High_byte-Low_byte”.
[b)] Data of word (2 bytes) or long (4 bytes) type are transmitted in the “big-endian order”, that is “High_word − Low_word” and, inside a single word, “High_byte-Low_byte”.

the following operations are performed:

- Verify the Slave Address (broadcast not allowed).
- Verify Coil Start Number and Number Of Data limits.
- Verify that all the coil addresses are in the range 0..9999 (that is Coil Start Number = 9998 and Number Of Data = 10 is not a valid addressing).

- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if it exists one or more consecutive (and not overloaded) DDEFs whose addresses hold the range Coil Start Number + 1,[3)], .. Coil Start Number + Number Of Data + 1. All the DDEFs must have the DataType equals to DI.
- The values are taken from the internal memory for DDEFs ‘polled’ or are requested to REF542 for DDEFs ‘not polled’.

As many REF 542plus coils have 3 values;

0 = off

1 = on

9 = unused

a multiple addressing with at least one ‘unused’ coil generates an ILLEGAL_DATA_ADDRESS error answer. To avoid this behaviour, the control system may modify the accessing DDEF, reducing the access range (that is from R2I1/7 to R2I1/6 if the 7 coil is unused) or consider such coils as ‘analog variables’ and use function 3.

10.1.2. Function 2 (read input status)

Specification 3.2. of [1] is fully implemented, according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 2 message,

Field	Slave Address	Function	Input Start Number	Number Of Data
Dimension (in bytes)	1	1	2	2
Value	xx	2	0 ... 9999	1 ... 2000

the following operations are performed :

- Verify the Slave Address (broadcast not allowed).
- Verify Input Start Number and Number Of Data limits.
- Verify that all the input addresses are in the range 0..9999 (that is Input Start Number = 9998 and Number Of Data = 10 is not a valid addressing).

[3)] The Modbus protocol specifies that CoilNumbers start from 0, while addresses start from 1.

- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if there exists one or more consecutive (and not overloaded) DDEFs whose addresses hold the range InputStartNumber + 10001, .. InputStartNumber + NumberOfData + 10001. All the DDEFs must have the DataType equals to DI.
- The values are taken from the internal memory for DDEFs ‘polled’ or are requested to the REF 542plus for DDEFs ‘not polled’.

10.1.3. Function 3 (read output register)

Specification 3.3. of [1] is fully implemented, according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 3 message,

Field	Slave Address	Func-tion	Register Start Number	Number Of Data
Dimension (in bytes)	1	1	2	2
Value	xx	3	0 ... 9999	1 ... 125

the following operations are performed :

- Verify the Slave Address (broadcast not allowed).
- Verify Input Start Number and Number Of Data limits.
- Verify that all the input addresses are in the range 0..9999 (that is Input Start Number = 9998 and Number Of Data = 10 is not a valid addressing).
- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if there exists one or more consecutive (and not overloaded) DDEFs whose addresses hold the range RegisterStartNumber + 40001, .. RegisterStartNumber + NumberOfData + 40001. All the DDEFs must have the DataType equals to AI.
- The values are taken from the internal memory for DDEFs ‘polled’ or are requested to REF 542plus for DDEFs ‘not polled’.

The values received by REF 542plus are scaled (and broken to the integer value) according to the optional scale factor (sf) defined inside the DDEF: Modbus Value = REF 542plus Value * 10^{SF}.

10.1.4. Function 4 (read input register)

Specification 3.4. of [1] is fully implemented according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 4 message,

Field	Slave Address	Function	Register Start Number	Number Of Data
dimension (in bytes)	1	1	2	2
value	xx	4	0 ... 9999	1 ... 125

the following operations are performed:

- Verify the Slave Address (broadcast not allowed).
- Verify Register Start Number and Number Of Data limits.
- Verify that all the register addresses are in the range 0..9999 (that is Register Start Number = 9998 and Number Of Data = 10 is not a valid addressing).
- If the limits are not respected, an error message is sent back indicating ILLEGAL_DATA_ADDRESS error.
- Verify if there exists one or more consecutive (and not overloaded) DDEFs whose addresses hold the range Register Start Number + 30001, .. Register Start Number + Number Of Data + 30001. All the DDEFs must have the DataType equals to AI.
- The values are taken from the internal memory for DDEFs ‘polled’ or are requested to REF542 for DDEFs ‘not polled’.

The values received by REF 542plus are scaled (and broken to the integer value) according to the optional scale factor (sf) defined inside the DDEFModbus Value = REF 542plus Value * 10^{SF}.

Reading the REF 542plus variable M31 (that is a single line of a disturbance record), whose predefined Modbus address is 39900, the received bytes are packed transforming each pair of ASCII chars (2 bytes) into the corresponding binary value (1 byte). That is, the pair “5D” is transformed into the byte 0x5D.

10.1.5. Function 5 (preset single coil)

Specification 3.5. of [1] is fully implemented, according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 5 message,

Field	Slave Address	Function	Coil Number	Value
Dimension (in bytes)	1	1	2	2
Value	xx	5	0 ... 9999	0x0 (OFF) or oxFF00 (ON)

the following operations are performed :

- Verify the Slave Address (broadcast is allowed).
- Verify Coil Number and Value limits.
- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if there exists one DDEF whose address is Coil Number + 1; the DDEF must have the DataType equals to DO.

If the bit WC of the DDEF is set and the control system is not within a writing command session, an error message is sent back, indicating ILLEGAL_FUNCTION error.

Otherwise, the value is converted (0/1) and transferred to REF 542plus and a Modbus message answer is sent back after the REF 542plus operation.

10.1.6. Function 6 (preset single register)

Specification 3.6. of [1] is fully implemented according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 6 message,

Field	Slave Address	Function	Register Number	Value
Dimension (in bytes)	1	1	2	2
Value	xx	6	0 ... 9999	0x0 ... 0xFFFF

the following operations are performed :

- Verify the Slave Address (broadcast is allowed).
- Verify RegisterNumber limits.
- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if it exists one DDEF whose address is Coil Number + 40001; the DDEF must have the DataType equals to AO.

- If the bit WP of the DDEF is set and the control system is not within a writing parameter session, an error message is sent back, indicating ILLEGAL_FUNCTION error.
- Otherwise, the value is converted (0/1) and transferred to REF 542plus and a Modbus message answer is sent back after the REF 542plus operation.

Before being transferred to REF 542plus, the value is scaled according to the optional scale factor (sf) defined inside the DDEF. Modbus Value = REF 542plus Value * 10^{SF}.

As function 6 can address only a single Modbus register (2 bytes), it is necessary to use function 16 in order to preset a 4-bytes SPA bus parameter.

10.1.7. Function 7 (read exception status)

The read exception status code is designed to allow the control system to control certain events in the communication board in order to ensure correct operations.

The communication board handles the following events:

Position	Meaning
0x80 (bit 8)	1 = Communication board operating: REF 542plus is connected and working 0 = Communication board not operating
0x40 (bit 7)	1 = All the DDEFs are correct 0 = Not all the DDEFs are correct
0x20 (bit 6)	1 = The polling list is correct 0 = The polling list is not correct
0x10 (bit 5)	1 = The Modbus connection is active on port 2 0 = The Modbus connection is active on port 1

When the communication board receives a Function 7 message,

Field	Slave Address	Function
dimension (in bytes)	1	1
Value	xx	7

the following operations are performed :

- Verify the Slave Address (broadcast not allowed).
- Pack the 4 events.
- Sent back the response, in case the communication board is correctly operating and has the following format:

Field	Slave Address	Function	CoilValue
Dimension (in bytes)	1	1	1
Value	xx	7	0xE0/0xF0

10.1.8. Function 8 (loopback test)

Specification 3.8. of [1] allows the control system to control the communication subsystem by means of several subcodes. The communication board implements a subset of such subcodes, as showed in the following table:

Subcode	Meaning	Implemented
0	Return Query Data (that is echo)	yes
1	Restart Comm Option	yes
2	Return Diagnostic Register	yes
3	Change Input Delimiter Character	no
4	Force Slave to Listen Only Mode	yes
10	Clear Counter and Diagnostic Register	yes
11	Return Bus Message Count	yes
12	Return Bus CRC Error Count	yes
13	Return Bus Exception Error Count	yes
14	Return Slave Message Count	yes
15	Return Slave No Message Count	yes
16	Return Slave NAK Count	no
17	Return Slave Busy Count	no
18	Return Bus Character Overrun Count	yes
19	Return Overrun Error Count	yes
20	Clear Overrun Error Count and Flag	yes
> 20		no

When the communication board receives a Function 8 message,

Field	Slave Address	Function	DiagCode	DiagData
Dimension (in bytes)	1	1	2	2
Value	xx	8	0 ... 20	0 ... 0xFFFF

the following operations are performed :

- Verify the Slave Address (broadcast not allowed) and the DiagData field
- Send back the response with the following format:

Field	Slave Address	Function	DiagCode	DiagData
Dimension (in bytes)	1	1	2	2
Value	xx	8	0 ... 20	0 ... 0xFFFF

10.1.9. Function 11 (fetch communications event counter)

This function is not implemented.

10.1.10. Function 12 (fetch communications event log)

This function is not implemented.

10.1.11. Function 15 (force multiple coils)

Specification 3.11. of [1] is fully implemented according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 15 message,

Field	Sl. Addr.	Function	Coil Start Number	N.Coils	N. Bytes	Byte1	…	Byte N
Dimension (in bytes)	1	1	2	2	1	1		1
Value	Xx	15	0 ... 9999	1 ... 10000	1.255	any		any

the following operations are performed :

- Verify the Slave Address (broadcast is allowed).
- Verify the values of the fields Coil Start Number, N. Coils and N. Bytes: the number of bytes must be compatible with the number of coils (8 coils for byte). The total address range must be within the range 0..9999.
- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if there exists one or more consecutive (and not overloaded) DDEFs, whose addresses hold the range Coil Start Number + 1, .. Coil Start Number + N. Coils + 1. All the DDEFs must have the Data Type equals to DO.
- If the bit WC of at least one DDEF is set and the control system is not within a writing command session, an error message is sent back, indicating ILLEGAL_FUNCTION error.
- Otherwise, the values are transferred to REF 542plus and a Modbus message answer is sent back after the REF 542plus operation. In case of success, the answer has the following format:

Field	Slave Address	Function	CoilStartNumber	N.Coils
Dimension (in bytes)	1	1	2	2
Value	xx	15	As in the input message	As in the input message

10.1.12. Function 16 (preset multiple registers)

Specification 3.12. of [1] is fully implemented according to the addressing range defined by the DDEF table currently configured.

When the communication board receives a Function 16 message,

Field	Sl. Addr.	Function	Reg Start Number	N. Regs	N. Bytes	Reg1		Reg N
Dimension (in bytes)	1	1	2	2	1	2		2
Value	Xx	16	0 ... 9999	1 ... 60	2.120	any		any

the following operations are performed :

- Verify the Slave Address (broadcast is allowed).
- Verify the values of the fields Reg Start Number, N. Regs and N. Bytes: the number of bytes must be compatible with the number of registers. The total address range must be within the range 0..9999.
- If the limits are not respected, an error message is sent back, indicating ILLEGAL_DATA_ADDRESS error.
- Verify if there exists one or more consecutive (and not overloaded) DDEFs whose addresses hold the range Reg Start Number + 40001, .. Reg Start Number + N. Regs + 40001. All the DDEFs must have the DataType equals to AO.
- If the bit WP of the DDEF is set and the Control System is not within a writing parameter session, an error message is sent back indicating ILLEGAL_FUNCTION error.
- Otherwise, the values are transferred to REF 542plus and a Modbus message answer is sent back after the REF 542plus operation. In case of success, the answer has the following format:

Field	Slave Address	Func-tion	RegStart Number	N.Regs
Dimension (in bytes)	1	1	2	2
Value	xx	16	As in input message	As in input message

Before being transferred to REF 542plus, all the values are scaled according to the optional scale factor (sf) defined inside the DDEFModbus Value = REF 542plus $\boldsymbol{Value * 10}_{SF}$.

10.1.13. Function 17 (report slave ID)

According to the specification, the control system is allowed to ask Modbus slave about the type, the operation light status and other specific information of the protection device.

When the communication board receives a Function 17 message (broadcast not allowed),

Field	Slave Address	Function
Dimension (in bytes)	1	1
Value	Xx	17

an answer message is sent back with the following format:

Field	Slave Address	Func-tion	Byte-Count	Slave Id	Runlight	Device-Depen-ded
dim	1	1	1	1	1	20
value	Xx	17	2	0x32	0x00 = off 0xFF = on	reserved

The function 17 may be accessed also by using the pseudo-broadcast address 255:

Field	Slave Address	Function
Dimension (in bytes)	1	1
Value	255	17

In this case, the answer message has the following format:[4)]

Field	Slave Address	Func-tion	Byte-Count	Slave Address[a]	Version string
dim	1	1	1	1	10
value	Xx	17	11	Xx	Software Version string[b]

[a)] The slave address is replied inside the message according to the need of some configuration tools.
[b)] For version V2.6 the version string is “AP_V2.6 “

10.1.14. Function 19 (reset communications link)

This function is not implemented.

10.1.15. Function 20 (read general reference)

Specification 3.14 of [1] is fully implemented over the following extended memory files, restricted to the read-only (‘r’) and read-write (‘r/w’) registers.

File 1: Data Definition table

File 2: RCE table (REF542 historical register of events)

File 3: General Register table

File 4: Serial Channel Descriptor table

File 5: RCE extended table (REF542 historical register of events in extended form)

[4)] This behaviour is not compliant with [1].

File 6: Configuration and Program Descriptor table

File 7:	Time and Date
File 8:	Program Command table
File 9:	Program Image
File 10:	Configuration and Program Descriptor table (Not deleted with the rest of the board)
File 11:	(RCE table): as File 2 - Read and Clear of Events
File 12:	(RCE extended table): as File 5 - Read and Clear of Extended Events

The file structures are the following:

Table 10.1.15.-1 File 1 (DataDefinition table)

Register	Meaning	Direction
0 ... 21	Data Definition 1	r/w
22 ... 43	Data Definition 2	r/w
...		
(1000*22) ... (1000*22) -1	Data Definition 1000	r/w
(1000*22) ... 0xFFFF	Unused	error

The format of each DDEF has been defined in
Section 9.2. Modbus address configuration (Data Definitions).

The dimension of the file 1 is 1000*22 Modbus registers. It is not compliant with [1] that limits a file to 10000 registers.

Table 10.1.15.-2 File 2 (RCE table)

Register	Meaning	Direction
0	Number of available events	r
1	Overflow indicator	r
2	Clear RCE	w
3	Event 1	r
…		
D+2		

Table 10.1.15.-2 File 2 (RCE table) (Continued)

Register	Meaning	Direction
D+3	Event 2	r
…		
(D*2) + 2		
…	…	...
(D*(MAX-1)) + 3	Event MAX	r
…		
(D*MAX) + 2		
(D*MAX) + 3	Unused	error
…	…	...
9999	Unused	error

With D = sizeof(Event)/2 = 14 bytes2 = 7 word

MAX = 100

The full description of the RCE table is in Section 11.1.15. Reading events.

Table 10.1.15.-3 File 3 (General Register table)

Register	Pos.	Meaning	Direction
Session Config	0	0 = Request of ending configuration session using current values 1 = Request of starting configuration session 2 = Request of ending configuration session with abort	w
Session Parameter	1	0 = Request of ending parameter session 1 = Request of starting parameter session	w
Session Command	2	0 = Request of ending command session 1 = Request of starting command session	w
Status Config	3	0 = No configuration session is active 1 = Configuration session is active on channel 1 2 = Configuration session is active on channel 2	r

Table 10.1.15.-3 File 3 (General Register table) (Continued)

Register	Pos.	Meaning	Direction
Status Parameter	4	0 = No parameter session is active 1 = Parameter session is active on channel 1 2 = Parameter session is active on channel 2	r
Status Command	5	0 = No command session is active 1 = Command session is active on channel 1 2 = Command session is active on channel 2	r
Session Programming	6	0 = Request of ending programming session using current values 1 = Request of starting programming session 2 = Request of ending programming session with abort	w
Status Programming	7	0 = No programming session is active 1 = Programming session is active on channel 1 2 = Programming session is active on channel 2	r
8 ... 9999	8 ... 9999	unused	error

The full description of the General Registers is in Section 11.1.23. Sessions.

Table 10.1.15.-4 File 4 (Serial Channel Descriptor table)

Channel	Register	Pos.	Values	Direction
1	Channel-Type	0	0 = RS485 (default)[a] 1 = optical fibre 2 = unused	r/w
	SlaveAd-dress	1	1 ... 247 = MODBUS Slave address (default = 99)	r/w
	BaudRate	2	0 = 300 1 = 600 2 = 1200 3 = 2400 4 = 4800 5 = 9600 6 = 19200 (default) 7 = 38400 8 = 76800 9 = 153600	r/w
	NumBits	3	8 = 8 data bits (fixed value)	r/w
	Parity	4	0 = no parity (default) 1 = even parity 2 = odd parity	r/w
	NumStop-Bit	5	0 ... 2 (default = 1)	r/w
	Timeout	6	3 = number of characters timeout (fixed value)	r/w
	Reserved	7 ... 9	0	n/a

Table 10.1.15.-4 File 4 (Serial Channel Descriptor table) (Continued)

Channel	Register	Pos.	Values	Direction
2	Channel-Type	0	0 = RS485 (default) 1 = optical fibre2 = unused	r/w
	SlaveAd-dress	1	1 ... 247 = MODBUS Slave address (default = 99)	r/w
	BaudRate	2	0 = 300 1 = 600 2 = 1200 3 = 2400<para ishresult="show">4 = 4800 5 = 9600 6 = 19200 (default) 7 = 38400 8 = 76800 9 = 153600	r/w
	NumBits	3	8 = 8 data bits (fixed value)	r/w
	Parity	4	0 = no parity (default) 1 = even parity 2 = odd parity	r/w
	NumStop-Bit	5	0 ... 2 (default = 1)	r/w
	Timeout	6	3 = number of characters timeout (fixed value)	r/w
	Reserved	7 ... 9	0	n/a
	Unused	20 ... 9999		error

[a)] Note : the channel type configuration has no effect on the Modbus board.

The full description of the Serial Channel Descriptor table is in Section 11.1.2. Serial Channel Descriptor table configuration.

Table 10.1.15.-5 File 5 (RCE Extended table)

Register	Meaning	Direction
0	Number of available events	r
1	Overflow indicator	r
2	Clear RCE	w

Table 10.1.15.-5 File 5 (RCE Extended table) (Continued)

Register	Meaning	Direction
3	Event 1	r
…		
D+2		
D+3	Event 2	r
…		
(D*2) + 2		
…	…	…
(D*(MAX-1)) + 3	Event MAX	r
…		
(D*MAX) + 2		
(D*MAX) + 3	Unused	error
…	…	…
9999	Unused	error

With D = sizeof(Event)/2 = 22 bytes2 = 11 word

MAX = 100

The full description of the RCE extended table is in Section 11.1.16. Reading events in extended form.

Table 10.1.15.-6 File 6 (Configuration and Program Descriptor table)

Register	Meaning	Direction
0 ... 9	Application FW version	r
10 ... 19	BootLoader version	r
20 ... 29	Configurator version	r/w
30 ... 39	Configuration time and date	r/w
40 ... 49	Configuration Filename	r/w
50 ... 59	Configuration file time and date	r/w
60 ... 9999	Unused	error

The full description of the Configuration and Program Descriptor table is in Section 11.1.3. Configuration and program description.

Table 10.1.15.-7 File 7 (Time and Date)

Register	Meaning	Direction
0 ... 3	Time and date	r/w
4 ... 9999	Unused	error

The full description of the Time and Date file is in Section 11.1.20. Time synchronization.

Table 10.1.15.-8 File 8 (Program Command table)

Register	Meaning	Direction
0 ... 1	Program checksum	r
2 ... 3	Program start address	w
4 ... 5	Program length	w
6 ... 9999	Unused	error

The full description of the Program command table is in Section 11.1.22. Re-programming the communication board.

Table 10.1.15.-9 File 9 (Program Image)

Register	Meaning	Direction
0 ... MAX-1	Program image	w
MAX ... 9999	Unused	error

Where MAX = 2032

The method of accessing the file is not compliant with [1]. Each register addresses a virtual stream of 64 bytes, and therefore the maximum program length is 2032*64 = 127Kbytes.

The full description of the Program image file is in Section 11.1.22. Re-programming the communication board

Table 10.1.15.-10
File 10 (Program Image)

Register	Meaning	Direction
0 ... 9	Board serial number	r/w
10 ... 19	Production time and date	r
20 ... 29	Main board serial number	r
30 ... 9999	Unused	error

File 11 (RCE table): as File 2 - Read and Clear of Events

File 12 (RCE extended table): as File 5 - Read and Clear of Extended Events

10.1.16. Function 21 (write general reference)

Specification 3.14 of [1] is fully implemented over the extended memory files described in Section 10.1.15. Function 20 (read general reference) restricted to the write-only (‘w’) and read-write (‘r/w’) registers.

11. REF 542plus data handling

11.1. Communication board configuration

Communication board configuration means sending or modifying file registers relevant to:

Data Definition table

Serial Channel Descriptor table

In order to perform writing operation on such registers, it is necessary that the control system has been enabled to a configuration session, see Section 11.1.23. Sessions.

Registers modified during a configuration session are actually used only at the end of the session. If the session ends with an ‘abort’ request, all the modifications are lost and the communication board configuration remains the one defined before the starting of the communication session.

11.1.1. Data Definition table configuration

In order to send or receive the DDEF table or part of it, it isnecessary to access to the File 1 registers, with the following structure:

Register	Meaning	Direction
0 ... 10	Data Definition 1	r/w
11 ... 21	Data Definition 2	r/w
...		
(254*11)... (255*11)-1	Data Definition 255	r/w
(255*11) ... 9999	Unused	error

Each record in File 1 stores one DDEF, whose structure is defined in Section 9.2. Modbus address configuration (Data Definitions).

To configure the complete DDEF table, the control system must perform the following procedures:

- To request the start of the configuration session.
- To make x-cycles (where ***x*** is the number of DDEFs) sending one DDEF at a time.

11.1.2. Serial Channel Descriptor table configuration

The communication board can be connected to one or two control systems by means of its two serial channels. The characteristics of each serial channel are configurable accessing to the File 4 registers.

The File 4 structure is described in
Section 10.1.15. Function 20 (read general reference).

Writing on each register at a time, or more efficiently, writing the entire descriptor with a single message can perform a serial channel configuration. In any case, the writing operation is allowed only within a configuration session.

For example, in order to configure channel 2 with RS485, 19200 baud, no parity, 1 stop bit, 8 data bits, 3 characters timeout and 1 as Modbus Slave Address, the control system may use the following message:

Field	**Slave Address**	**Function**	**Byte Count**	**Ref Type**	**File number**	**Req Address**	**Reg Count**	**…**
Dimension	1	1	1	1	2	2	2	
Value	Xx	21	21	6	4	10	7	

…	**Data1 (PortType)**	**Data2 (Address)**	**Data3 (BaudRate)**	**Data4 (NumBits)**	**Data5 (Parity)**	**Data6 (StopBits)**	**Data7 (Timeout)**
	2	2	2	2	2	2	2
	0	1	6	8	0	1	3
	(rs485)	(1)	(19200)	(8 bit)	(no parity)	(1 stop bit)	(3 chars)

The communication board, in case of success, answers with the complete echo of the received message.

After ending a configuration session, if a channel descriptor has been modified, the communication channel will be restarted with the new parameters. During this operation, the communication will be lost for a few seconds. This is true even if the modified channel descriptor is not the channel on which the configuration is running. Be careful during configuring the channel descriptors when both channels are in use.

To read a channel descriptor or a part of it, the control system must use function 21 as in the following example (reading channel 2 descriptor):

Field	Slave Address	Function	Byte-Count	Ref-Type	File-number	Req-Address	Reg-Count	…
Dimension	1	1	1	1	2	2	2	
Value	Xx	21	7	6	4	10	7	

The communication board answers with a similar message:

Field	Slave Address	Function	Byte Count	Sub resp byte Count	Type ref	…
Dimension	1	1	1	1	1	
Value	xx	21	16	15	6	

…	Data1 (Port-Type)	Data2 (Address)	Data3 (BaudRate)	Data4 (Num-Bits)	Data5 (Parity)	Data6 (Stop-Bits)	Data7 (Time-out)
	2	2	2	2	2	2	2
	0	1	6	8	0	1	3
	(rs485)	(1)	(1920-0)	(8 bit)	(no parity)	(1 stop bit)	(3 chars)

Reading the Channel Descriptor table or a part of it may be performed outside the configuration session too.

11.1.2.1. Modbus slave address configuration through the main board

The SlaveAddress field of one or both channels may also be configured by using the main board configuration.

When the communication board is notified by the main board that a new Modbus Slave address(es) is(are) available on the common interface, the communication board overwrites the old(s) address(es) in the file 4.

The new address(es) may be read by using function 21 as usual.

11.1.3. Configuration and program description

The extended file 6 and 10 contains information describing the configuration and the programming of the communication board and REF542 containing it:

The first table (File 6) contains information that is deleted with the board reset, while the second table (File 10) contains information that remains with the board reset.

Register	Meaning	Direction
0 ... 9	Application FW version	r
10 ... 19	BootLoader version	r
20 ... 29	Configurator version	r/w
30 ... 39	Configuration time and date	r/w
40 ... 49	Configuration Filename	r/w
50 ... 59	Configuration file time and date	r/w
60 ... 9999	Unused	error
Register	**Meaning**	**Direction**
0 ... 9	Board serial number	r/w
10 ... 19	Production time and date	r
20 ... 29	Main board serial number	r
30 ... 9999	Unused	error

The meaning of the fields is the following:

Application FW version	It is the version string of the application program resident on the communication board; it is automatically available after start-up. The string is 0-terminated.
BootLoader version	It is the version string of the boot loader program resident on the communication board; it is automatically available after start-up. The string is 0-terminated.
Configurator version	It is the version string of the configuration tool that downloaded the last DDEF table. The value is written by the configuration tool itself or by other master Modbus tools. The string should be 0-terminated.
Configuration time and date	It is the absolute time (in the string format gg/mm/aaaa hh:mm:ss) of the last DDEF table downloading. The value is written by the configuration tool or by other master Modbus tools. The string should be 0-terminated. The communication board does not perform any control on the format received.

Configuration filename	It is the name of the file containing (in the host environment) the last DDEF table downloading. The value is written by the configuration tool or by other master Modbus tools. The string should be 0-terminated.
Configuration file time and date	It is the absolute time (in the string format gg/mm/aaaa hh:mm:ss) of the last DDEF table modification in the host environment. The value is written by the configuration tool or by other master Modbus tools. The string should be 0-terminated. The communication board does not perform any control on the format received.
Board serial number	It is the serial number of the communication board (string format). The value is written by the configuration tool or by an other master Modbus tools. The string should be 0-terminated.
Production time and date	It is the absolute time (in the string format gg/mm/aaaa hh:mm:ss) of REF 542plus containing the communication board. The value is automatically available after start-up.
Main board serial number	It is the serial number of the main board of the REF 542plus containing the communication board. The value is automatically available after start-up.

All strings are 20 bytes long.

All the writable fields are optional as the communication board does not use any of them. In any case, the writing operations are allowed only within a configuration session.

All fields may be read and/or written by using the function 20 and 21 messages on file 6.

For example, in order to set the Configuration Time and Date field, the control system may use the following message:

Field	**Slave Address**	**Function**	**Byte Count**	**Ref Type**	**File number**	**Req Address**	**Reg Count**	**…**
Dimension	1	1	1	1	2	2	2	
Value	Xx	21	27	6	6	30	10	
…				**Data1..20**				
				20				
				“12/04/2002 14:55:50”				

The communication board, in case of success, answers with the complete echo of the received message. Anyway, the string is permanently stored only after a “stop configuration” command.

11.1.4. Reading measurements

In order to access to a given REF 542plus measurement in reading mode, a DDEF with DataType AI and Start Modbus Address in the range 30001..40000 must be configured. The same DDEF may map more than one measurement at a time.

It is possible:

- To read a single measurement.
- To read all the measurements mapped in one DDEF.
- To read measurements mapped in more than one consecutive DDEFs.

Reading is requested by means of a function 4 message applied to the Start Modbus Address of the first DDEF.

If the DDEF maps more than one measurement, it is not possible to request the value of a single measurement, unless the first measurement in DDEF.

For each DDEF having a scale factor, all the measurements belonging to it are scaled before sending them to the control system. The returned values are always in signed integer format (2 or 4 bytes).

Example

If the actual currents IL1, IL2, IL3 of channel 1 are mapped by:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
Value	30031	AI	4	LP	“R1I1/3”	3	3

A reading request will be:

Field	Slave Address	Function	Register Start Number	Number Of Regs
dimension (in bytes)	1	1	2	2
Value	xx	4	30	6

The answer could be:

Field	Address	Function	Byte Count	Data 1 (30031)	Data 2 (30033)	Data 3 (30035)
dimension (in bytes)	1	1	2	4	4	4
Value	xx	4	12	AAAA	BBBB	CCCC

That means the REF 542plus current values are:

IL1 = AAAA / 1000

IL2 = BBBB / 1000

IL3 = CCCC / 1000.

11.1.5. Reading input states

In order to access in reading mode to a given REF 542plus input, a DDEF with Data Type DI and Start Modbus Address in the range 10001..20000 must be configured.

The same DDEF may map more than one input at a time.

It is possible:

- To read a single input.
- To read all the inputs mapped in one DDEF.
- To read inputs mapped in more than one consecutive DDEFs.

Reading is requested by means of a function 2 message applied to the Start Modbus Address of the first DDEF.

If the DDEF maps more than one input, it is not possible to request the value of a single input, unless it is the first input in DDEF.

Example

If the 7 digital inputs of the input_board_1 are mapped by:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
dim	2	1	1	1	14	1	1
value	10001	DI	2	HP	“R2I1/7”	0	7

A reading request for the 7 inputs will be:

Field	Address	Function	Input Start Number	Number Of Input
Dimension (in bytes)	1	1	2	2
Value	xx	2	0	7

The answer could be:

Field	Address	Function	Byte Count[a]	Data 1 (10001)
Dimension (in bytes)	1	1	2	1
Value	xx	2	1	0x2F

[a] A single byte is sent, even if the register length is 2 bytes.

That means the REF 542plus input_board_1 inputs are:

Input	7	6	5	4	3	2	1	0
Data 1	0	0	1	0	1	1	1	1
Status	n/a	Off	On	Off	On	On	On	On

11.1.6. Writing commands

In order to access to a given REF 542plus command[5] in writing mode, a DDEF with Data Type DO, Start Modbus Address in the range 1..10000 must be configured.

The commands can be protected (in this case the WC bit is set) or not protected. It is possible to preset one or more protected commands only within a command session. Otherwise, the request ends with an “ILLEGAL FUNCTION” message error.

The same DDEF may map more than one command at a time.

It is possible:

- To preset a single command, refer to Section 10.1.5. Function 5 (preset single coil).
- To preset all the commands mapped in one DDEF, refer to Section 10.1.11. Function 15 (force multiple coils).
- To preset commands mapped in more than one consecutive DDEFs, all with WC set, refer to Section 10.1.11. Function 15 (force multiple coils).

[5] The term command means the setting of a digital output (coil).

If the DDEF maps more than one command, it is not possible to preset the value of a single command, unless it is the first command in DDEF.

Example

If the coils O1 and O2 of channel 10 are mapped by:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
O1	30	DO & WC	2	Off	“W10-O1”	0	1
O2	31	DO & WC	2	Off	“W10-O2”	0	1

The closing command (status = ON) on the single O1 coil will be:

Field	Address	Function	Coil Number	Value
dimension (in bytes)	1	1	2	2
Value	Xx	5	29	0xFF00[a]

a) Remember that ON = 0xFF00 and OFF = 0x0

The successful answer is the message echo:

Field	Address	Function	Coil Number	Value
dimension (in bytes)	1	1	2	2
Value	xx	5	29	0xFF00

The closing command on both coils will be:

Field	Address	Function	Coil Start Number	N. Coils	N. bytes	Coils value
dimension (in bytes)	1	1	2	2	1	1
Value	xx	15	29	2	1	0x3

The successful answer is the partial message echo:

Field	Address	Function	Coil Start Number	N. Coils
dimension (in bytes)	1	1	2	2
Value	xx	15	29	2

11.1.7. Reading parameters

In order to access to a given REF 542plus parameter in reading mode, a DDEF with DataType AI and Start Modbus Address in the range 40001 ... 50000 must be configured The same DDEF may map more than one parameter at a time.

It is possible:

- To read a single parameter.
- To read all the parameters mapped in one DDEF.
- To read parameters mapped in more than one consecutive DDEFs.

Reading is requested by means of a function 3 message applied to the Start Modbus Address of the first DDEF.

If the DDEF maps more than one parameter, it is not possible to request the value of a single parameter, unless the first parameter in DDEF.

For each DDEF having a scale factor, all the parameters belonging to it are scaled before sending them to the control system. The returned values are always in signed integer format (2 or 4 bytes).

Example

If the parameters “Overcurrent value Set 1” and “Overcurrent op. time Set 1” are mapped by:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
Value	40047	AI	4	Off	“R51-S1/2”	3	2

A reading request for the 2 parameters will be:

Field	Address	Function	Register Start Number	Number Of Regs
Dimension (in bytes)	1	1	2	2
Value	xx	3	46	4

The answer could be:

Field	Address	Function	Byte Count	Data 1 (40047)	Data 2 (40049)
Dimension (in bytes)	1	1	2	4	4
Value	xx	3	8	AAAA	BBBB

That means the REF 542plus parameter values are:

- “Overcurrent value Set 1” = AAAA / 1000
- “Overcurrent op. time Set 1” = BBBB / 1000

11.1.8. Writing parameters

In order to access in writing mode to a given REF 542plus parameter, a DDEF with DataType AO and Start Modbus Address in the range 40001 ... 50000 must be configured.

The parameters can be protected (in this case the WP bit is set) or not protected. It is possible to preset one or more protected parameters only within a parameter session. Otherwise, the request ends with an “ILLEGAL FUNCTION” message error.

The same DDEF may map more than one parameter at a time.

It is possible:

- To pre-set a single parameter, refer to Section 10.1.6. Function 6 (preset single register).
- To pre-set all the parameters mapped in one DDEF, refer to Section 10.1.12. Function 16 (preset multiple registers) .
- To pre-set parameters mapped in more than one consecutive DDEFs, all with WP set, refer to Section 10.1.12. Function 16 (preset multiple registers) .

If the DDEF maps more than one parameter, it is not possible to pre-set the value of a single parameter, unless it is the first parameter in DDEF.

For each DDEF having a scale factor, all the parameters belonging to it are scaled before transferring them to REF 542plus. The transmitted values are in signed integer format (if 2 bytes wide) or in IEE float format (if 4 bytes wide).

Example

To modify with a single preset and with a multiple preset the two parameters “Overcurrent value Set 1” (OvS1) and “Overcurrent op. time Set 1”, (OvS2) already defined in Section 11.1.7. Reading parameters, it is necessary to add two DDEFs as follows:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
OvS1	40047	WP|AO	4	Off	“W51-S1”	3	1
OvS2	40049	WP|AO	4	Off	“W51-S2”	3	1

A single preset request (for OvS1) could be:

Field	Address	Function	Reg Number	Value
Dimension (in bytes)	1	1	2	4
Value	xx	6	46	AAAA

(The value AAAA is divided by 100 before sending it to REF 542plus)

In case of success, the communication board answers with the echo of the message:

Field	Address	Function	Reg Number	Value
Dimension (in bytes)	1	1	2	4
Value	xx	6	46	AAAA

A multiple preset request could be:

Field	Address	Function	RegStart Number	N. Regs	N. bytes	Reg 1/2	..Reg 3/4
Dimension (in bytes)	1	1	2	2	1	4	4
Value	xx	16	46	4	8	AAAA	BBBB

(The values AAAA and BBBB are divided by 1000 before sending them to REF 542plus)

In case of success, the communication board answers with the partial echo of the message:

Field	Address	Function	Reg Start Number	N.Regs
Dimension (in bytes)	1	1	2	2
Value	xx	16	46	4

11.1.9. Reset alarms

The alarm reset is treated as any command on the REF 542plus variable O100, channel 101.

The corresponding DDEF could be:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
Value	22	DO & WC	2	Off	“W101-O100”	0	1

with a reset command like:

Field	Address	Function	Coil Number	Value
Dimension (in bytes)	1	1	2	2
Value	xx	5	21	0xFF00

In case of success, the communication board answers with the echo of the message:

Field	Address	Function	CoilNumber	Value
Dimension (in bytes)	1	1	2	2
Value	xx	5	21	0xFF00

The same rules are valid also for the other kinds of reset:

- Reset maximum bar, variable O102 of channel 101.
- Reset energy, variable O103 of channel 101.
- Reset switch cycles, variable O104 of channel 101.

11.1.10. Reading/writing FUPLA variables

Two cases are considered:

- Access to a read-type variable
- Access to a write-type variable

11.1.11. Read-type variables

They are the variables I1 ... I99 of channel 101 (read-only variables). Their handling is similar to the case of reading measurements. Therefore, to read read-type FUPLA variables, the control system must use functions 4 over the address range 30001 ... 40000.

Example

If the 4 variables I10 ... I13 are mapped by:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
Value	30101	AI	2	LP	“R101-I10/13”	0	4

A reading request will be:

Field	Address	Function	Register Start Number	Number Of Regs
dimension (in bytes)	1	1	2	2
Value	xx	4	100	4

A possible communication board answer is:

Field	Address	Function	Byte Count	Data 1 (3010-1)	Data 2 (3010-2)	Data 3 (3010-3)	Data 4 (3010-4)
dimension (in bytes)	1	1	2	2	2	2	2
Value	xx	4	8	AAAA	BBBB	CCCC	DDDD

That means the REF 542plus values are:

I10 = AAAA; I11 = BBBB; I12 = CCCC; I13 = DDDD.

11.1.12. Write-type variables

They are the variables O1 ... O99 of channel 101 (write-only variables). Their handling is similar to the case of reading parameters. Therefore, to preset write-type FUPLA variables, the control system must use functions 6 and 16 over the address range 40001 ... 50000.

The setting of the WP bit is not mandatory.

Example

If the 4 variables O10 ... O13 are mapped by:

Field	Data Address	Data Type	Data Length	Polling	SPA-BUS Telegram	Scale Factor	Members
Dim	2	1	1	1	14	1	1
Value	40101	AO	2	off	“W101-O10/13”	0	4

A multiple preset request could be:

Field	Address	Function	Reg Start Number	N. Regs	N. bytes	Reg 1	Reg 2	Reg 3	Reg 4
Dimension	1	1	2	2	1	2	2	2	2
Value	Xx	16	100	4	8	AAAA	BBBB	CCCC	DDDD

In case of success, the communication board answers with the partial echo of the message:

Field	Address	Function	Reg Start Number	N. Regs
Dimension (in bytes)	1	1	2	2
Value	xx	16	100	4

11.1.13. Reading system parameters

The reading access to the system parameters (see “System parameters” of [5]) is the same as the one described about the reading access to the normal parameters or to the input states.

A possible classification (not mandatory) is the following:

Parameter	Referring class	SPABUSTele-gram	Note
Status of test mode	Reading input states (DI)	“R0V8”	
Interlocking status	Reading input states (DI)	“R0V9”	
Sensor 1 ... 7	Reading parameters (AI)	“R0V11/17”	
Current (sensors 1 ... 6)	Reading parameters (AI)	“R0V18”	
Voltage (sensors 1 ... 6)	Reading parameters (AI)	“R0V19”	
Voltage/Current (s.7)	Reading parameters (AI)	“R0V20”	
Frequency	Reading parameters (AI)	“R0V25”	
Event mask 1 ... 64	Reading parameters (AI)	“R0V21/24”	
Data comm. address	Reading parameters (AI)	“R0V200”	
	Writing parameters (DO)	“W0V200”	
Data transfer rate	Reading parameters (AI)	“R0V201”	

Table 10.1.15.-10 (Continued)

Parameter	Referring class	SPABUSTele-gram	Note
Read last event registers	Reading parameters (AI)	“R0L”	Not available
Re-read last event regs	Reading parameters (AI)	“R0B”	Not available
Event on/off	Reading input states (DI)	“R0V52”	
	Writing commands (DO)	“W0V52”	
Programming/run mode	Reading input states (DI)	“R0S198”	
	Writing commands (DO)	“W0S198”	
Data store in EEPROM	Writing commands (DO)	“W0V151”	

11.1.14. Reading contiguous areas

Variables mapped relating to the consecutive Modbus Addresses can be accessed with a single function 1-2 (for digital values) or a single function 3-4 (for analog values) even if the variables are heterogeneous from the REF 542plus point of view.

The following rules hold:

Function 1:

- The maximum number of the addressed coils must be lower than or equal to 2000 (125 Modbus registers).
- Start Modbus Address must correspond to a DDEF of type DI with the range 1 ... 10000.
- All the addressed registers must be mapped by one or more DDEFs of type DI in the range 1 ... 10000. All the DDEFs must be consecutive and have a total number of registers greater than or equal to the number of addressed registers.

Function 2:

- The maximum number of addressed inputs must be lower than or equal to 2000 (125 Modbus registers).
- Start Modbus Address must correspond to a DDEF of type DI with the range 10001 ... 20000.
- All the addressed registers must be mapped by one or more DDEFs of type DI in the range 10001 ... 20000. All the DDEFs must be consecutive and have a total number of registers greater than or equal to the number of addressed registers.

Function 3:

- The maximum number of the analog outputs addressed must be lower than or equal to 125.
- The Start Modbus Address must correspond to a DDEF of type AI with the range 40001 ... 50000.
- All the addressed registers must be mapped by one or more DDEFs of type DI in the range 40001 ... 50000. All the DDEFs must be consecutive and have a total number of registers greater than or equal to the number of addressed registers. The number of registers supplied by each DDEF is given by the formula: DataLength * Members \ 2.

Function 4:

- The maximum number of the analog input's addressed inputs must be lower than or equal to 125.
- The Start Modbus Address must correspond to a DDEF of type AI with the range 30001 ... 40000.
- All the addressed registers must be mapped by one or more DDEFs of type DI in the range 30001 ... 40000. All the DDEFs must be consecutive and have a total number of registers greater than or equal to the number of addressed registers. The number of register supplied by each DDEF is given by the formula: DataLength * Members \ 2.

The use of such a method could be very helpful, for instance, in case of reading polled variables.

It is not possible to read with a single request not homogeneous variables from the Modbus protocol point of view (that is., coils and inputs states together).

11.1.15. Reading events

The communication board keeps in its permanent memory the last 100 events raised by REF 542plus (RCE).

The control system may read one or more REF 542plus events accessing to the file 2 registers according to the rules described in Section 10.1.15. Function 20 (read general reference).

The file 2 structure is the following:

Register	Meaning	Direction	Values
0	Number of events stored	r	0..100
1	Overflow flag	r	0 = no overflow 1 = overflow
2	Clear RCE	w	0xFF00 = clear other = error
3	Event 1 (the oldest)	r	
...			
9			
10	Event 2	r	
...			
16			
…	...	...	
696	Event 100	r	
...			
702			
703	Unused	error	
	...	...	
9999	Unused	error	

The structure of the single event (14 bytes long) is:

```
typedef struct {
unsigned short ChanNum; // channel number
unsigned char Reserved;
unsigned char EventType; // event type
unsigned short EventNum; // event number (0..127)
unsigned long time; // number of milliseconds
// elapsed between the
// REF 542plus power-on
// and the event raising
signed long par1; // optional parameter
} DPM_EVENT
```

Events may be read one at a time or by groups (consecutive events). After reading, the events are still in RCE and cleared only after writing the value 0xFF00 in the register 2.

The clear command has no effect on the RCE of the other channel.

If RCE is full and new events are raised by REF 542plus, the oldest events are lost and the overflow flag is set to 1. The overflow flag is cleared only clearing the entire RCE (writing 0xFF00 in the register 2).

Possible operations on RCE are:

Requiring the value of the incremental counter (reading register 0), that is the number of available events Mod 100. The incremental counter:

Corresponds to the number of stored events, if the overflow flag is 0.

Corresponds to the number of events generated after the overflow, if the overflow flag is 1. In this case, the number of stored events is 100.

Field	Address	Function	Byte Count	Ref Type	File Number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	2	0	1

If two events are stored, the communication board answers with:

Field	Address	Function	Byte Count	Sub resp ByteCount	Ref Type	Data
Dimension	1	1	1	1	1	2
Value	xx	20	4	3	6	2

Reading the overflow flag (reading register 1):

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	2	1	1

In case of RCE overflow, the communication board answers with:

Field	Address	Function	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimension	1	1	1	1	1	2
Value	xx	20	4	3	6	1

Clearing RCE (writing 0xFF00 on register 2):

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data
Dim.	1	1	1	1	2	2	2	2
Value	Xx	21	9	6	2	2	1	0xFF00

The communication board answers with the echo message:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data
Dim.	1	1	1	1	2	2	2	2
Value	Xx	21	9	6	2	2	1	0xFF00

Reading a single event (reading the relating registers):

The request is valid if, and only if, the addressed event is available and the register range equals the dimension of the event structure.

For example, if two events are stored, the following request (read the last event) is valid:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	2	10	7

The communication board answers with the following message:

Field	Address	Func-tion	Byte Count	Sub req byte Count	Ref Type	...
Dimen-sion	1	1	1	1	1	
Value	xx	20	16	15	6	

...	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
	2	2	2	2	2	2	2
	xx						
	Num-Chan	EvType	EvNum	Time		Parameter	

On the contrary, the following message (still valid) does not return a valid event for the address (it is not an event starting register) or for the dimension (different from 7):

Field	Ad-dress	Func-tion	Byte Count	Ref Type	File num-ber	Req Ad-dress	Reg Count
Dimen-sion	1	1	1	1	2	2	2
Value	xx	20	7	6	2	11	6

In this case, the communication board answers with an INVALID DATA error message.

Reading a group of consecutive events (reading a block of registers):

In order to minimize the communication overhead during the RCE transmission, it is helpful to require event blocks, where each block contains the maximum number of events sending with a single Modbus message.

As the Modbus function 20 allows up to 250 bytes to be transmitted, a single event block can contain up to 250/14 = 17 events.

The easier way to request an event block is to use a single sub request starting at the beginning of the first event in the block and to have as many registers as (7 * number of events).

For example, the request for event 18 to 34 is:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	2	3+7*17	7*17

If all the 17 events are stored, the communication board answers with a message like:

Field	Address	Function	Byte Count	Sub req byte Count	Ref Type	Data 1	...	Data 7*17
Dimension	1	1	1	1	1	2		2
Value	xx	20	14*17 + 2	14*17 + 1	6	6		6

11.1.16. Reading events in extended form

The last 100 events raised by REF 542plus (RCE) may also contain the raising absolute time.

The control system may read one or more REF 542plus events with the absolute time tagging (event in extended form) accessing to the file 5 registers according to the rules described in
Section 10.1.15. Function 20 (read general reference).

The file 5 has the same structure described for file 2, while the only difference regards the single event structure:

Register	Meaning	Direction	Values
0	Number of events stored	r	0..100
1	Overflow flag	r	0 = no overflow 1 = overflow
2	Clear RCE	w	0xFF00 = clear other = error
3	Event 1 (the oldest)	r	
...			
13			

Table 10.1.15.-10 (Continued)

Register	Meaning	Direction	Values
14	Event 2	r	
...			
24			
...	...	...	
1091	Event 100	r	
...			
1102			
1103	Unused	error	
	...	...	
9999	Unused	error	

The structure of the single event in extended format (22 bytes long) is:

```
typedef struct
{
unsigned short ChanNum; // channel number
unsigned char Reserved;
unsigned char EventType; // event type
unsigned short EventNum; // event number (0..127)
unsigned long time; // number of milliseconds
// elapsed between the
// REF542 power-on
// and the event raising
signed long par1; // optional parameter
CP56Time2a AbsoluteTime // time of the event raising
char Reserved;
} DPM_EVENT_EXT
```

with:

```
typedef struct
{
char :1;
char year :7;
char :4;
char month :4;
char day_of_week :3;
char day_of_month :5;
char summer_time :1;
char :2;
char hour :5;
char invalid :1;
char :1;
char min :6;
char msec_high :8;
char msec_low :8;
} CP56Time2a ;
```

All the operation described for file 2 (event file) is also valid for file 5 (event file in extended form).

Operations on file 5 apply on both file 2 and 5 and viceversa. It means, for example, that the “Clearing RCE command” (writing 0xFF00 on file 5 register 2) clears both file 2 and file 5.

Refer to Section 11.1.15. Reading events the previous paragraph for a full description of operation on file 5. Just note that each event in extended form is 22 bytes long while an event in standard form is 14 bytes long.

11.1.17. Reading and clearing events

The control system may read one or more REF 542plus events accessing to the file 2 registers according to the rules described in Section 10.1.15. Function 20 (read general reference).

It is possible to read n older events.

The rules are the same as in “Reading Events with file 2”, with the only difference that the Reg Address filed must have the value 3 (corresponding to the last event).

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	11	3	7

The board answers as in the case of reading events and after that deletes the read events.

Field	Address	Function	Byte Count	Sub req byte Count	Ref Type	...
Dimension	1	1	1	1	1	
Value	xx	20	16	15	6	

...	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
	2	2	2	2	2	2	2
	xx						
	NumChan	EvType	EvNum	Time		Parameter	

In case the RegAddress is not 3, the board answers with ILLEGAL_DATA_ADDRESS.

The Read & Clear:

- Deletes the last n read events.
- Puts the overflow flag to false.
- Puts EVENT COUNTER = EVENT COUNTER − n (if the overflow flag was TRUE, the EVENT COUNTER becomes 100 - n).

11.1.18. Reading and clearing events in extended form

The control system may read one or more REF 542plus events accessing to the file 5 registers according to the rules described in Section 10.1.15. Function 20 (read general reference).

It is possible to read n older events in extended form .

The rules are the same of “Reading events in extended form with file 5”, with the only difference that the Reg Address field must have the value 3 (corresponding to the last event).

The Read & Clear:

- Deletes the last n read events
- Puts the overflow flag to false
- Puts EVENT COUNTER = EVENT COUNTER − ***n*** (if the overflow flag was TRUE, the EVENT COUNTER becomes 100 - n.

11.1.19. Reading disturbance records

REF 542plus makes available the disturbance records according to the procedure described in [3]; the only difference is that the record data format is customized instead of EVE format. With the provided software “SMSCONVERT” the record data can be converted in COMTRADE format, see Section 11.1.19.1. Conversion of disturbance data.

The SPA bus variables used in this procedure are:

V20

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing func-tion	Values
Read	2 / 1	“R0V20”	AI	4	Number of available records: 0..20
Write	2 / 1	“W0V20”	AO	6	0 = session end 1 = select ‘uncompressed’ mode 2 = select ‘compressed’ mode other = error

M28

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing func-tion	Values
Read	1 / 34	“R0M28”	AI	4	Directory information, see [3]
Write	2 / 1	“W0M28”	AO	6	Record index : 1.. V20 current value

M29

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing func-tion	Values
Read	2 / 1	“R0M29”	AI	4	Number of lines (128 bytes each) contained in the record addressed by M28; range: 0..1023 0 = M28 not valid
Write	n/a	N/a	n/a	n/a	Error

M30

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing function	Values
Read	2 / 1	“R0M30”	AI	4	Last line read (0..1023)
Write	2 / 1	“W0M30”	AO	6	Line to read (1..1023) Set to 1 on every access to M28 Incremented by REF 542plus to each valid access to M31

M31

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing function	Values
Read	1 / 128-64	“R0M31”	AI	4	Single record line (128 byte if in ASCII format or 64 bytes if in binary format)
Write	n/a	n/a	n/a	n/a	Error

As pointed out in Section 9.1. Modbus RTU addressing, the Modbus Address 39900 is reserved for reading access to the variable M31 to speed up disturbance record transferring. This way all the pairs of ASCII characters included in each line of a record are compressed in a single binary byte, that is the pair “5D” becomes the byte 0x5D.

To use this feature, the control system must configure a DDEF with:

- Start Address = 39900
- DataType = AI
- DataLength = 1
- SPA bus telegram = “R0M31”
- Members = 64

To avoid this optimization, the control system must configure a DDEF with :

- Start Address = any address in the range 30001..40000, but 39900
- DataType = AI
- DataLength = 1

- SPA bus telegram = “R0M31”
- Members = 128

V16

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing function	Values
Read	2 / 1	“R0V16”	AI	4	REF disturbance record memory status: ‘0’ = the memory is not full and is available ‘1’ = the memory is full and available ‘2’ = the memory is not available
Write	2 / 1	“W0V16”	AO	6	Clear the oldest disturbance record after a double writing access: first access : ‘0’ second access : ‘1’

Be careful in clearing the oldest record, because the other Modbus connection is not notified about the event.

V17

Direc-tion	Data Length / Member	SPABUS Tele-gram	Modbus access mode	Modbus acces-sing function	Values
Read	2 / 1	“R0V16”	AI	4	Current compression percentage (Delta)
Write	2 / 1	“W0V17”	AO	6	Compression percentage (Delta) for the compression algorithmRange: 0..5

To make the complete procedure, the control system should perform the following steps:

- Open the session by writing the transferring mode (‘uncompressed’ (1) or ‘compressed’ (0)) on the V20 variable.
- Verify that some disturbance records exist by reading V20.

- If V20 > 0, select a record by writing the index on M28.
- Require the record directory by reading M28.
- Require the number of lines of the selected record by reading M29.
- Loop reading on M31 to obtain single lines. The operation can be manually controlled by reading/writing on M30, or leaving an automatic control to REF 542plus that increments M30 to every access to M31.
- Clear the last read record by means of two consecutive writing on V16, the first with value 0 and the second with value 1. Wait for the end of the memory re-ordering by reading V16: during re-ordering, the V16 value is 2.
- Close the session by writing 0 on V20.
- At any moment, V16 can be read to verify whether the REF 542plus recording memory is full (1) or not (0).
- At any moment, before requiring a record, V17 can be set with a new value for the compression percentage (Delta).

11.1.19.1. Conversion of disturbance data

The uploaded file with record data must be converted with smsconvert.exe.

Important: note that smsconvert.exe is only compatible with the binary files uploaded in compress mode (the M31 register must be 39900, see Section 11.1.19. Reading disturbance records).

The input file must have this standard name “rexxx.xxx” where “x” must be a number (i.d.re001.002).

It will be converted in two files in the standard COMTRADE format.

- The first (.cfg) with the configuration of the fault recorder. It follows the description of the configuration for 32 digital inputs, frequency, control of analogue section, control of digital section, digital data and analogue data.
- The second (.dat) with the stored data.

The procedure to get the file is:

- From DOS, it must be used a command line with two parameters (input and output file):

  Smsconvert.exe inputFile outputFile

  Where inputFile is the uploaded file, while outputFile is the path name of the two output files.

11.1.20. Time synchronization

The absolute time may be sent from the control system to the communication board by using the extended file 7:

File 7 (Time and Date):

Register	Meaning	Direction
0 ... 3	Time and date	r/w
4 ... 9999	Unused	error

The “Time and date” field is mapped on the following structure:

```
typedef struct
{
CP56Time2a AbsoluteTime
char Reserved;
} TimeType
```

Where CP56Time2a is the structure described in Section 11.1.16. Reading events in extended form.

Each time the communication board receives a writing message on file 7, it forwards the entire AbsoluteTime field to the main board that applies the time.

That means the entire file must be filled with a single function 21 message, otherwise a corrupted time value may be transferred to the main board.

The communication board does not refuse partial writing.

The communication board does not control the time format.

In order to permit network synchronization, it is possible to write the file 7 in broadcast mode, even if this behavior is not allowed by [1].

Each register of the file 7 may be read by using function 20 messages, even with partial accesses. The communication boards returns the last value set and not the real current absolute time.

For example, in order to set the main board absolute time, the control system may use the following message:

Field	**Slave Address**	**Function**	**Byte Count**	**Ref Type**	**File number**	**Req Address**	**Reg Count**	**…**
Dimension	1	1	1	1	2	2	2	
Value	Xx	21	15	6	7	0	4	

…	**Data1..8**
	7 + 1 (not significant)
	Value in Cp56time2a format

The communication board, in case of success, answers with the complete echo of the received message.

The same message may be sent in broadcast mode:

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	…
Dimension	1	1	1	1	2	2	2	
Value	0	21	15	6	7	0	4	
…				**Data1..8**				
				7 + 1 (not significant)				
				Value in Cp56time2a format				

In this case the communication board does not answer.

The REF 542plus generates an event E43 to indicate that is synchronized with the Modbus date-time.

11.1.21. Time setting and reading

11.1.21.1. Time reading

The SPA message to read the full REF 542plus time-date in Cp56time2a format is the following:

>99R0M10:XXcr

To read the time the following Ddef has to be mapped and downloaded:

Date-Hour-R	Info	>300-00 & <400-00	AI	1	0	R0-M10	0	22

The slave reply is:

99<D:yy-mo-dd hh.mm;ss.sss:XXcrlf

The date-time data is on 11 registers (22 bytes with the last = 0) corresponding to the values of the single chars.

11.1.21.2. Time setting

The time setting is an update of the date-time of REF 542plus in Cp56time2a format.

The Cp56time2a time can be forwarded to REF 542plus and generates an event E46 each time the updating of date-hour from the REF 542plus is performed.

The SPA message to write the full REF 542plus time-date has the following format (same as SPA date time):

>99W0M10:Dyy-mo-dd hh.mm;ss.sss:XXcr

Note: The initial character D must be added to interpret the message as string and not as number.

To set the time, the following DDEF has to be downloaded:

Date-Hour-W	Info	>400-00 & <500-00	AO	1	0	W0-M10	0	22

Now, with the Modbus Function 16 it is possible to set the values of the 11 registers mapped in the DDEF by assigning the char D to the low byte of the first register.

Field	**Sl. Addr.**	**Func-tion**	**Reg Start Num-ber**	**N. Regs**	**N. Bytes**	**Reg1 (Hex)**	**Reg2 (Hex)**	**Reg 11**
Di-men-sion (in bytes)	1	1	2	2	1	2	2	2
Value	Xx	16	0..99-99	11	22	<443-0>	<332-D>	any

Where the reg1 corresponds to “Dy” and reg2 to “y-“

11.1.22. Re-programming the communication board

The application program may be dinamically updated by using file 8 and 9:

Table 11.1.22.-1 File 8 (Program Command Table):

Register	Meaning	Direction
0 ... 1	Program checksum	r
2 ... 3	Program start address	w
4 ... 5	Program length	w
6 ... 9999	Unused	error

Program checksum

Reports the checksum and the validity flag of the downloaded program according to the following structure:

Byte 0	Byte 1	Byte 2	Byte 3
Validity flag		Checksum	
0	0/1	High byte	Low byte

The Validity flag sub-field is 1 set to 1 if:

- The programming session is active.
- The Program start address register has a valid value.
- The Program Length register has a valid value.

The Checksum sub-field is the 16 bits one's complement of the first Program Length / 2 words inside the Program Image. If the Validity flag is 0, the Checksum is not significant.

The Program Checksum registers are computed each time they are accessed in reading.

The Program Checksum registers must be read with a single command; the use of a function 20 message reading only one register is refused by the communication board.

If Program Length register is null, Program Checksum returns 0x0001FFFF if the Program start address register is valid otherwise, 0x0000FFFF.

The Program Checksum registers may be read outside a programming session.

The Program Checksum registers cannot be written by the control system.

Program start address

Using this 32 bit field, the control system communicates the physical start address (in the flash memory space) where the updated program must be written at.

Valid addresses are:

- 0x1010000for BootLoader updates
- 0x1040000for Application Program updates

Different addresses are refused by the communication board

The Program start address must be written with a single command; the use of a function 21 message writing only one register is refused by the communication board.

The Program start address must be written inside a programming session.

The Program start address may be written in any phase of the programming session, even inside a program download.

The Program start address may be written in broadcast mode.

When the programming session is closed (stop or abort) the Program start address is reset to 0.

Program length

Using this 32 bit field, the control system communicates the length (in bytes) of the updated program.

The maximum length is 0x1FC00 (127Kb).

Higher values are refused by the communication board.

Writing the value 0 means a request to clear the program at location Program start address without substituting it.

The Program length must be written with a single command; the use of a function 21 message writing only one register is refused by the communication board.

The Program length may be written inside a programming session only.

The Program length may be written in any phase of the programming session, even inside a program download.

The Program length may be written in broadcast mode.

When the programming session is closed (stop or abort) the Program length is reset to 0xFFFFFFFF.

File 9 (Program Image):

Register	Meaning	Direction
0 ... 2031	Program image	W
2031 ... 9999	Unused	error

The file 9 is a virtual image for incoming updated programs.

Each register writing access allows a writing on an internal area (127Kbytes long) storing the incoming program.

When the i^{th} register is writing accessed, the data part of the message is written in the internal area starting at offset ***i*64***.

The last register in file 9 is:

127K / 64 − 1 = 130048 / 64 − 1 = 2032 − 1 = ***2031***

The Program Image registers may be written inside a programming session only.

The Program Image register may be written in broadcast mode.

The Application Program is updated with a valid Program Image at the end of the programming session with a Stop Programming command.

Re-programming procedure

1. Start a programming session, see Section 11.1.27. Programming session.
2. Send a function 21 message (on registers 2 and 3 of file 8) with a valid program start address (0x1040000).

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data 1..4
Dimension	1	1	1	1	2	2	2	4
Value	Xx or 0	21	27	6	8	2	2	01h 04h 00h 00h

The communication board (if not in Broadcast mode) answers with the echo of the message

3. Send a function 21 message (on registers 4 and 5 of file 8) with the length of the program to download (0x10644).

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data 1..4
Dimension	1	1	1	1	2	2	2	4
Value	Xx or 0	21	27	6	8	4	2	00h 01h 06h 44h

The communication board (if not in Broadcast mode) answers with the echo of the message

4. Send the program image:
   - Send a first function 21 message (starting at register 0 of file 9); the data part may contain any number of bytes (within the limit of Modbus messages). The simplest way is to use 64 bytes per message (32 registers); the most efficient way is to use 192 bytes per message (= 64 * 3, or 96 registers). 256 bytes cannot be used being outside the message limit.

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data 1..192
Dimension	1	1	1	1	2	2	2	192
Value	Xx or 0	21	199	6	9	*0*	96	...

The communication board (if not in Broadcast mode) answers with the echo of the message

   - Send the second frame, starting at register 1 if the previous message contained 64 bytes, or at register 3 if the previous message contained 192 bytes, or at any other register.

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data 1..192
Dimension	1	1	1	1	2	2	2	192
Value	Xx or 0	21	199	6	9	*3*	96	...

The communication board (if not in Broadcast mode) answers with the echo of the message.

- Loop on file 8 registers, with a boundary of 1 (data part of 64 bytes) or 3 (data part of 192 bytes) until the last full message.
- Send the last function 21 message with the remaining bytes (if necessary).

5. Verify the correct download by reading the program checksum with a function 20 message (on registers 0 and 1 of file 8):

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	Xx	20	7	6	8	*0*	2

The communication board answers with a similar message:

Field	Slave Address	Function	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimension	1	1	1	1	2	4
Value	Xx	20	6	5	6	00h 01h A3h 43h

- The download is correct if the valid subfield (register 0) is 1 and the Checksum subfield equals the program checksum (computed by the control system in any off-line way).

6. If the download is valid, close the programming session with a Stop programming command, otherwise close the programming session with an Abort programming command or retry the procedure from step 2.

Erasing program procedure

1. Start a programming session, see Section 11.1.27. Programming session.
2. Send a function 21 message (on registers 2 and 3 of file 8) with a valid program start address (0x1040000).

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data 1..4
Dimension	1	1	1	1	2	2	2	4
Value	Xx or 0	21	27	6	8	2	2	01h 04h 00h 00h

The communication board (if not in Broadcast mode) answers with the echo of the message.

3. Send a function 21 message (on registers 4 and 5 of file 8) with a 0 length for the program.

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data 1..4
Dimension	1	1	1	1	2	2	2	4
Value	Xx or 0	21	27	6	8	4	2	00h 00h 00h 0h

The communication board (if not in Broadcast mode) answers with the echo of the message.

4. Do not send any program image but just verify the correct program checksum with a function 20 message (on registers 0 and 1 of file 8):

Field	Slave Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	Xx	20	7	6	8	0	2

The communication board answers with the message:

Field	Slave Address	Func-tion	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimen-sion	1	1	1	1	2	4
Value	Xx	20	6	5	6	00h 01h FFh FFh

Close the programming session with a Stop programming command.

11.1.22.1. Re-Programming the Boot Loader

The Boot Loader may be re-programmed by using the same rules described in the Section 11.1.22. Re-programming the communication board.

The only difference regards the Program start address field of the file 8 that must be filled with the value 0x1010000.

11.1.23. Sessions

As two different control systems may be connected to REF 542plus via communication board at the same time, it is necessary to protect those operations requiring a mutually exclusive access.

Critical operations belong to 3 classes:

- Commanding digital outputs.
- Pre-setting parameters.
- Transferring or modifying the Communication board configuration.

The Communication board defines then 3 respective sessions:

- Writing command session.
- Writing parameter session.
- Configuration session.

The mutually exclusive access is guaranteed among operations within the same session and not among operations belonging to different sessions.

To operate within a given session, each control system must perform the following steps:

- Require the starting of the session by writing the value 1 in the file 3 relevant register. The communication board acknowledges by echoing the receiving message; if another control system already holds the session, the communication board answers with an ILLEGAL_DATA_VALUE error message.
- Operate inside the class; in this phase the control system may also perform operations outside the class.
- Closing the session by writing the value 0 in the same register used to require the starting.

In any case, the communication board automatically closes a session after a given time after the last operation over an element of the class.

11.1.24. Writing command session

All the variables mapped on DDEFs with type DO, address in the range 1..10000 and bit WC set are to be considered commands.

Pre-setting a command is allowed only within a writing command session.

To start the session, the control system must write the value 1 in the register SessionCommand (register 2) of file 3 with the following message:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data
Dimension	1	1	1	1	2	2	2	2
Value	Xx	21	9	6	3	2	1	1

If the request is accepted, the communication board answers with the echo of the message Otherwise (the session is already held by the other connection), the communication board answers with an ILLEGAL_DATA_VALUE error message.

Reading a command status is allowed outside the session as well.

To close the session, it is necessary to write the value 0 in the register SessionCommand (register 2) of file 3.

At any moment, the writing command session status may be required by reading the register StatusCommand (register 5) of file 3:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	3	5	1

The communication board returns:

- '0' if no writing command session is currently active
- '1' if a writing command session is currently active by the control system connected on port 1
- '2' if a writing command session is currently active by the control system connected on port 2

Field	Address	Function	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimension	1	1	1	1	1	2
Value	xx	20	4	3	6	0 ... 2

11.1.25. Writing parameter session

All the variables mapped on DDEFs with type AO, address in the range 40001 ... 50000 and bit WP set, are to be considered protected parameters.

Pre-setting a protected parameter is allowed only within a writing parameter session.

To start the session, the control system must write the value 1 in the register SessionParameter (register 1) of file 3 with the following message:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data
Dimension	1	1	1	1	2	2	2	2
Value	Xx	21	9	6	3	1	1	1

If the request is accepted, the communication board answers with the echo of the message. Otherwise (the session is already hoed by the other connection), the communication board answers with an ILLEGAL_DATA_VALUE error message.

Reading a parameter value is allowed outside the session as well.

To close the session, it is necessary to write the value 0 in the register SessionParameter (register 1) of file 3.

At any moment, the writing parameter session status may be required by reading the register StatusParameter (register 4) of file 3:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	Xx	20	7	6	3	4	1

The communication board returns:

- '0' if no writing parameter session is currently active
- '1' if a writing parameter session is currently active by the control system connected on port 1
- '2' if a writing parameter session is currently active by the control system connected on port 2

Field	Address	Function	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimension	1	1	1	1	1	2
Value	Xx	20	4	3	6	0 ... 2

Not protected parameters may be preset outside a writing parameter session. Anyway, in order to preset parameters (protected or not), it is necessary to set the REF 542plus mode:

- Set the Program mode before pre-setting parameters (either by using a writing parameter session or not).
- Set Run mode after the pre-setting phase.
- Send a Store parameter command for permanently storing parameters.
- Program mode (SPA telegram W0S198 with value 0), Run mode (SPA telegram W0S198 with value 1) and store in EEPROM (SPA telegram W0V151 with value 1) must be defined as commands or parameters themselves (protected or not) inside the DDEF list.

11.1.26. Configuration session

Any writing access to the file 1 and/or file 4 registers, containing the DDEF table and the Serial Channel Descriptor table is to be considered configuration.

The configuration register writing is allowed only within a configuration session.

To start the session, the control system must write the value 1 in the register SessionConfig (register 0) of file 3 with the following message:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data
Dimension	1	1	1	1	2	2	2	2
Value	Xx	21	9	6	3	0	1	1

If the request is accepted, the communication board answers with the echo of the message. Otherwise (the session is already held by the other connection), the communication board answers with an ILLEGAL_DATA_VALUE error message.

Reading a configuration register is allowed outside the session, too.

To close the session it is necessary to write in the register SessionConfig (register 0) of file 3:

The value 0, all new values are applied

The value 2, all new values are discarded

At any moment, the configuration session status may be required by reading the register StatusConfig (register 3) of file 3:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	3	3	1

The Communication board returns:

- ‘0’ if no configuration session is currently active
- ‘1’ if a configuration session is currently active by the control system connected on port 1
- ‘2’ if a configuration session is currently active by the control system connected on port 2

Field	Address	Function	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimension	1	1	1	1	1	2
Value	Xx	20	4	3	6	0 ... 2

11.1.27. Programming session

Any writing access to the file 8 and/or file 9 registers, containing the Program Parameters and the Program Image is to be considered programming.

The programming register writing is allowed only within a programming session.

To start the session, the control system must write the value 1 in the register SessionProgramming (register 6) of file 3 with the following message:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count	Data
Dimension	1	1	1	1	2	2	2	2
Value	Xx or 0	21	9	6	3	6	1	1

If the request is accepted, the communication board answers with the echo of the message. Otherwise (the session is already held by the other connection), the communication board answers with an ILLEGAL_DATA_VALUE error message.

After a valid start programming session, the communication board fills with all 0's the entire file 8, in order to reset any previous programming.

Reading a programming register (file 8 only) is allowed outside the session, as well.

To close the session, it is necessary to write in the register SessionProgramming (register 7) of file 3:

The value 0, all new values are applied and the Communication board restarts if a valid program image has been transferred

The value 2, all new values are discarded

At any moment, the programming session status may be required by reading the register StatusProgramming (register 7) of File 3:

Field	Address	Function	Byte Count	Ref Type	File number	Req Address	Reg Count
Dimension	1	1	1	1	2	2	2
Value	xx	20	7	6	3	7	1

The communication board returns:

- '0' if no programming session is currently active.
- '1' if a programming session is currently active by the control system connected on port 1.
- '2' if a programming session is currently active by the control system connected on port 2.

Field	Address	Function	Byte Count	Sub resp Byte Count	Ref Type	Data
Dimension	1	1	1	1	1	2
Value	Xx	20	4	7	6	0 ... 2

The start/stop/abort programming commands may be also sent in broadcast mode, in order to re-program all the slaves in the network with a single procedure. This behavior is not compliant with [1].

11.1.28. Other operations

11.1.28.1. Reading the communication board working status

By means of function 7, the control system may control the operating status of the communication board:

Field	Address	Function
Dimension (in bytes)	1	1
Value	Xx	7

The communication board answers with:

Field	Address	Function	CoilValue
Dimension (in bytes)	1	1	1
Value	Xx	7	XX

Where XX is described in Section 10.1.7. Function 7 (read exception status).

11.1.29. Communication channel test

By means of function 8 (loopback test), the control system may control the operating status of the communication channel to which it is connected to:

Field	Address	Function	DiagCode	DiagData
Dimension (in bytes)	1	1	2	2
Value	xx	8	0 ... 20	XX

where:

DiagCode	Meaning	DiagData
0	Return Query Data (echo)	Value to return
1	Restart Comm Option	n/a
2	Return Diagnostic Register	n/a
4	Force Slave to Listen Only Mode	n/a
10	Clear Counter and Diagnostic Register	n/a
11	Return Bus Message Count	n/a
12	Return Bus CRC Error Count	n/a
13	Return Bus Exception Error Count	n/a
14	Return Slave Message Count	n/a
15	Return Slave No Message Count	n/a
16	Return Slave NAK Count	n/a
17	Return Slave Busy Count	n/a
18	Return Bus Character Overrun Count	n/a
19	Return Overrun Error Count	n/a
20	Clear Overrun Error Count and Flag	n/a

The communication board answers with:

Field	Address	Function	DiagCode	DiagData
Dimension (in bytes)	1	1	2	2
value	xx	8	0 ... 20	XX

Where:

DiagCode	Meaning	Effect	DiagData
0	Return Query Data (echo)	Echo the request	Input data
1	Restart Comm Option	(1)	n/a
2	Return Diagnostic Register	Returns the bitwise status of the diagnostic register	Always 0
4	Force Slave to Listen Only Mode	The communication board does not answer any more on this channel	n/a
10	Clear Counter and Diagnostic Register	Counter and diagnostic register are cleared	n/a
11	Return Bus Message Count	Returns the number of all the messages that the communication board has processed since the last restart (or power-on) was issued	Bus Message Count
12	Return Bus CRC Error Count	Returns the number of CRC errors that the communication board has encountered since the last restart (or power-on) was issued	CRC Error Count

Table 11.1.22.-1 (Continued)

DiagCode	Meaning	Effect	DiagData
13	Return Bus Exception Error Count	Returns the number of exception codes that the communication board has returned since the last restart (or power-on) was issued	Exception Error Count
14	Return Slave Message Count	Returns the number of messages addressed to the communication board since the last restart (or power-on) was issued	Slave Message Count
15	Return Slave No Message Count	Returns the number of times the communication board has failed to respond since the last restart (or power-on) was issued	Slave No Message Count
16	Return Slave NAK Count	Returns the number of times the communication board responded with a NACK (Negative Acknowledgement) since the last restart (or power-on) was issued	Slave Nack Count
17	Return Slave Busy Count	Returns the number of times the communication board responded as busy since the last restart (or power-on) was issued	Slave Busy Count

Table 11.1.22.-1 (Continued)

DiagCode	Meaning	Effect	DiagData
18	Return Bus Character Overrun Count	Returns the number of times a message was not handled due to an overrun or to a hardware malfunction since the last restart (or power-on) was issued	Bus Overrun Count
19	Return Overrun Error Count	As above	Bus Overrun Count
20	Clear Overrun Error Count and Flag	Clear the Bus Overrun Count	n/a

(1) The communication board port is initialized and restarted and all of its communications event counters are cleared. If the port is currently in Listen Only Mode, no response is returned. This function is the only one that brings the port out of Listen Only Mode. If the port is not currently in Listen Only Mode, a normal response is returned. This occurs before the restart is executed.

When the communication board receives the query, it attempts a restart and executes its power-up confidence tests. Successful completion of the tests will bring the port online.

11.1.30. Reading REF 542plus information

The control system may receive information about REF 542plus using the function 17 (report slave ID),

Field	Address	Function
Dimension (in bytes)	1	1
Value	xx	17

In this version, the type and the run-light status are handled only:

Field	Address	Function	Byte Count	Slave Id	Run-light	Device Dependent
Dimension	1	1	1	1	1	20
value	xx	17	2	0x32	0x00 = off 0xFF = on	Unused

12. Uploading events with function 3 and 6

12.1. General description

The implementation of the reading of the events with the functions 3 and 6 can be performed with configurable addresses of the Modbus registers.

The registers can be mapped with specific SPA telegrams at addresses between 40000 and 49999. Each register must have one of the 6 specific events SPA telegrams called R0P0, R0P1, W0P2, R0P3, R0P4, W0P5.

1. R0P0 to read if there is an overflow on the number of events. The overflow is present if the events registered in the communication board and raised from the REF 542plus are more or equal to 100.
2. R0P1 to read the number of events. This number can be from 0 to 100 and stands for:
   - The number of available elements in the case the OVERFLOW flag is FALSE.
   - The number of lost events in the case the OVERFLOW flag is TRUE.
3. R0P3 to read and clear the oldest event in extended format (with time stamp). The length of events is 11 words by using the CP56Time2a format.
4. W0P2 to reset events. In this case, all the events and OVERFLOW flag are cleared.
5. R0P4 to read without clearing the oldest event.
6. W0P5 to clear the oldest event.

12.2. Uploading procedures

The events upload can be performed in two different methods:

- The procedure to read the events from a generic SCADA with the read and clear of the oldest event in a unique telegram (R0P3) is faster but you can lose events.

  The procedure is:

  - Read if the OVERFLOW flag with R0P0 is present.
  - Read if the number of events is more than zero by reading the register where R0P1 is mapped.
  - Read the oldest event and delete it by reading the 11 registers where R0P3 is mapped.
  - It is also possible to clear all the events by writing 1 to the register where W0P2 is mapped.
- The procedure to read the events from a generic SCADA with the read and clear of the oldest event in two different telegrams (R0P4 and W0P5) is slower, but you are sure not to lose events.

- Read if the OVERFLOW flag with R0P0 is present.
- Read if the number of events is more than zero by reading the register where R0P1 is mapped.
- Read the oldest event by reading the 11 registers where R0P4 is mapped.
- Clear the oldest event writing 1 to the register where W0P5 is mapped.

12.3. DDEF

A possible DDEF is with all the columns fixed except for the address that can be configurable in the range between 40000 and 49999:

DEFINITION NAME	CATEGORY	ADDRESS	TYPE	LEN	POLL	SPA TEL.	FACT	MEMBERS
Oldest_event_ReadCl	Parameters	40017	AI	1	0	R0P3	0	22
Number_Of_Events	Parameters	40028	AI	2	0	R0P1	0	1
Events_overflow	Parameters	40029	AI	2	0	R0P0	0	1
Oldest_event_Read	Parameters	40030	AI	1	0	R0P4	0	22
Clear_events	Parameters	40405	AO	2	0	W0P2	0	1
Clear_oldest_event	Parameters	40406	AO	2	0	W0P5	0	1

13. Configuring the Modbus board

To configure the Modbus board, the Modbus suite (minimum version is 1.8) must be used.

The Modbus suite is able to download/upload channel descriptors, see file 4 in Section 10.1.15. Function 20 (read general reference) and DDEFs for each REF 542plus on the bus, see Section 9.11. Class.

In order to set the right connection between the PC and the Modbus board do the following settings:

- Select automatic (smart) or not automatic handshake RTS485 converter type according to your device.
- Select the communication settings on the PC port (the default settings of the Modbus board are 19200, n, 8, 1).

The procedure to configure the Modbus board with the DDEFs and channel descriptors is:

1. Scan the network:
   - Single node: peer to peer to connect to a single REF.
   - Multiple nodes: on the bus to connect to all the REFs.
2. Try to connect in one of the following ways:
   - Manually with the right communication settings.
   - Automatically (all the settings are automatically updated to get the right one).
3. Return to the main, select one slave and go on line.

The Modbus suite can configure both the channels descriptors for each port and the DDEFs table.

- To change the channel descriptors of each port:
  - Read the descriptors from the Modbus board.
  - Start a session.
  - Choose the new channel descriptor settings.
  - Send the descriptors to REF.
  - Store or abort the session. If the storing option is chosen, the communication will be lost if descriptors of the channel you are connected to are modified.

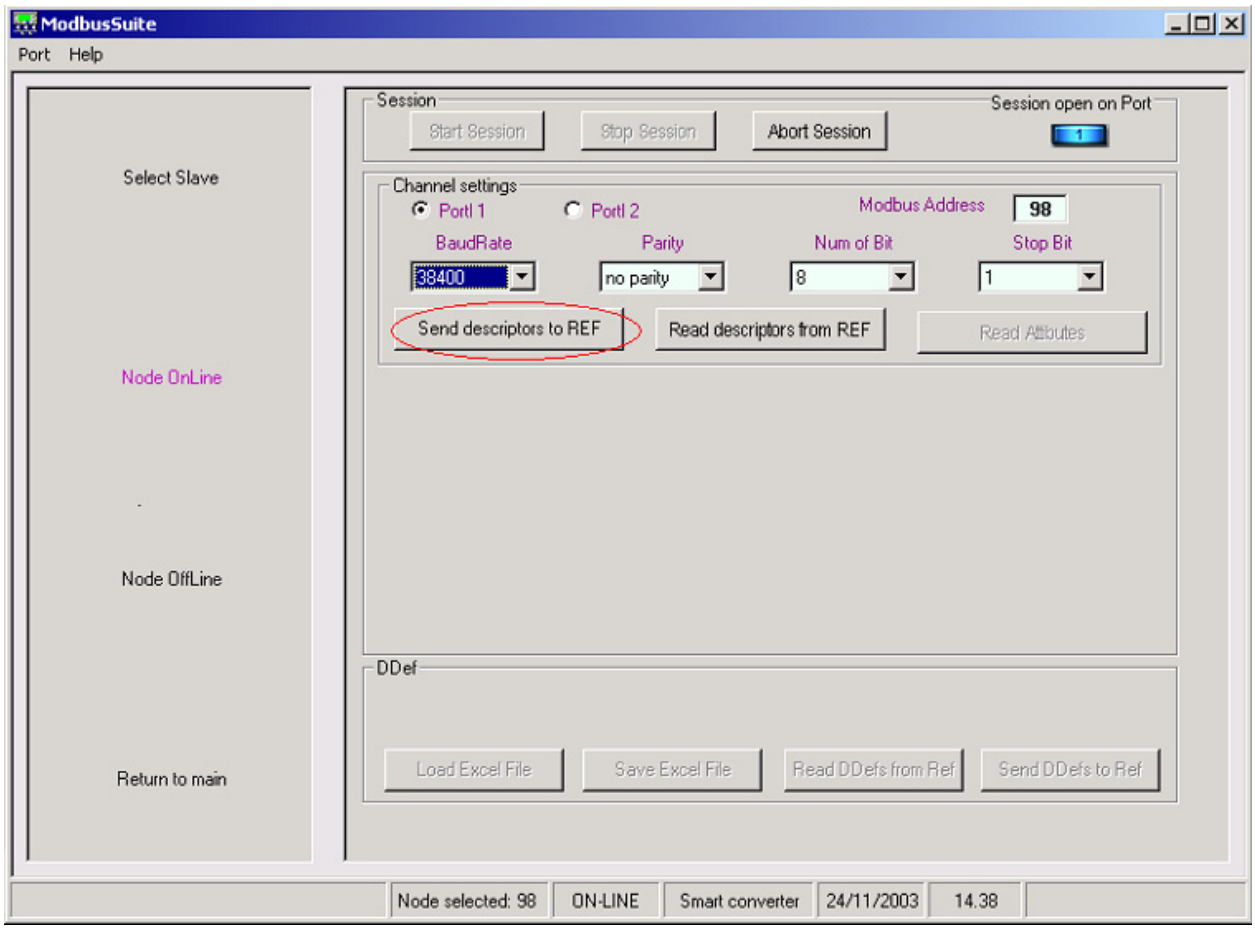
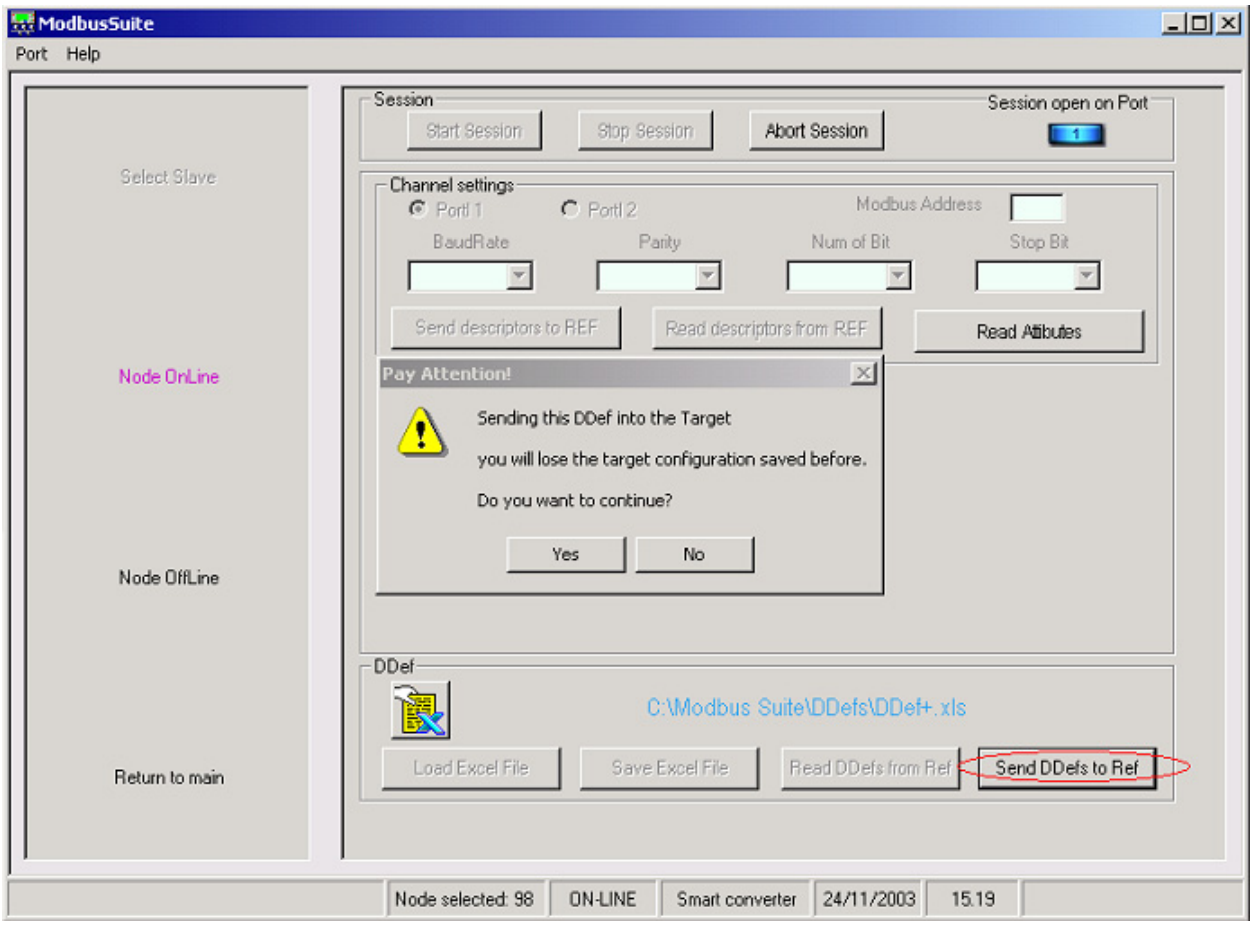


Fig. 13.-1 Send channel descriptors to the Modbus

- To upload / download the DDEF the procedure is:

  Downloading:

  - Load an Excel file with the DDEF to download (see the DedfExample. xls and the Modbu config table commented.xls file given with the Modbus suite as examples).
  - Start a session.
  - Send the DDEF to REF.
  - If the download is OK stop the session to store in flash memory the new configuration.



A051142

Fig. 13.-2 DDEF download

Uploading:

- Read and save the DDEFs from the Modbus board.
- After saving the DDEF file, it is possible to modify some parameters by opening the file in the yellow icon.
- To download the modified file, close it and start a download session.

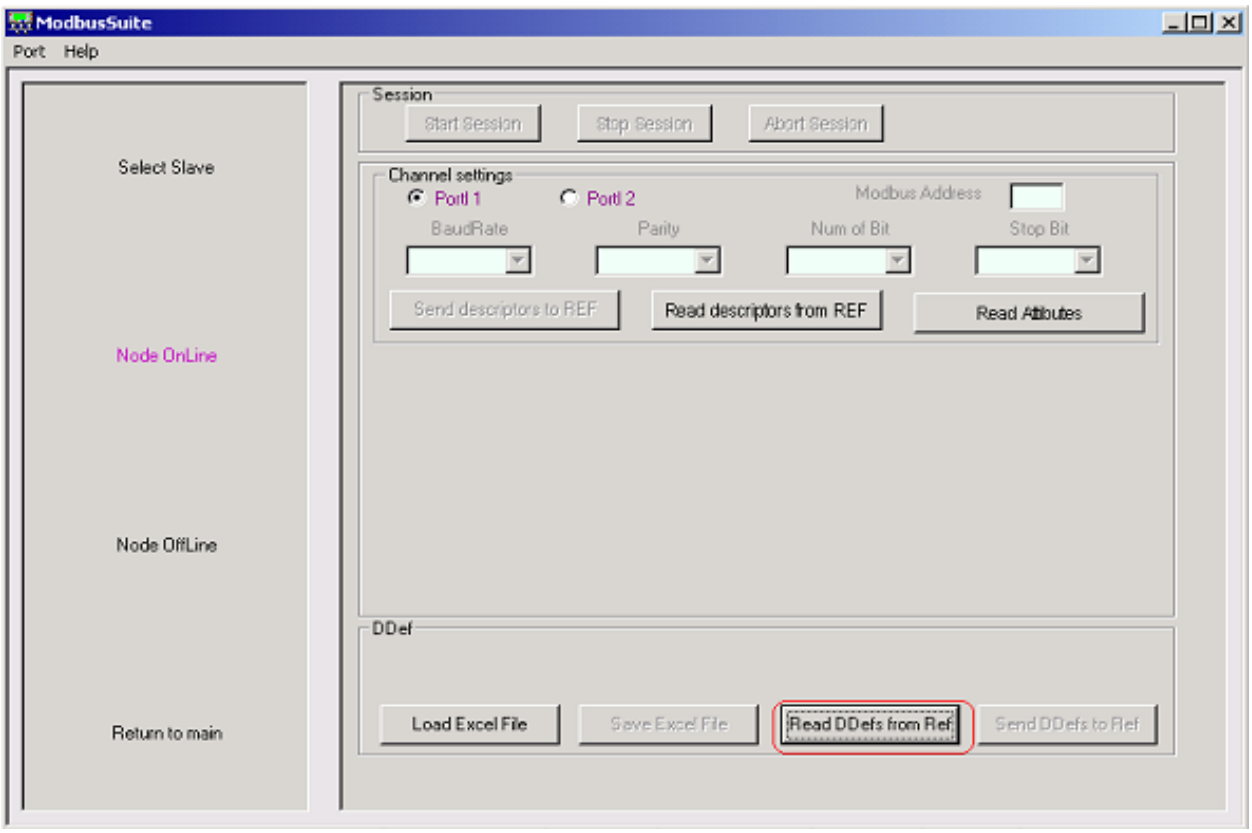


Fig. 13.-3 DDEF upload

14. Commissioning

The commissioning section has the following functionality:

- Measurement reading
- States reading
- Commands
- Parameter reading /writing
- FUPLA reading /writing
- Events reading
- Disturbance record uploading
- Read REF 542plus general info and perform time reading/setting.

To enable the commissioning section the procedure is:

- Scan the network and get the REF 542plus connected.
- Select the commissioning section.
- Select the REF 542plus slave.
- Upload the configuration from REF 542plus. The Modbus suite will divide the DDEF following the category in the related section.

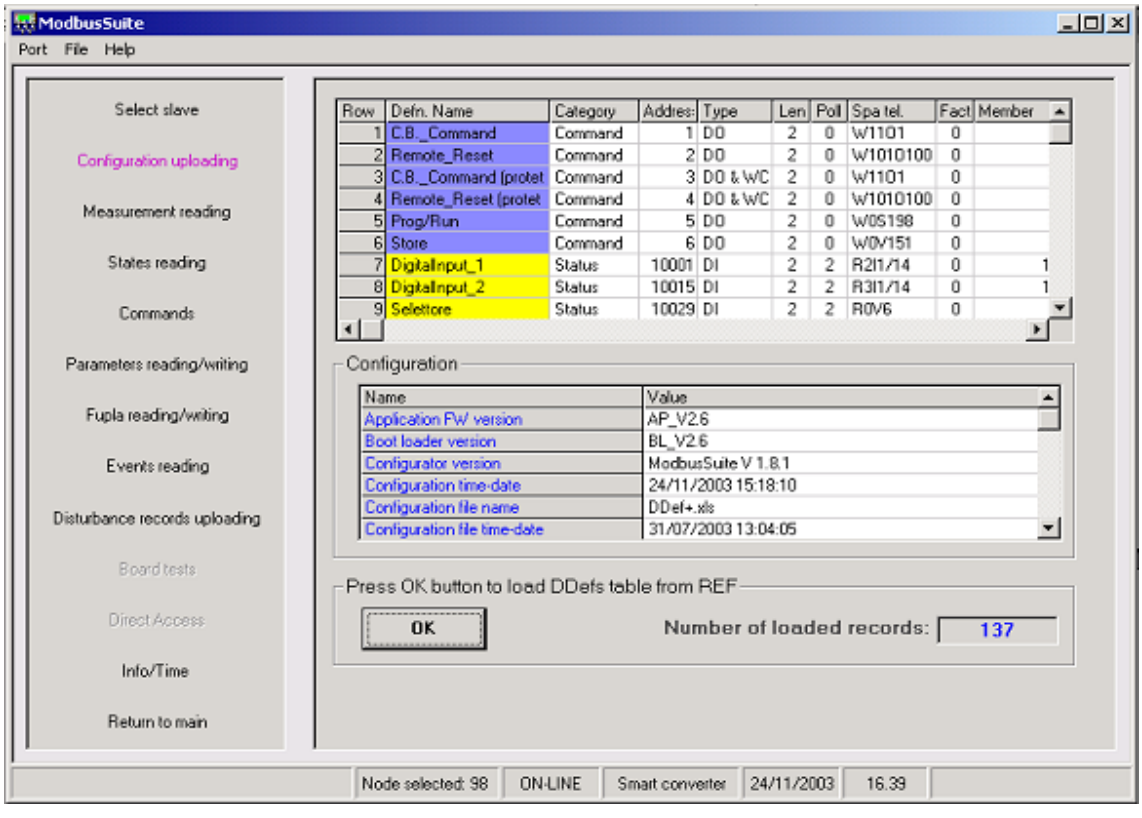


Fig. 14.-1 Commissioning dialog

Now, it is possible to enter the desired section. Each of them has a detailed online help to guide the usage.

15. Configuring SPA communication

To configure the SPA protocol on the Modbus / SPA communication board, you need to configure mainly the baudrate, parity, number of bits and stop bits for each port, since the settings are taken from the Modbus port configuration (the default settings are 19200, n, 8, 1). The SPA settings are instead of 19200 as maximum baudrate, parity even, data bits 7 and stop bits 1. The only way to change the default Modbus settings to the SPA settings is to connect the board with Modbus protocol and to change the settings to the required SPA settings, see File Transfer in
Section 11.1.2. Serial Channel Descriptor table configuration. The communication board will not work with the new settings (the standard number of bits for Modbus protocol is 8, while for SPA protocol is 7), but needs to be reconfigured as a SPA board.

The procedure is:

1. The communication board must be configured as a Modbus board from the REF 542plus Configuration Tool.
2. From the Modbus suite, it is possible to connect the Modbus board and to configure the new channel descriptors with the required SPA protocol channel settings. Scan the network, select the slave REF and then go inside the section “Node On Line”. To send the SPA descriptors, the following operations have to be followed: read the current descriptors, start a session, select the required descriptors and stop the session (see Section 13. Configuring the Modbus board).

   After changing the settings of the channel you are connected to, the Modbus suite loses the connection with the Modbus module due to the fact that the standard number of bits for SPA protocol is 7, while for Modbus protocol it is 8.
3. Configure the communication module with SPA protocol from the REF 542plus Configuration Tool. Select the Configure tab and then the section “hardware”. Then select SPABUS protocol in the **Field bus** drop-down menu, and the channel address in the SPA bus Parameters dialog..

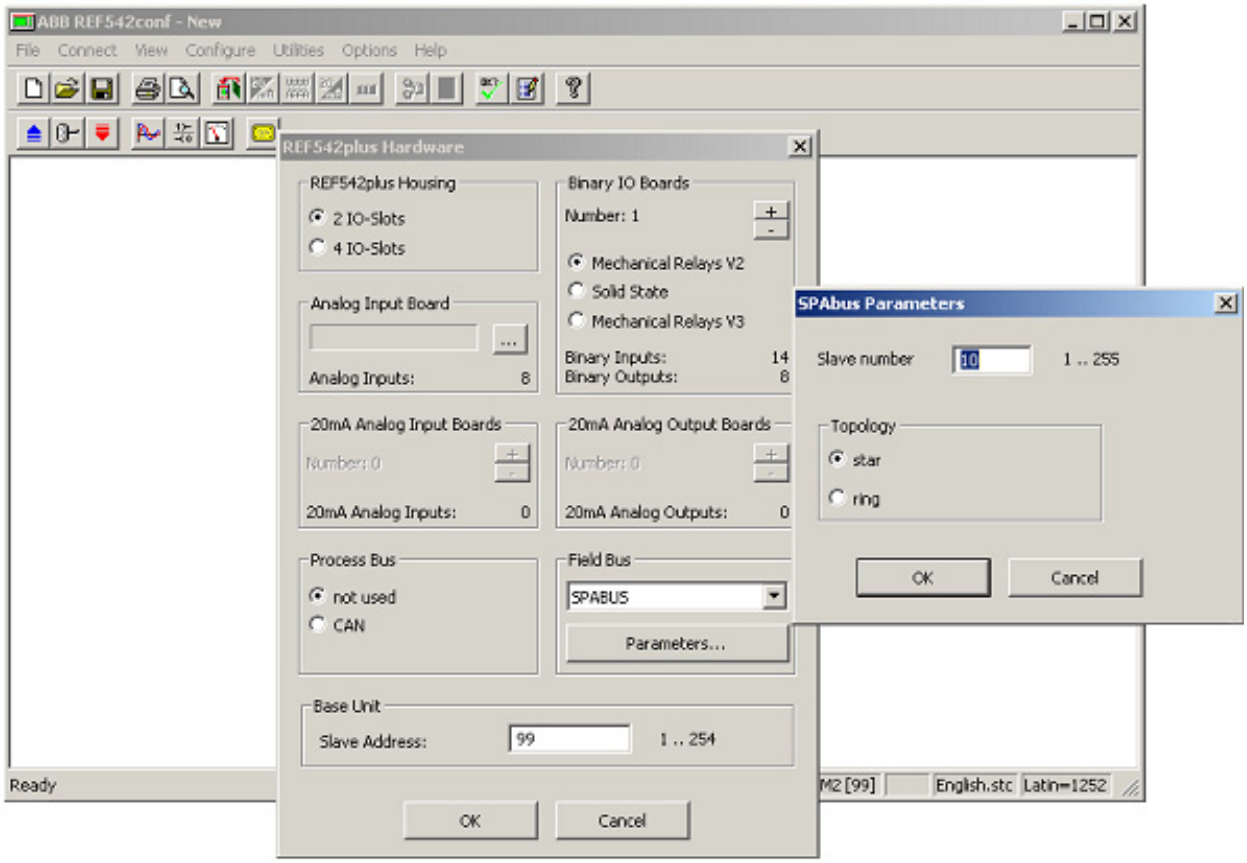


Fig. 15.-1 Configuring SPA protocol

4. Download the configuration file to the main board.

To come back to the Modbus communication, the only way is to reset the Modbus board in order to come back to the default Modbus settings (see Section 13. Configuring the Modbus board).

16. Updating firmware

The firmware update is possible with

- Modbus Suite from the version 1.8
- An automatic handshake RTS485 converter

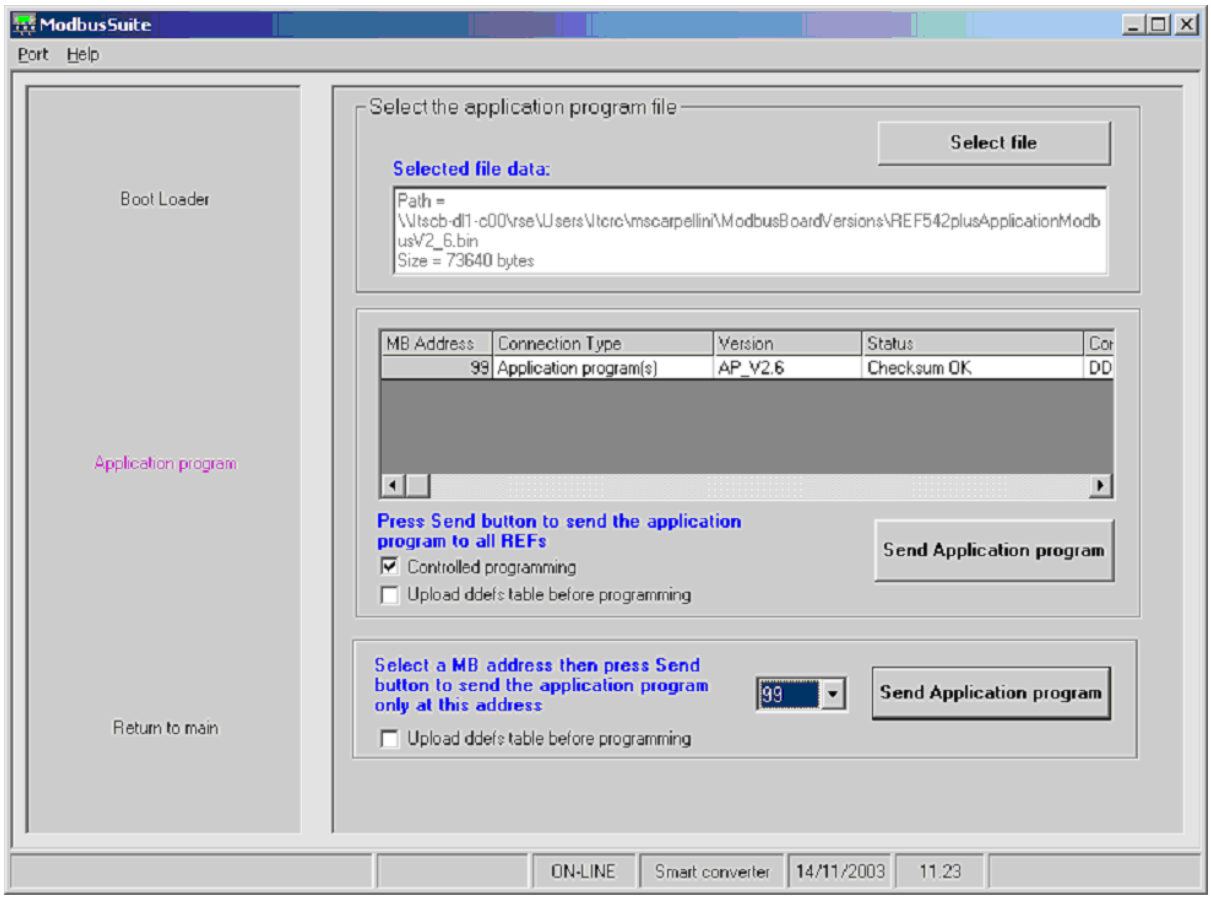
The firmware update is possible only if the minimum previous version is AP_V2.0 and can be performed:

- Peer to peer from the Modbus suite to a single REF 542plus
- On the bus to all the REFs connected

From the Modbus suite it is possible to re-program both the firmware of the boot loader and the application program.

Procedure:

- Scan the network to get all the REFs in the bus and to connect them to the Modbus suite.
- Select the programmer section and choose bootloader or application re-programming.
- Select the binary file whose name must start with "REF542plusBootLoaderModbus" or "REF542plusApplicationModbus".
- It is possible to upgrade more than one REF on the network in broadcast way or a single REF. If you choose to download the upgraded firmware to all the connected REFs, you must enable the flag with controlled programming to check the validity of the downloaded program before the storing in the Flash memory.
- If the procedure is without any error, you have to wait for the restart of the board with the downloaded firmware as described in Section 19. Led lighting sequences with the right blinking.



A051139

Fig. 16.-1 Application re-programming of a single REF 542plus

17. Abbreviations

Abbreviation	Description
AI	Analog Input
AIS	Air Isolated Switchgear
AO	Analog Output
ASCII	American Standard Code for Information Interchange
DDEF	Data Definition
DI	Digital Input
DO	Data Object
EEPROM	Electrically Erasable Programmable Read-Only Memory
FUPLA	Function block Programming Language; Function Plan; Function Chart
GIS	Gas Insulated Switchgear
LED	Light-Emitting Diode
RHMI	Remote Human-Machine Interface as control unit
RTU	Remote Terminal Unit
SCADA	Supervision, Control and Data Acquisition
SPA	Data communication protocol developed by ABB

Appendix A: Modbus error codes

Code	Mnemonic	Description
01	ILLEGAL_FUNCTION	The received message function is not implemented
02	ILLEGAL_DATA_ADDRESS	The address specified in the data field is not an allowable address
03	ILLEGAL_DATA_VALUE	The value specified in the data field is not valid
04	FAILURE_IN_DEVICE	Generic error during interaction between communication board and REF 542plus

Appendix B: Led lighting sequences

The communication board owns 5 leds near the serial connectors:

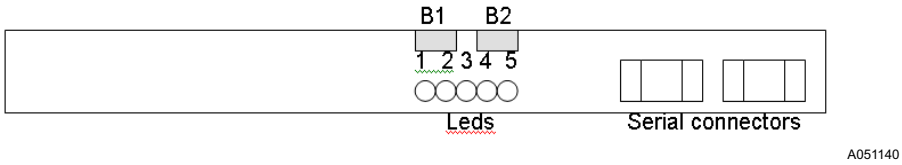


Fig. B-1 Leds on communication board

Leds 2 to 5 are used for traffic monitoring:

- Led 2 = channel 1 Rx
- Led 3 = channel 1 Tx
- Led 4 = channel 2 Rx
- Led 5 = channel 2 Tx

Led 1 is used for communication board monitoring:

- At start-up
- As continuous monitoring
- The communication board owns 2 buttons B1 and B2 over the leds.

Reset the board

- Push together the two buttons B1 and B2 for at least 5 seconds.
- Release the button B2: the first LED 1 blinks 3 times (while the board restarts), then it is off for some seconds and starts blinking continuously with 0.5 s frequency.
- Release the button B1 and the board restarts.

Startup monitoring

At startup, different led sequences are performed according to the different communication board situations:

- Board without BootLoader and without Application Program
  - 2 blinks
- Board with BootLoader but without Application Program
  - 3 blinks
- Board with Application Program but without BootLoader
  - 4 blinks if starting as Modbus or
  - 6 blinks if starting as SPA bus
  - Continuous monitoring after a few seconds
- Board with BootLoader and Application Program starting as Modbus
  - 3 blinks
  - Led1 off for about 5 seconds when the BootLoader is connectable

  - 2 blinks at Application Program start-up
  - Continuous monitoring after a few seconds
- Board with BootLoader and Application Program starting as SPA bus
  - 3 blinks
  - Led1 off for about 5 seconds when the BootLoader is connectable
  - 4 blinks when the Application Program starts
  - Continuous monitoring after a few seconds
- Board with BootLoader re-programming the communication board
  - 8 blinks
  - Led1 off for a few seconds while the BootLoader re-programs
  - Some behaviour as one of the previous cases
- Board with a manual reset
  - 2 blinks
  - Led1 off for a few seconds during the pressing of the reset button
  - 4 blinks
  - Some behaviour as one of the previous cases

Continuous monitoring

The communication board shows its operating status by blinking the Led 1 at different frequencies:

- 0,250 Hz (4 seconds period)

  The Communication board is well configured (if in Modbus mode) and only good messages have been received in the last 30 seconds (on both channels).

- 0,5 Hz (2 seconds period)

  The communication board is well configured (if in Modbus mode) but not all the messages received in the last 30 seconds are good (that is, some communication error has occurred in one or both channels).

- 2 Hz (0,5 seconds period)

  The communication board is badly configured (only if in Modbus mode: the Ddef table is wrong or missing or the polling list is wrong) or only bad messages have been received in the last 30 seconds (on both channels).

If no messages (nor bad messages) are received in the last 30 seconds, the communication board still shows the previous status.

Appendix C: Network communication led on RHMI

The communication network led on RHMI is red both for static or dynamic errors:

- Static errors: at the Modbus board startup, the DDEF is not correct or is absent. The DDEF is not correct if:
  - The Polling field is not right: the only registers that can have a polling different from 0 are the DI (10000 < address < 19999) or the AI (30000 < address < 39999)
- Dynamic errors: at least one communication error in the last 30 seconds (errors as REF Not Ready, illegal data address from REF, illegal data value from REF or low level communication errors as CRC).

The communication led is green if no dynamic errors are present in the last 30 s and the DDEF is correct.

The communication led goes from red to green if at least one packet has been sent with success, otherwise it has an amber color.

1MRS755868 EN 9/2006

ABB Oy
Distribution Automation
P.O. Box 699
FI-65101 Vaasa
FINLAND
+358 10 2211
+358 10 224 1080
www.abb.com/substationautomation