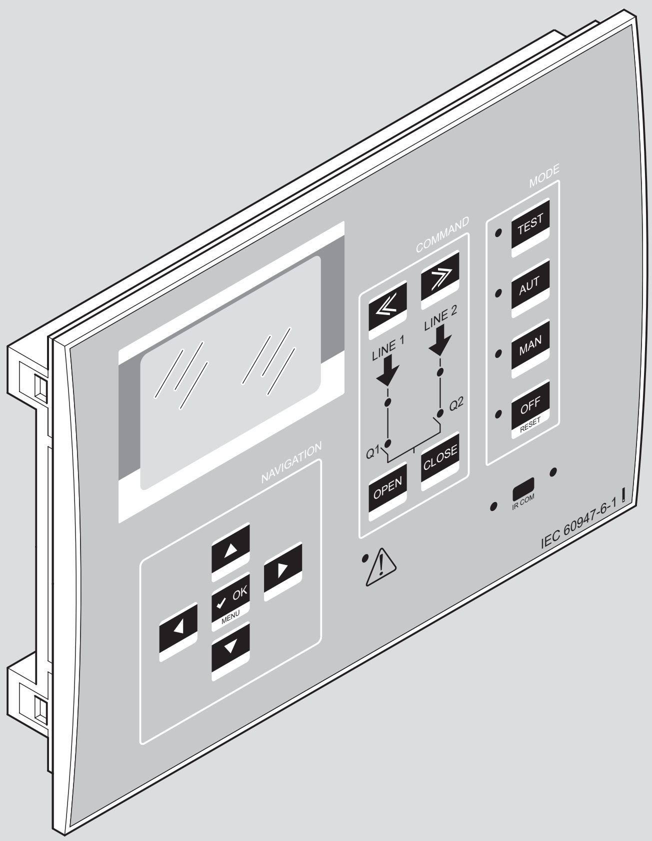
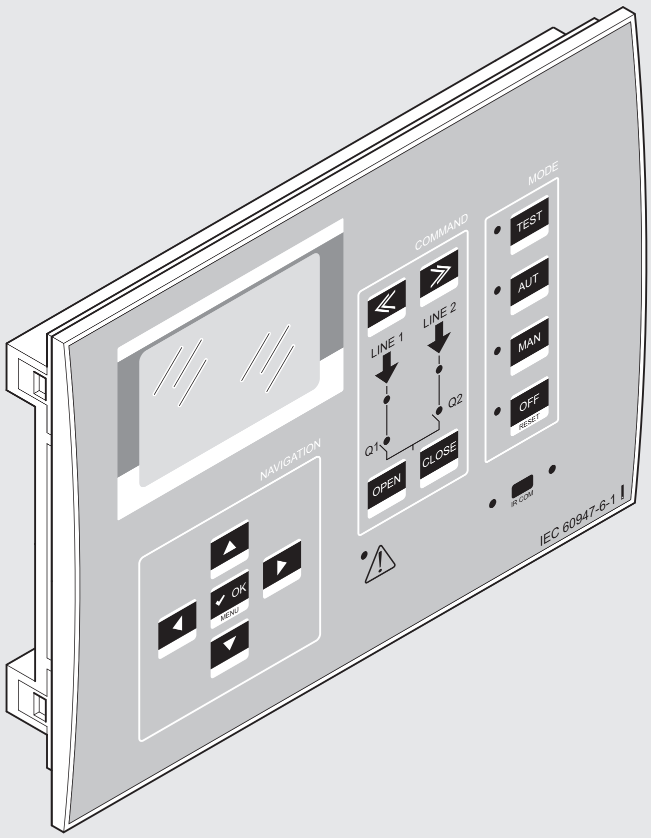
Automatic transfer switch
4 226 83

Modbus communication Protocol



Automatic transfer switch
4 226 83



Index

legrand®

1. Modbus protocol

The 4 226 83 automatic transfer switch controller support the communication protocols Modbus RTU and Modbus ASCII on optical interface and on expansion modules:

- 4 226 87 IR-USB frontal dongle
- 4 226 88 IR-USB frontal dongle

Using this function it is possible to read the device status and to control the unit through supervision software or through other master devices supporting Modbus, like PLCs.

2. Parameters setting

To configure the Modbus protocol, enter SETUP MENU and choose the M10 menu for the desired communication channel port (1 to 3).

Menu parameters

M10– COMMUNICATION (COMn, n=1…3)		UoM	Default	Range
P10.n.01	Node serial address		05	01-247 (248 … 255 internal use)
P10.n.02	Serial port speed	bps	19200	1200 2400 4800 9600 19200 38400 57600 115200
P10.n.03	Data format		8 bit, even	8 bit –no par. 8 bit, odd 8 bit, even 7 bit, odd 7 bit, even
P10.n.04	Stop bits		1	1-2
P10.n.05	Protocol		Modbus RTU	Modbus RTU Modbus ASCII

Note: this menu is divided into 3 sections for communication channels COM1…3. The front IR communication port for connection with **SW ACU** and **APP** via WiFi or USB has fixed communication parameters, so no setup is required.

- **P10.n.01** – Serial (node) address of the communication protocol.
- **P10.n.02** – Communication port transmission speed.
- **P10.n.03** – Data format. 7 bit settings can be used for ASCII protocol only.
- **P10.n.04** – Stop bit number.
- **P10.n.05** – Select communication protocol.
- **P10.n.06…P10.n.08** – Not available.
- **P10.n.09** – Not available.
- **P10.n.10** – Not available.
- **P10.n.11…P10.n.13** – Not available.

3. Modbus RTU protocol

For Modbus RTU protocol, the communication message has the following structure:

T1T2T3	Address (8 bit)	Function (8 bit)	Data (N x 8 bit)	CRC (16 bit)	T1T2T3

- The Address field contains the serial address of the slave destination device.
- The Function field contains the code of the function that must be executed by the slave.
- The Data field contains data sent to the slave or data received from the slave in response to a query (maximum length for data field is 80 16-bit registers, so 160 bytes).
- The CRC field allows the master and slave devices to check the message integrity.
  If a message has been corrupted by electrical noise or interference, the CRC field allows the devices to recognize the error and thereby to ignore the message.
- The T1 T2 T3 sequence corresponds to a time in which data must not be exchanged on the communication bus to allow the connected devices to recognize the end of one message and the beginning of another. This time must be at least 3.5 times the time required to send one character.

The ATS measures the time that elapses from the reception of one character and the following. If this time exceeds the time necessary to send 3.5 characters at the selected baudrate, then the next character will be considered as the first of a new message.

legrand®

4. Modbus functions

The available functions are:

03 = Read Multiple Holding Registers	It allows to read the ATS internal registers
04 = Read input register	It allows to read the ATS inputs
06 = Preset single register	It allows writing parameters
07 = Read exception	It allows to read the device status
10 = Preset multiple register	It allows writing several parameters
17 = Report slave ID	It allows to read information about the device

Example:
To read the number of switching alarms of breaker on line 1 (Q1), which resides at location 58 ($3A_{hex}$), from the ATS with serial address 01, the message to send is the following:

01	04	00	39	00	02	A1	C6

Where:
01= slave address
04 = Modbus function 'Read input register'
00 39 = Address of the required register (number of switching alarms of breaker on Line 1) decreased by one
00 02 = Number of registers to be read beginning from address 22
A1 C6 = CRC Checksum

The ATS answer will be the following:

01	04	04	00	00	00	0A	7B	83

Where:
01 = ATS address (Slave 01)
04 = Function requested by the master
04 = Number of bytes sent by the ATS
00 00 00 0A = Hex value of number of switching alarms of breaker on Line 1 = 10
7B 83 = CRC checksum

4.1 Function 04: read input register

The Modbus function 04 allows to read one or more consecutive registers from the slave memory. The address of each measure is given in the table 7.1. As for Modbus standard, the address in the query message must be decreased by one from the effective address reported in the table. If the measure address is not included in the table or the number of requested registers exceeds the acceptable max number, the ATS will return an error code (see error table in chap. 5).

Example:
Master query:

Slave address	08_{hex}
Function	04_{hex}
MSB address	00_{hex}
LSB address	$0F_{hex}$
MSB register number	00_{hex}
LSB register number	08_{hex}
LSB CRC	$C1_{hex}$
MSB CRC	56_{hex}

In the above example, slave 08 is requested for 8 consecutive registers beginning with address 10_{hex}. Thus, registers from 10_{hex} to 17_{hex} will be returned. As usual, the message ends with the CRC checksum.

Slave response:

Slave address	08 $_{hex}$
Function	04 $_{hex}$
Byte number	10 $_{hex}$
MSB register 10_{hex}	00 $_{hex}$
LSB register 10_{hex}	00 $_{hex}$
--	----
MSB register 17_{hex}	00 $_{hex}$
LSB register 17_{hex}	00 $_{hex}$
LSB CRC	8A $_{hex}$
MSB CRC	B1 $_{hex}$

The answer is always composed of the slave address, the function code requested by the master and the contents of the requested registers. The answer ends with the CRC.

4.2 Function 06: preset single register

This function allows to write in the registers. It can be used only with registers with address higher than 1000_{hex}. For instance, it is possible to change setup parameters. If the value is not in the correct range, the ATS will answer with an error message. In the same way, if the parameter address is not recognised, the ATS will send an error response.
The address and the valid range for each parameter are indicated in Table 7.3.

Example:
Master message:

Slave address	08 $_{hex}$
Function	06 $_{hex}$
MSB address	2F $_{hex}$
LSB address	0F $_{hex}$
MSB register number	00 $_{hex}$
LSB register number	0A $_{hex}$
LSB CRC	31 $_{hex}$
MSB CRC	83 $_{hex}$

Slave response:
The slave response is an echo to the query, that is the slave sends back to the master the address and the new value of the variable.

4. Modbus functions

4.3 Function 07: read exception status

This function allows to read the status of the automatic transfer switch.

Example:
Master query:

Slave address	08_{hex}
Function	07_{hex}
LSB CRC	47_{hex}
MSB CRC	$B2_{hex}$

The following table gives the meaning of the status byte sent by the ATS as answer:

BIT	MEANING
0	Operative mode OFF / Reset
1	Operative mode MAN
2	Operative mode AUT
3	Operative mode TEST
4	On error
5	AC power supply ok
6	DC power supply ok
7	Global alarm on

4.4 Function 16: preset multiple register

This function allows to modify multiple parameters with a single message, or to preset a value longer than one register.

Example:
Master message:

Slave address	08_{hex}
Function	10_{hex}
MSB register address	20_{hex}
LSB register address	01_{hex}
MSB register number	00_{hex}
LSB register number	02_{hex}
Number of byte *(it is the double of above)*	04_{hex}
MSB data	00_{hex}
LSB data	00_{hex}
MSB data	00_{hex}
LSB data	00_{hex}
LSB CRC	85_{hex}
MSB CRC	$3E_{hex}$

Slave response:

Slave address	08_{hex}
Function	10_{hex}
MSB register address	20_{hex}
LSB register address	01_{hex}
MSB byte number	00_{hex}
LSB byte number	02_{hex}
LSB CRC	$1B_{hex}$
MSB CRC	51_{hex}

4.5 Function 17: report slave ID

This function allows to identify the device type.

Example:

Master query:

Slave address	08 $_{hex}$
Function	11 $_{hex}$
LSB CRC	C6 $_{hex}$
MSB CRC	7C $_{hex}$

Slave response:

Slave address	08 $_{hex}$
Function	11 $_{hex}$
Bytes counter	08 $_{hex}$
Data 01 (Type) ❶	76 $_{hex}$
Data 02 (SW release)	01 $_{hex}$
Data 03 (HW release)	00 $_{hex}$
Data 04 (Parameters release)	01 $_{hex}$
Data 05 (product type) ❷	04 $_{hex}$
Data 06 (reserved)	00 $_{hex}$
Data 07 (reserved)	00 $_{hex}$
Data 08 (reserved)	00 $_{hex}$
LSB CRC	B0 $_{hex}$
MSB CRC	2A $_{hex}$

❶ 119 - 77$_{hex}$ = 4 226 83
❷ 2 - 02$_{hex}$ = Legrand serie

5. Errors

In case the slave receives an incorrect message, it answers with a message composed by the queried OR-ed function with 80$_{hex}$, followed by an error code byte.
In the following table the error codes sent by the slave to the master:

CODE	ERROR
01	Invalid function
02	Invalid address
03	Parameter out of range
04	Function execution impossible
06	Slave busy, function momentarily not available

6. Modbus ASCII protocol

The Modbus ASCII protocol is normally used in applications that require to communicate through a couple of modems. The functions and addresses available are the same as for the RTU version, but the transmitted characters are in ASCII and the message end is delimited by Carriage Return CR and Line Feed LF instead of a transmission pause.
If parameter P10.n.05 is set as Modbus ASCII protocol, the communication message on the correspondent communication port has the following structure:

:	Address (2 chars)	Function (2 chars)	Data (N chars)	LRC (2 chars)	CR LF

- The Address field contains the serial address of the slave destination device.
- The Function field contains the code of the function that must be executed by the slave.
- The Data field contains data sent to the slave or data received from the slave in response to a query. The maximum allowable length is of 80 consecutive registers.
- The LRC field allows the master and slave devices to check the message integrity. If a message has been corrupted by electrical noise or interference, the LRC field allows the devices to recognize the error and thereby ignore the message.
- The message ends always with CRLF control character (0D 0A).

Example:
To read the value of the current phase L3, which resides at location 12 ($0C_{hex}$) from the slave with serial address 08, the message to send is the following:

:	08	04	00	0B	00	02	E7	CRLF

Where:
: = ASCII $3A_{hex}$ message start delimiter
08 = slave address
04 = Modbus function ‘Read input register’
00 0B = Address of the required register (L3 current phase) decreased by one
00 02 = Number of registers to be read beginning from address 04
E7 = LRC Checksum
CRLF = ASCII $0D_{hex}$ $0A_{hex}$ = Message end delimiter

The answer is the following:

:	08	04	04	00	00	A8	AE	9B	CRLF

Where:
: = ASCII $3A_{hex}$ message start delimiter
08 = address (Slave 08)
04 = Function requested by the master
04 = Number of bytes sent by the device
00 00 A8 AE = Hex value of the current phase of L3 (= 4.3182 A)
9B = LRC checksum
CRLF = ASCII $0D_{hex}$ $0A_{hex}$ = Message end delimiter

7. Tables

7.1 Measures supplied by serial communication protocol

To be used with functions 03 and 04.

ADDRESS	WORDS	MEASURE	UNIT	FORMAT
02_{hex}	2	Voltage of line 1 L1-N	V	Unsigned long
04_{hex}	2	Voltage of line 1 L2-N	V	Unsigned long
06_{hex}	2	Voltage of line 1 L3-N	V	Unsigned long
08_{hex}	2	Voltage of line 1 L1-L2	V	Unsigned long
$0A_{hex}$	2	Voltage of line 1 L2-L3	V	Unsigned long
$0C_{hex}$	2	Voltage of line 1 L3-L1	V	Unsigned long
$0E_{hex}$	2	Voltage of line 2 L1-N	V	Unsigned long
10_{hex}	2	Voltage of line 2 L2-N	V	Unsigned long
12_{hex}	2	Voltage of line 2 L3-N	V	Unsigned long
14_{hex}	2	Voltage of line 2 L1-L2	V	Unsigned long
16_{hex}	2	Voltage of line 2 L2-L3	V	Unsigned long
18_{hex}	2	Voltage of line 2 L3-L1	V	Unsigned long
$1A_{hex}$	2	Frequency of line 1	Hz/10	Unsigned long
$1C_{hex}$	2	Frequency of line 2	Hz/10	Unsigned long
$1E_{hex}$	2	Battery voltage (DC power supply)	VDC / 10	Unsigned long
20_{hex}	2	Total operation time	s	Unsigned long
22_{hex}	2	Line 1 ok total time	s	Unsigned long
24_{hex}	2	Line 2 ok total time	s	Unsigned long
26_{hex}	2	Line 1 not ok total time	s	Unsigned long
28_{hex}	2	Line 2 not ok total time	s	Unsigned long
$2A_{hex}$	2	Line 1 breaker closed total time	s	Unsigned long
$2C_{hex}$	2	Line 2 breaker closed total time	s	Unsigned long
$2E_{hex}$	2	Breaker opened total time	s	Unsigned long
30_{hex}	2	(not used)	--	Unsigned long
32_{hex}	2	Number of operations of line 1 breaker in AUT	nr	Unsigned long
34_{hex}	2	Number of operations of line 2 breaker in AUT	nr	Unsigned long
36_{hex}	2	Number of operations of line 1 breaker in MAN	nr	Unsigned long
38_{hex}	2	Number of operations of line 2 breaker in MAN	nr	Unsigned long
$3A_{hex}$	2	Number of switching alarms of breaker 1	nr	Unsigned long
$3C_{hex}$	2	Number of switching alarms of breaker 2	nr	Unsigned long
$3E_{hex}$	2	(not used)	--	Unsigned long
50_{hex}	2	Minimum battery voltage	V	Unsigned long
52_{hex}	2	Maximum battery voltage	V	Unsigned long
54_{hex}	2	Maintenance hours line 1	nr	Unsigned long

(continued)

7. Tables

ADDRESS	WORDS	MEASURE	UNIT	FORMAT
56_{hex}	2	Maintenance hours line 2	nr	Unsigned long
58_{hex}	2	Operations to the maintenance of the breaker 1	nr	Signed long
$5A_{hex}$	2	Operations to the maintenance of the breaker 2	nr	Signed long
$21C0_{hex}$	1	OR of all limits	bits	Unsigned int
$1D00_{hex}$	2	Counter CNT 1	UM1	long
$1D02_{hex}$	2	Counter CNT 2	UM2	long
$1D04_{hex}$	2	Counter CNT 3	UM3	long
$1D06_{hex}$	2	Counter CNT 4	UM4	long
$1D08_{hex}$	2	Counter CNT 5	UM5	long
$1D0A_{hex}$	2	Counter CNT 6	UM6	long
$1D0C_{hex}$	2	Counter CNT 7	UM7	long
$1D0E_{hex}$	2	Counter CNT 8	UM8	long
$9A_{hex}$	2	Alarms ❶	bits	Unsigned long
$9C_{hex}$	2	Alarms ❷	bits	Unsigned long

❶ Reading the words starting at address 9A $_{hex}$ will return 32 bits with the following meaning:

BIT	CODE	ALARM
0	A01	Battery voltage too low
1	A02	Battery voltage too high
2	A03	Q1 breaker timeout
3	A04	Q2 breaker timeout
4	A05	Not used
5	A06	Incorrect phase sequence Line S.Q1
6	A07	Incorrect phase sequence Line S.Q2
7	A08	Not used
8	A09	Load timeout not powered
9	A10	Local battery charger failure
10	A11	Genset battery charger 1 failure
11	A12	Genset battery charger 2 failure
12	A13	Not used
13	A14	Emergency
14	A15	Q1 breaker protection trip
15	A16	Q2 breaker protection trip
16	A17	Not used
17	A18	Q1 breaker withdrawn
18	A19	Q2 breaker withdrawn
19	A20	Not used
20	A21	S.Q1 genset line not available
21	A22	S.Q 2 genset line not available
22	A23	Not used
23	A24	Maintenance hours S.Q1
24	A25	Maintenance hours S.Q2
25	A26	Not used
26	A27	Maintenance operations Q1
27	A28	Maintenance operations Q2
28	A29	Not used
29	A30	Auxiliary voltage breaker alarm
30	A31	Non-priority load breaker timeout
31	A32	Tie breaker QC timeout

❷ Reading the words starting at address 9C $_{hex}$ will return 32 bits with the following meaning:

BIT	CODE	ALARM
0	A33	Not used
1	A34	NPL (Non-Priotity-Load) breaker protection trip
2	A35	QC tie breaker protection trip
3	A36	Not used
4	A37	NPL (Non-Priotity-Load) breaker withdrawn
5	A38	QC tie breaker withdrawn
6	A39	Not used
7	UA1	User alarms
8	UA2	User alarms
9	UA3	User alarms
10	UA4	User alarms
11	UA5	User alarms
12	UA6	User alarms
13	UA7	User alarms
14	UA8	User alarms
15	-	(not used)
16	-	(not used)
17	-	(not used)
18	-	(not used)
19	-	(not used)
20	-	(not used)
21	-	(not used)
22	-	(not used)
23	-	(not used)
24	-	(not used)
25	-	(not used)
26	-	(not used)
27	-	(not used)
28	-	(not used)
29	-	(not used)
30	-	(not used)
31	-	(not used)

7. Tables

7.2 Status bits

To be used with functions 03 and 04.

ADDRESS	WORDS	FUNCTION	FORMAT
2070 hex	1	Front panel keyboard status ❶	Unsigned integer
2100 hex	2	Digital inputs status (by pin) ❷	Unsigned integer
2140 hex	2	Digital outputs status (by pin) ❸	Unsigned integer
2074 hex	1	Line 1 voltage status ❹	Unsigned integer
2075 hex	1	Line 1 breaker status ❺	Unsigned integer
2176 hex	1	Line 2 voltage status ❹	Unsigned integer
2177 hex	1	Line 2 breaker status ❺	Unsigned integer
2083 hex	1	Not used	Unsigned integer
2084 hex	1	Not used	Unsigned integer
2078 hex	2	Input function status ❻	Unsigned integer
207A hex	1	Output function status ❼	Unsigned integer
207B hex	1	Display messages status ❽	Unsigned integer
207C hex	1	Controller general status ❾	Unsigned integer
207E hex	1	Frontal LED status	Unsigned integer
207F hex	1	Frontal LED status	Unsigned integer
2085 hex	1	Display massages	Unsigned integer

❶ Following table shows meaning of bits of the word at address 2070_{hex}:

BIT	KEY
0	UP
1	OFF/RESET
2	MAN
3	DOWN
4	AUT/ENTER
5…15	Not used

❷ Following table shows meaning of bits of the word at address 2100_{hex}:

BIT	INPUT
0	Input 1
1	Input 2
2	Input 3
3	Input 4
4	Input 5
5	Input 6
6	Input 7
7	Input 8
8	Input 9
9	Input 10
10	Input 11
11	Input 12
12	Input 13
13	Input 14
14	Input 15
15	Input 16
16	Input 17
17	Input 18
18	Input 19
19	Input 20

❸ Following table shows meaning of bits of the word at address 2140_{hex}:

BIT	OUTPUT
0	Output 1
1	Output 2
2	Output 3
3	Output 4
4	Output 5
5	Output 6
6	Output 7
7	Output 8
8	Output 9
9	Output 10
10	Output 11
11	Output 12
12	Output 13
13	Output 14
14	Output 15
15	Output 16
16	Output 17
17	Output 18
18	Output 19
19	Output 20
20..31	Not used

7. Tables

❹ Following table shows meaning of bits of the word at address 2074_{hex} (Line 1) and 2176_{hex} (Line 2):

BIT	LINE STATUS
0	Line values into limits
1	Line values into limits delayed
2	Voltage into limits
3	Voltage ok
4	Frequency into limits
5	Frequency ok
6	Voltage below min
7	Voltage above max
8	Voltage asymmetry
9	Voltage phase loss
10	Frequency below min
11	Frequency above max
12	Wrong phase sequence
13	All line parameters ok
14-15	Not used

❺ Following table shows meaning of bits of the word at address 2075_{hex} (Line 1) and 2177_{hex} (Line 2):

BIT	BREAKER STATUS
0	Breaker closed
1	Trip alarm
2	Not used
3	Command status (1 = close)
4	Close command output
5	Open command output
6…15	Not used

❻ Following table shows meaning of bits of the word at address 2178_{hex}:

BIT	INPUT FUNCTIONS STATUS
0	Line 1 breaker closed feedback
1	Line 1 breaker trip
2	Not used
3	Line 2 breaker closed feedback
4	Line 2 breaker trip
5	Not used
6	Transfer to secondary line
7	Inhibit return to main line
8	Emergency pushbutton
9	Generator start
10	Generator 1 ready
11	Generator 2 ready
12	Keyboard locked
13	Lock parameters
14	Not used
15	Alarms inhibited

❼ Following table shows meaning of bits of the word at address $207A_{hex}$:

BIT	OUTPUT FUNCTION STATUS
8	Load shed
9	Not used
10	Not used
11	Open all
12	Undervoltage coil 1
13	Undervoltage coil 2
14	Line 1 OK
15	Line 2 OK

❽ Following table shows meaning of bits of the word at address $207B_{hex}$:

BIT	DISPLAY MESSAGE STATUS
0	Generator 1 start
1	Generator 2 start
2	Generator 1 cooling
3	Generator 2 cooling
4	Load transfer 2 ➞ 1
5	Load transfer 1 ➞ 2

❾ Following table shows meaning of bits of the word at address $207C_{hex}$:

BIT	OUTPUT FUNCTION STATUS
0	Operative mode OFF / Reset
1	Operative mode MAN
2	Operative mode AUT
3	Operative mode TEST
4	On error
5	AC power supply present
6	DC power supply present
7	Global alarm on
8…15	Not used

7. Tables

7.3 Commands

To be used with function 06

ADDRESS	WORDS	STATUS
4F00 hex	1	Set remote variable REM1 ❶
4F01 hex	1	Set remote variable REM2
......		
4F07 hex	1	Set remote variable REM8
2F00 hex		Operative mode change ❷
2F0A hex	1	Front panel keystorke simulation ❸
2F03 hex	1	**Value 01** hex: memory save
		Value 04 hex: reboot
2F07 hex	1	**Value 00** hex: Reset device
		Value 01 hex: Reset device and save Fram
2FF0 hex	1	Command menu execution ❹
28FA hex	1	**Value 01** hex**:** Save real time clock setting

❶ Writing AA_{hex} to the indicated address the remote variable will be set to 1, writing BB_{hex} the remote variable will be set to 0.

❷ The following table shows the values to be written to address $2F00_{hex}$ to achieve the correspondent function:

VALUE	FUNCTION
0	Switch to OFF mode
1	Switch to MAN mode
2	Switch to AUT mode

❸ The following table shows the bit position of the value to be written to address $2F0A_{hex}$ to achieve the correspondent function:

BIT	MEANING
0	Key up
1	MAN mode
2	Key right
3	START
4	TEST mode
5	OFF mode
6	AUT mode
7	STOP mode

❹ Writing value between 0 and 15 to the specific address, the correspondent command will be executed:

	MEANING
0	Reset maintenance 1
1	Reset maintenance 2
2	Reset maintenance operations 1
3	Reset maintenance operations 2
4	Reset generic counters CNTx
5	Reset LIMx limits
6	Reset hours counter line 1/line 2
7	Reset hours counter Q 1/ Q 2
8	Reset breaker operation
9	Reset events list
10	Reset default parameters
11	Save parameters in backup memory
12	Reload parameters from backup memory
13	Forced I/O
14	Reset A03 – A04 alarms
15	Simulate line failure

7.4 Device global status

To be used with function 04.

ADDRESS	WORDS	STATUS	FORMAT
2210_{hex}	2	Device global status (bit 0-bit31) ❶	Unsigned integer

❶ Reading two words at address 2210_{hex} will return 32 bits with the following meaning:

BIT	MEANING
0	Device OFF
1	Device in MAN mode
2	Device in AUT mode
3	Device TEST mode
4	Voltage Line 1 OK
5	Voltage Line 2 OK
6	Voltage Line 3 OK
7	Global alarm A
8	Global alarm B
9	Automatic test line 1 in progress
10	Automatic test line 2 in progress
11	Not used
12	Remote control
13	Clock 100 msec
14…31	(not used)

7.5 Real time clock

To be used with functions 04 and 06.
To make effective the changes, store them using the dedicated command described in table 7.3.

ADDRESS	WORDS	FUNCTION	RANGE
$28F0_{hex}$	1	Year	2000..2099
$28F1_{hex}$	1	Month	1-12
$28F2_{hex}$	1	Day	1-31
$28F3_{hex}$	1	Hours	0-23
$28F4_{hex}$	1	Minutes	0-59
$28F5_{hex}$	1	Seconds	0-59

8. Event log reading

To read the events must do the following:

1. Perform the read of 1 register by using **function 04** at address 5030_{hex}, the most significant byte (MSB) indicates how many events are stored (value between 0 to 249), the least significant byte (LSB) is incremented each time an event is saved (value between 0 to 249). Once stored the 250 events the MSB will remain at 249 while the LSB will back to zero and after will continue to increase.
2. Set the index of the event that you want to read (less than the maximum number of events stored), to do this perform **function 06** at 5030_{hex}, specifying which event read.
3. Perform a read of 43 registers (with a single **function 04**) at address 5032_{hex}.
4. The value returned is a string of 86 ASCII characters, showing the same event description ATS visible on the display. The index of the event to be read is incremented automatically after a reading of the register 5032_{hex}, in order to speed up the download of events.
5. If you want to read the next event, repeat step 4, if you want to read any other event do step 3.

Example:

Step 1: Reading events stored.

MASTER
Function = 4 (04_{hex})
Address = 5030_{hex} (5030_{hex} – 0001_{hex} = $502F_{hex}$)
Nr. registers = 1 (01_{hex})

01	04	50	2F	00	01	11	03

ATS
Function = 4 (04_{hex})
Nr. bytes. = 1 (01_{hex})
MSB = 100 (64_{hex})
LSB = 2 (02_{hex})

01	04	02	64	42	13	C1

Step 2: Set the index of the event to read.

MASTER
Function = 6 (06_{hex})
Address = 5030_{hex} (5030_{hex} – 0001_{hex} = $502F_{hex}$)
Value = 1 (01_{hex})

01	06	50	2F	00	01	68	C3

ATS
Function = 6 (06_{hex})
Address = 5030_{hex} (5030_{hex} – 0001_{hex} = $502F_{hex}$)
Value = 1 (01_{hex})

01	06	50	2F	00	01	68	C3

Step 3: Read the event.

MASTER
Function = 4 (04_{hex})
Address = 5032_{hex} (5032_{hex} – 0001_{hex} =5031_{hex})
Nr. registers = 43 ($2B_{hex}$)

01	04	50	31	00	2B	F0	DA

ATS
Function = 4 (04_{hex})
Address = 5030_{hex} (5030_{hex} – 0001_{hex} = $502F_{hex}$)
Nr. bytes = 86 (56_{hex})

String = 2012/07/18;09:34:52;E1100,CHANGE MODE TO: MODE OFF

01	04	56	32	30	31	30	2F	30	31	2F	30	31	3B	30	30	3A	31	34	3A
30	31	3B	45	30															

legrand®

9. Parameter setting

Using the Modbus protocol it is possible to access the menu parameters.
To correctly understand the correspondence between the numeric value and the selected function and/or the unit of measure, please see the ATS operating manual.

PROCEDURE TO READ PARAMETERS

1. Write the value of the menu that you want to read by using the **function 06** at address 5000_{hex} ❶.
2. Write the value of the submenu (if it is present) wanted by using the **function 06** at address 5001_{hex} ❶.
3. Write the value of the parameter wanted by using the **function 06** at address 5002_{hex} ❶.
4. Perform the **function 04** at the address 5004_{hex}, with a number of registers appropriate to the length of the parameter (see table).
5. If wanted the next parameter (in the same menu/submenu) repeat step 4, otherwise perform step 1.

PROCEDURE TO WRITE PARAMETERS

1. Write the value of the menu that you want to change by using the **function 06** at address 5000_{hex} ❶.
2. Write the value of the submenu (if it is present) to change by using the **function 06** at address 5001_{hex} ❶.
3. Write the value of the parameter to change by using the **function 06** at address 5001_{hex} ❶.
4. Perform the **function 16** at address 5004_{hex}, with a number of registers appropriate to the length of the parameter
5. to write the next parameter in the same menu / submenu repeat step 4, otherwise perform step 1, if not go to step 6.
6. To make effective the changes made to setup parameters it is necessary to store the values in memory, using the dedicated command described in table 7 (write value 04 by using **function 06** at address $2F03_{hex}$).

TYPE OF PARAMETER	NUMBER OF REGISTER
Text length 6 characters (ex. M14.0x.06)	3 registers (6 byte)
Text length 16 characters (ex. M14.0x.05)	8 registers (16 byte)
Text length 20 characters (ex. M15.0x.03)	10 registers (20 byte)
Abs(Numeric value) < 32768 (ex M01.05)	1 register (2 byte)
Abs(Numeric value) > 32768 (ex M12.01)	2 registers (4 byte)
IP address (ex. M08.0x.06 M08.0x.07)	2 registers (4 byte)

❶ It's possible to read menus, submenus, and parameter stored at the addresses 5000_{hex}, 5001_{hex} and 5002_{hex} by using **function 04**.

Example:
Set to 8 the value of parameter M08.01.01

Step 1: Set menu 08.

MASTER
Function = 6 (06_{hex})
Address = 5000_{hex} ($5000_{hex} - 0001_{hex} = 4FFF_{hex}$)
Value = 8 (08_{hex})

01	06	4F	FF	00	08	AE	E8

ATS
Function = 6 (06_{hex})
Address = 5000_{hex} ($5000_{hex} - 0001_{hex} = 4FFF_{hex}$)
Value = 8 (08_{hex})

01	06	4F	FF	00	08	AE	E8

Step 2: Set submenu 01.

MASTER

Function = 6 (06_{hex})
Address = 5001_{hex} (5001_{hex} – 0001_{hex} =5000_{hex})
Value = 1 (01_{hex})

01	06	50	00	00	01	59	0A

ATS

Function = 6 (06_{hex})
Address = 5001_{hex} (5001_{hex} – 0001_{hex} =5000_{hex})
Value = 1 (01_{hex})

01	06	50	00	00	01	59	0A

Step 3: Set parameter 01.

MASTER

Function = 6 (06_{hex})
Address = 5002_{hex} (5002_{hex} – 0001_{hex} = 5001_{hex})
Value = 1 (01_{hex})

01	06	50	01	00	01	08	CA

ATS

Function = 6 (06_{hex})
Address = 5002_{hex} (5002_{hex} – 0001_{hex} = 5001_{hex})
Value = 1 (01_{hex})

01	06	50	01	00	01	08	CA

Step 3: Set value 8.

MASTER

Function = 16 (10_{hex})
Address = 5004_{hex} (5004_{hex} – 0001_{hex} =5003_{hex})
Nr. Register = 1 (01_{hex})
Nr. bytes = 2 (02_{hex})
Value = 8 (0008_{hex})

01	10	50	03	00	02	04	00	00	00	08	4E	7F

ATS

Function = 16 (10_{hex})
Address = 5004_{hex} (5004_{hex} – 0001_{hex} =5003_{hex})
Value = 2 (02_{hex})

01	10	50	03	00	02	A0	C8

Step 6: Save and reboot.

MASTER

Function = 6 (06_{hex})
Address = $2F03_{hex}$ ($2F03_{hex}$ – 0001_{hex} = $2F02_{hex}$)
Value = 4 (04_{hex})

01	6	2F	02	00	04	21	1D

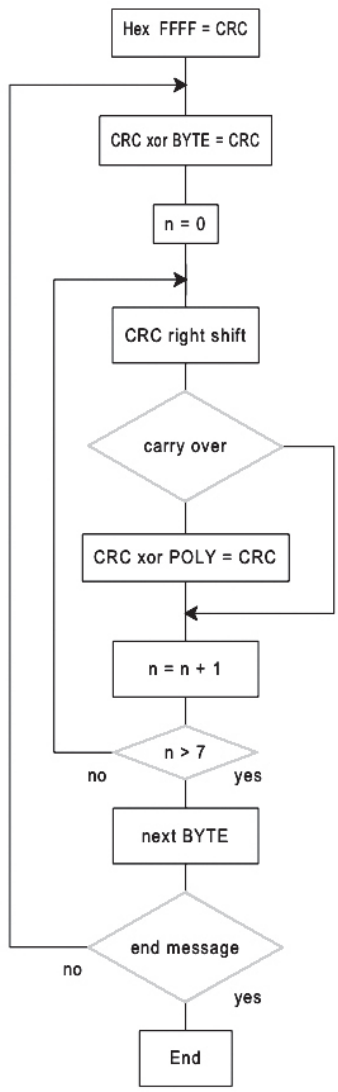
ATS

No answer.

A. CRC calculation (checksum for RTU)

CRC calculation algorithm
Example: Frame = 0207h

CRC initialization	**1111**	**1111**	**1111**	**1111**
Load the first byte			0000	0010
Execute xor with the first Byte of the frame	1111	1111	1111	1101
Execute 1st right shift	0111	1111	1111	1110 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1101	1111	1111	1111
Execute 2nd right shift	0110	1111	1111	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1100	1111	1111	1110
Execute 3rd right shift	0110	0111	1111	1111 **0**
Execute 4th right shift	0011	0011	1111	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1001	0011	1111	1110
Execute 5th right shift	0100	1001	1111	1111 **0**
Execute 6th right shift	0010	0100	1111	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1000	0100	1111	1110
Execute 7th right shift	0100	0010	0111	1111 **0**
Execute 8th right shift	0010	0001	0011	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Load the second byte of the frame			0000	0111
Execute xor with the second byte of the frame	1000	0001	0011	1001
Execute 1st right shift	0100	0000	1001	1100 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1110	0000	1001	1101
Execute 2nd right shift	0111	0000	0100	1110 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1101	0000	0100	1111
Execute 3rd right shift	0110	1000	0010	0111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1100	1000	0010	0110
Execute 4th right shift	0110	0100	0001	0011 **0**
Execute 5th right shift	0010	0100	0000	1001 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1001	0010	0000	1000
Execute 6th right shift	0100	1001	0000	0100 **0**
Execute 7th right shift	0010	0100	1000	0010 **0**
Execute 8th right shift	0001	0010	0100	0001 **0**
CRC Result	**0001**	**0010**	**0100**	**0001**
		12_{hex}		**41_{hex}**



Note: The byte 41_{hex} is sent first(even if it is the LSB), then12_{hex} is sent.

B. LRC CALCULATION (CHECKSUM for ASCII)

Example:

Address	01		00000001
Function	04		00000100
Start address high	00		00000000
Start address low	00		00000000
Number of registers	08		00001000
		Sum	00001101
1. complement			11110010
		+ 1	00000001
2. complement			11110101
LRC result			***F5*** $_{hex}$

LEGRAND
Pro and Consumer Service
BP 30076 - 87002
LIMOGES CEDEX FRANCE
www.legrand.com

Installer stamp