GUNNEBO®
For a safer world

Flap Gate for Airport and Mass Transit

Model: MFL / AFL gate

P0035

ModBus protocol description

Protocol version 2.0

Document:	Revision:
TE0073A	**04R**

DOCUMENT HISTORY

Rev.	**Data** ***Date***	**Description/** ***Descrizione***	**Author**	**Check**	**Approval**
00	23/01/2013	*First Issue*	VM	LC	VM
01	07/06/2013	*Upgraded the whole document*	VM	LC	VM
02	26/07/2013	*REG 0: updated Modbus version;* *REG 21, REG 131 to 138: 0x0400 bit – fraud alarm;* *REG 35: runtime buzzer management;* *REG 49: added PAX_PRESENCE bit;* *REG 57 to 60: changed definition and default;* *Table 5 and 6: updated.* *REG 189 to 252: changed default value*	VM	LC	VM
03	28/10/2013	*REG 0: updated Modbus version;* *REG 42: changed the definition (Jira MFL-43);* *REG 86: improved description* *REG 89: name and description have been changed in coherence with the real functionality (Jira MFL-43);* *Table 7: updated because of REG 89 name change*	VM	LC	VM
04	26/11/2013	*Added:* *Fault Device Messages;* *Color Table Cases;* *Tables 8,9,10 (Jira MFL-41) and 11 (Jira MFL-75);* *PAR 17,73 and 74 (Jira MFL-61);* *Removed:* *REG 36 and 37 (Jira MFL-67);* *PAR 87,88 and 94 (Jira MFL-67);* *PAR 79 (Jira MFL-75);* *Upgraded:* *Table 7 (Jira MFL-75);* *REG 41 (Jira MFL-26);* *REG 42,51,52,53 and 54 (Jira MFL-45);* *PAR 77,86,89,90,91,98,99,100,101,102, 103,106,142,143,149,151,157,163,164, 165,174 and 175 (Jira MFL-45);* *PAR 67 and 78 (Jira MFL-75);*	NB	LC	VM

SUMMARY

FIGURES

TABLES

1 Introduction

The scope of this document is to specify the Modbus RTU implementation for the MFL /AFL Gate product range.

The current implementation is based on the Modbus RTU standard, version 2.0.

In this document, the map and description of available registers is provided for development purpose.

2 Modbus RTU Standard

Starting from the Modbus RTU standard definitions, this chapter summarizes the fundamental concepts that have been implemented in the MFL /AFL Gate application.

The Modbus protocol is a serial protocol defining the format of the data exchanged, the synchronization between transmission and reception, and exception handling.

The Modbus protocol can be based on character or RTU (binary) encoding; the latter was chosen for MFL/AFL Gates because of its efficiency.

2.1 Modbus Data Format

The Modbus RTU (remote terminal unit) makes use of a compact, binary representation of the data for protocol communication.

Although the protocol allows a variety of serial communication settings, the MFL/AFL Gate version implements only the following one:

1 Start Bit, 8 bit data, 1 stop bit, No parity.

The byte are packaged with the following structure:

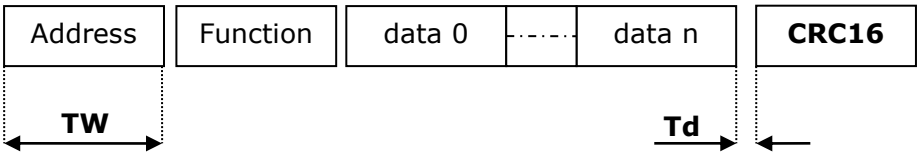

Figure 1: Modbus frame format

- **TW** is the duration time of a transmitted byte;
- **Td** is the gap between bytes within the same frame;

The following rule applies: **Td < 3.5 x TW**.

The Address field is the address of a target slave, in the range **1-247**; address **0** is reserved for broadcast messages.

The CRC16 field is the standard CRC16 polynomial parity algorithm for Modbus.

2.2 Modbus Synchronization

The transmission follows the classic Master – Slave scheme.

The maximum number of addressable slaves is 247; this protocol limit could be lowered by network physical limits (the RS485 bus allows only 31 devices in a single net).

There are only two possible transition schemes:

1) master sends a message to a defined slave - slave answer with a proper packet;
2) master sends a broadcast message - no answer from the slaves;

In case of transaction 1, the maximum delay time the master should wait for a slave answer is a parameter agreed among all the nodes in the net, along with the baud rate and the data encoding. In any case, this waiting time cannot be less than **3.5 × TW**.

2.3 Error Handling

During a transaction, there can be two types of errors:

1) corrupted data during the transmission;
2) incoherent data or data not available.

Corrupted Data

The data in the packet is considered corrupted when the CRC16 field is wrong. In case the packet is corrupted, there are several different actions:

- slave receives a corrupted packet: it discards that packet and do not answer the master;
- slave receives a broadcast corrupted packet: it discards that packet (no answer is expected anyway);
- master receives a corrupted answer packet from a remote slave: it discards that packet and arbitrarily retries the last transaction;
- master does not receive any answer from a polled slave before the maximum delay time expires: it arbitrarily retries the last transaction.

Incoherent Data or Data not Available

Although a received message can have a correct **CRC16,** its contents may not be valid for the addressed host: for example, the server could ask for the content of an internal register, but the register is not present in the slave. In this case, the slave will answer with an ERROR PACKET composed of: the Address, the received function field with the most significant bit set to 1, and an error code.

For example:

The master sends a command 0x0A to the Slave #0x01 with 1 data byte, but the slave cannot handle this specific command. The slave will answer with the following data packet:

0x01	0x**8**A	**<error**	**CRC16**

The standard error codes are:

Table 1: Error code messages

Code	Name	Meaning
0x01	Illegal Function	The addressed Slave cannot handle the function
0x02	Illegal data address	The address indicated in the data field is not compatible with the addressed Slave
0x03	Illegal data value	The data value isn't admissible with the required function
0x04	Slave failure	Slave failure for unrecognized reason due to an internal problem
0x05	Acknowledgment	The data requires more interrogations
0x06	Slave Busy	The command cannot be executed because the system is momentarily not available

2.4 Data Allocation

The Modbus protocol organizes data in registers. There are four different register types, allocated in four different address spaces:

- Discrete Inputs Registers: bit oriented, read-only data. The address <n> refers to the <n>th bit in the area.
- Coils Registers: bit oriented, read/write data. The address <n> refers to the <n>th bit in the area.
- Input Registers: 16-bit word oriented, read-only data. The address <n> refers to the <n>th word in the array.
- Holding Registers: 16-bit word oriented, read/write data. The address <n> refers to the <n>th word in the array.

Any type can allow 65536 different addresses.

The data type can be accessed only by the functions that operate over those data. What follows is the function-data correspondence:

Table 2: Registers allocation

Data type	Function code	Function description
Discrete Input Registers	0x2	READ INPUT STATUS
Coils Registers	0x1	READ COILS
	0x5	FORCE SINGLE COIL
	0xf	FORCE MULTIPLE COILS
Input Registers	0x4	READ INPUT REGISTER
Holding Registers	0x3	READ HOLDING REGISTER
	0x6	PRESET SINGLE REGISTER
	0x10	PRESET MULTIPLE REGISTERS

3 MFL /AFL Implementation

In the MFL / AFL Gate application all the data are allocated into the **Holding Register** space.
The data are divided into two logic groups, depending on the physical memory allocation:

- "Registers": data are in RAM memory;
- "Parameters": data are in EEPROM memory.

3.1 General setup

The following protocol attributes can be set by mean of internal parameters:

- General Enable Modbus communication;
- Baud rate: from 9600 to 115200 b/s;
- Serial link channel: RS485/RS232 physical interfaces;
- Identification address ID: from 1 to 247;
- Timeout communication;
- End of Frame timeout settable from 4ms to 1s;
- Answer delay time: from 4ms to 1s;
- Enable register data modification;
- Enable Password for register data modification;
- 64bit Password setting (if enabled).

THIS PAGE INTENTIONALLY LEFT BLANK

3.2 Registers Map

For every register, the read (R) or write (W) accessibility is indicated. If a register is read-only it means that it is safe to read but it must not be written. Please note that writing to a read-only register can lead to unpredictable functionality. All registers are 16-bit unsigned short integers.

#		Register Name	Description	Default	Unit	Min	Max
0	*R*	*REG_MODBUS_ VERSION*	H - Modbus protocol major version; L - Modbus protocol minor version.	*0x0200*		*0x0*	*0xFFFF*
1	*R*	*REG_FIRMWARE_ VERSION*	H – Firmware project number; L - Firmware release number.			*0x0*	*0xFFFF*
2	*R*	*REG_FIRMWARE_ REVISION*	Identifies the firmware within Gunnebo's version control system.			*0x0*	*0xFFFF*
3		*REG_MDB_PSW_OK*	The Modbus Password has been inserted				
4		*REG_MDB_PSW_ ATTEMPT*	The residual attempt to enter a new password				
5	*R*	*REG_AISLE_MODE*	Aisle Mode: AISLE_NO / AISLE_NC 0 = AISLE_NC Normally Closed Aisle 1 = AISLE_NO Normally Open Aisle	*PAR_AISLE_MODE*		*0*	*1*

#		Register Name	Description	Default	Unit	Min	Max
6	*R*	*REG_RUNNING_MODE*	Current Running Mode 0 = *TRANSIT_MODE* The Running mode is in Managing a Transit 1 = *FIRE_ALARM_MODE* The Running mode is managing a Fire Alarm condition 2 = *CAN_TIMEOUT_MODE* The CAN is not connected 3 = *POWERDOWN* The Power-down event has been signalled 4 = *HOLD_OPEN_MODE* Hold Open Mode 5 = *HOLD_CLOSE_MODE* Hold Close Mode 6 = *FAULT_MODE* Fault Condition	*0*		*0*	*6*
7	*R*	*REG_PRIORITY*	Current defined priority (direction) 0 = *PRIORITY_NOT_DEFINED* The Priority has not defined yet 1 = *PRIORITY_A* The Access Priority is set to A 2 = *PRIORITY_B* The Access priority is set to B	*0*		*0*	*2*
8	*R*	*REG_MODE_A*	Current Transit Mode in A direction 1 = *LOCKED_MODE* Locked mode Transit 2 = *FREE_MODE* Free mode Transit 3 = *CONTROLLED_MODE* Controlled mode transit	*PAR_MODE_A*		*0*	*3*
9	*R*	*REG_STACK_A*	Current Stack A content Number of validation in reader stack A	*0*		*0*	*255*
10	*R*	*REG_MODE_B*	Current Transit Mode in B direction 1 = *LOCKED_MODE* Locked mode Transit 2 = *FREE_MODE* Free mode Transit 3 = *CONTROLLED_MODE* Controlled mode transit	*PAR_MODE_B*		*0*	*3*

#		Register Name	Description	Default	Unit	Min	Max
11	*R*	*REG_STACK_B*	Current Stack B content Number of validation in reader stack A	*0*		*0*	*255*
12	*R*	*REG_FAULT_1*	Fault Register #1 (see Table 8)	-		-	-
13	*R*	*REG_FAULT_2*	Fault Register #2 (see Table 8)	-		-	-
14	*R*	*REG_FAULT_3*	Fault Register #3 (see Table 8)	-		-	-
15	*R*	*REG_FAULT_4*	Fault Register #4 (see Table 8)	-		-	-
16	*R*	*REG_FRAUD*	Fraud Code 0 = *FRAUD_NONE* No Fraud pending 1 = *FRAUD_INTRUSION_ZONE_1_VALIDATION* The Zone 1 (Entry) sensors have been obscured too long during the validation process 2 = *FRAUD_INTRUSION_ZONE_2_VALIDATION* The Zone 2 (Exit) sensors have been obscured too long during the validation process 3 = *FRAUD_INTRUSION_BLOCKED_AISLE* A Person entered a locked side 4 = *FRAUD_INTRUSION_LONG_TRANSIT* A Person took too time to end the transit 5 = *FRAUD_TAIL_GATING* Tailgating has been detected 6 = *FRAUD_WRONG_WAY* Wrong Way has been detected	*0*		*0*	*6*

#		Register Name	Description	Default	Unit	Min	Max
17		*REG_FRAUD_COUNT*	Fraud with success	*0*		*0*	*255*
18	*R*	*REG_SENSORS*	Sensors activation	*0x0*		*0x0*	*0xFFFF*
19	*R*	*REG_SENSOR_MASK*	Sensor Mask This mask is fixed and derived from Cabinet configuration and shows all the sensors available in the aisle. A bit 1 means sensor available.	*0x0*		*0x0*	*0xFFFF*
20	*R*	*REG_SENSOR_TRANSIT_MASK*	Sensor Transit Exclusion Mask This mask is dynamically calculated and derived from Paddle configuration. It shows all the sensors available from the half aisle during a transit when doors are open in a direction or in another. Some sensors are not available due to physical paddle interference.	*0x0*		*0x0*	*0xFFFF*
21	*R*	*REG_FLAGS*	Status Flags • 0x0001: reader A enable/disable • 0x0002: reader B enable/disable • 0x0004: counting pulse in direction A • 0x0008: counting pulse in direction B • 0x0010: reader A acknowledge • 0x0020: reader B acknowledge • 0x0040: priority in direction A • 0x0080: priority in direction B • 0x0100: doors are closed • 0x0200: doors are open • 0x0400: fraud alarm • 0x0800: out of service • 0x1000: fire alarm echo • 0x2000: power fail alarm	*0x0*		*0x0*	*0xFFFF*

#		Register Name	Description	Default	Unit	Min	Max
22		*REG_LAST_READONLY_REGISTER*	Don't move from this position	*0*		*0*	*0*
23		*REG_MDB_PSW_0*	Special registers to write the Modbus password 0	*0*		*0*	*65535*
24		*REG_MDB_PSW_1*	Special registers to write the Modbus password 1	*0*		*0*	*65535*
25		*REG_MDB_PSW_2*	Special registers to write the Modbus password 2	*0*		*0*	*65535*
26		*REG_MDB_PSW_3*	Special registers to write the Modbus password 3	*0*		*0*	*65535*
27	*R/W*	*REG_EXT_CONTROL*	Enable or Disable the External Control over the cabinet. When external control is enabled, the following registers controls the gate: *REG_EXT_RUNNING_MODE* *REG_EXT_AISLE_MODE* *REG_EXT_MODE_A* *REG_EXT_MODE_B* 0 = DISABLED 1 = ENABLED	*0*		*0*	*1*
28	*R/W*	*REG_EXT_RUNNING_MODE*	Sets the External Running mode when *REG_EXT_CONTROL* is enabled 0 = *TRANSIT_MODE* The Running mode is in Managing a Transit 1 = *FIRE_ALARM_MODE* The Running mode is managing a Fire Alarm condition 2 = *CAN_TIMEOUT_MODE* The CAN is not connected 3 = *POWERDOWN* The Power-down event has been signalled 4 = *HOLD_OPEN_MODE* Hold Open Mode 5 = *HOLD_CLOSE_MODE* Hold Close Mode 6 = *FAULT_MODE* Fault Condition	*0*		*0*	*6*

#		Register Name	Description	Default	Unit	Min	Max
29	*R/W*	*REG_EXT_AISLE_MODE*	Sets the External Aisle mode when *REG_EXT_CONTROL* is enabled	*PAR_AISLE_MODE*		*0*	*3*
30	*R/W*	*REG_EXT_MODE_A*	Current Transit Mode in A direction sets by external when *REG_EXT_CONTROL* is enabled	*PAR_MODE_A*		*0*	*3*
31	*R/W*	*REG_EXT_MODE_B*	Current Transit Mode in B direction sets by external when *REG_EXT_CONTROL* is enabled	*PAR_MODE_B*		*0*	*3*
32	*R/W*	*REG_EXT_READER_A_MAX_STACK*	Max Number of Validation in Reader A	*PAR_READER_A_MAX_STACK*		*0*	*10*
33	*R/W*	*REG_EXT_READER_B_MAX_STACK*	Max Number of Validation in Reader B	*PAR_READER_B_MAX_STACK*		*0*	*10*
34	*R/W*	*REG_EXT_VALIDATION_SENSORS*	Specify the zone where Validation is NOT Allowed. • 0: Zone disabled • 1: S1 | S2 | S3 | S4 | S5 | S6 | S7 • 2: S2 | S3 | S4 | S5 | S6 | S7 • 3: S3 | S4 | S5 | S6 | S7 • 4: S4 | S5 | S6 | S7 • 5: S5 | S6 | S7 • 6: S6 | S7 • 7: S7	*PAR_VALIDATION_SENSORS*		*0*	*7*
35	*R/W*	REG_BUZZER_RUNTIME	The host writes the single beep time duration into register. When the sound stops, the register is cleared.	0	128 ms	0	*0xFFFF*
36			Not used				
37			Not used				

#		Register Name	Description	Default	Unit	Min	Max
38	*R*	*REG_EXT_ COUNT_A*	This is the count of the number of passages in direction A (including tailgating)	*0*		*0*	*65535*
39	*R*	*REG_EXT_ COUNT_B*	This is the count of the number of passages in direction B (including tailgating)	*0*		*0*	*65535*
40	*R/W*	*REG_EXT_ BUZZER_ENABLE*	This Enables/Disables the Buzzer activation	*PAR_BUZZER_ ENALBLE*		*0*	*1*
41	*R/W*	*REG_EXT_ALARMS*	Enable / Disable the diagnostic on the gate's sub-systems • 0: - • 1: enable fraud detection • 2: enable wrong way detection • 3: - • 4: - • 5: - • 6: - • 7: - • 8: enable encoder faults • 9: enable safety device faults • 10: enable relay faults • 11: enable sensor faults • 12: enable motor faults • 13: - • 14: - • 15: -	*PAR_FIX_SENSOR_ FAULT_ENABLE*		*0*	*1*
42	*R/W*	*REG_EXT_ PASSAGE_CANC_ TIMEOUT*	Maximum time to pass through the gate after validation (writing this register has no effect for current transit)	*PAR_PASSAGE_ CANC_TIMEOUT*	*10 ms*	*0*	*65535*

<table>
<tr><th>#</th><th></th><th>Register Name</th><th>Description</th><th>Default</th><th>Unit</th><th>Min</th><th>Max</th></tr>
<tr><td>43</td><td></td><td>REG_EXT_DRIVER_FAULT_ENABLE</td><td>Enable the driver fault</td><td>PAR_DRIVER_FAULT_ENABLE</td><td></td><td>0</td><td>1</td></tr>
<tr><td>44</td><td>R/W</td><td>REG_EXT_FAULT_DOOR_STATUS</td><td>Sets the status of the door when in fault
0: close
1: open A
2: open B
3: stop
4: disabled</td><td>PAR_FAULT_DOOR_STATUS</td><td></td><td>0</td><td>4</td></tr>
<tr><td>45</td><td>R/W</td><td>REG_EXT_TRANSIT_A_AUTH</td><td>Performs a single or multiple transit on side A<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>CM</td><td>CS</td><td>M</td><td>S</td></tr></table>For more info see the Table 3</td><td>0</td><td></td><td>0</td><td>3</td></tr>
<tr><td>46</td><td>R/W</td><td>REG_EXT_TRANSIT_B_AUTH</td><td>Performs a single or multiple transit on side B<table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr><tr><td>-</td><td>-</td><td>-</td><td>-</td><td>CM</td><td>CS</td><td>M</td><td>S</td></tr></table>For more info see the Table 3</td><td>0</td><td></td><td>0</td><td>3</td></tr>
<tr><td>47</td><td>R/W</td><td>REG_PRINTER_LED</td><td>This controls the LED illumination around the ticket printer throat
For more info see the Table 4.</td><td>0</td><td></td><td>0</td><td>4</td></tr>
</table>

#		Register Name	Description	Default	Unit	Min	Max
48	*R/W*	*REG_BUTTON*	This is SET by the gate to indicate that the internal "Reset" button has been pressed by Airport Ground Staff. It is not normally SET by the host. This may be CLEARED by the host to 0. • 0: Unknown (i.e. cleared) • 1: Not pressed (this state is not currently used by the gate) • 2: Pressed	*0*		*0*	*2*
49	*R/W*	*REG_SUMMARY*	Transit information summary • 9: *PAX_PRESENCE* • 8: *PAX_CLEAR_BTN* • 7: *PAX_EXIT_FRAUD* • 6: *PAX_ENTRY_FRAUD* • 5: *PAX_TAILGATE* • 4: *PAX_NOT_READY* • 3: *PAX_READY* • 2: *PAX_CANCEL* • 1: *GATE_OK* • 0: *PAX_DETECTED* For more info see the Table 5	*0x0*		*0x0*	*0xFFFF*
50	*R/W*	*REG_CTRL_CLOSE_ DELAY*	Close delay time in Controlled mode	*PAR_CTRL_CLOSE_ DELAY*	*16 ms*	*0*	*65535*
51	*R/W*	*REG_BUZZER_ SHUT_TIMER*	Shut the buzzer after a time out	*PAR_BUZZER_ SHUT_TIMER*	*10 ms*	*0*	*65535*
52	*R/W*	*REG_INTRUSION_ ZONE_1_TIMEOUT*	Validation zone 1 time out	*PAR_INTRUSION_ ZONE_1_TIMEOUT*	*10 ms*	*0*	*65535*

<table>
<tr><th>#</th><th></th><th>Register Name</th><th>Description</th><th>Default</th><th>Unit</th><th>Min</th><th>Max</th></tr>
<tr><td>53</td><td>R/W</td><td>REG_INTRUSION_ ZONE_2_TIMEOUT</td><td>Validation zone 2 time out</td><td>PAR_INTRUSION_ ZONE_2_TIMEOUT</td><td>10 ms</td><td>0</td><td>65535</td></tr>
<tr><td>54</td><td>R/W</td><td>REG_WRONG_ WAY_TIMEOUT</td><td>Timeout for the exit sensor in Wrong Way alarm</td><td>PAR_WRONG_ WAY_TIMEOUT</td><td>10 ms</td><td>0</td><td>65535</td></tr>
<tr><td>55</td><td>R/W</td><td>REG_MODBUS_ RW_COUNTS</td><td>This may be used to check for missing Modbus communications.
H Wrap-around count of Modbus write commands;
L Wrap-around count of Modbus read commands</td><td>0</td><td></td><td>0</td><td>655535</td></tr>
<tr><td>56</td><td>R/W</td><td>REG_LIGHT_COLOR _TABLE_NUMBER</td><td>Color table for lights management
For more info see the Table 6</td><td>PAR_LIGHT_ COLOR_TABLE_ NUMBER</td><td></td><td>0</td><td>3</td></tr>
<tr><td>57</td><td>R/W</td><td>REG_CUSTOM_ COLOR_MASTER_ LIGHT2_LIGHT1</td><td>Run-time colour for master lights 1 and 2
<table>
<tr><th colspan="2">H Byte: LIGHT2</th><th colspan="2">L Byte: LIGHT1</th></tr>
<tr><td>H nibble</td><td>L nibble</td><td>H nibble</td><td>L nibble</td></tr>
<tr><td>SEQUENCE</td><td>COLOR</td><td>SEQUENCE</td><td>COLOR</td></tr>
</table>
<table>
<tr><th colspan="2">SEQUENCE</th><th colspan="2">COLOR</th></tr>
<tr><td>0:</td><td>OFF</td><td>0:</td><td>OFF</td></tr>
<tr><td>1:</td><td>ON</td><td>1:</td><td>GREEN</td></tr>
<tr><td>2:</td><td>FLASH</td><td>2:</td><td>RED</td></tr>
<tr><td></td><td></td><td>3:</td><td>YELLOW</td></tr>
</table>
LOOKUP:0xFF (sequence and colour according to COLOR_TABLE)</td><td>0xFFFF</td><td></td><td>0x0</td><td>0xFFFF</td></tr>
<tr><td>58</td><td>R/W</td><td>REG_CUSTOM_ COLOR_MASTER_ LIGHT4_LIGHT3</td><td>Run-time colour for master lights 3 and 4
<table>
<tr><th colspan="2">H Byte: LIGHT4</th><th colspan="2">L Byte: LIGHT3</th></tr>
<tr><td>H nibble</td><td>L nibble</td><td>H nibble</td><td>L nibble</td></tr>
</table></td><td>0xFFFF</td><td></td><td>0x0</td><td>0xFFFF</td></tr>
</table>

<table>
<tr><th>#</th><th></th><th>Register Name</th><th>Description</th><th>Default</th><th>Unit</th><th>Min</th><th>Max</th></tr>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td>SEQUENCE</td><td>COLOR</td><td>SEQUENCE</td><td>COLOR</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td colspan="2">SEQUENCE</td><td colspan="2">COLOR</td></tr>
<tr><td>0:</td><td>OFF</td><td>0:</td><td>OFF</td></tr>
<tr><td>1:</td><td>ON</td><td>1:</td><td>GREEN</td></tr>
<tr><td>2:</td><td>FLASH</td><td>2:</td><td>RED</td></tr>
<tr><td></td><td></td><td>3:</td><td>YELLOW</td></tr>
<tr><td colspan="4">LOOKUP:0xFF (sequence and colour according to COLOR_TABLE)</td></tr>
</table>
</td><td></td><td></td><td></td><td></td></tr>
<tr><td>59</td><td>R/W</td><td>REG_CUSTOM_
COLOR_SLAVE_
LIGHT2_LIGHT1</td><td>
<table>
<tr><td colspan="4">Run-time colour for slave lights 1 and 2</td></tr>
<tr><td colspan="2">H Byte: LIGHT2</td><td colspan="2">L Byte: LIGHT1</td></tr>
<tr><td>H nibble</td><td>L nibble</td><td>H nibble</td><td>L nibble</td></tr>
<tr><td>SEQUENCE</td><td>COLOR</td><td>SEQUENCE</td><td>COLOR</td></tr>
<tr><td colspan="4"></td></tr>
<tr><td colspan="2">SEQUENCE</td><td colspan="2">COLOR</td></tr>
<tr><td>0:</td><td>OFF</td><td>0:</td><td>OFF</td></tr>
<tr><td>1:</td><td>ON</td><td>1:</td><td>GREEN</td></tr>
<tr><td>2:</td><td>FLASH</td><td>2:</td><td>RED</td></tr>
<tr><td></td><td></td><td>3:</td><td>YELLOW</td></tr>
<tr><td colspan="4">LOOKUP:0xFF (sequence and colour according to COLOR_TABLE)</td></tr>
</table>
</td><td>0xFFFF</td><td></td><td>0x0</td><td>0xFFFF</td></tr>
<tr><td>60</td><td>R/W</td><td>REG_CUSTOM_
COLOR_SLAVE_
LIGHT4_LIGHT3</td><td>
<table>
<tr><td colspan="4">Run-time colour for slave lights 3 and 4</td></tr>
<tr><td colspan="2">H Byte: LIGHT4</td><td colspan="2">L Byte: LIGHT3</td></tr>
<tr><td>H nibble</td><td>L nibble</td><td>H nibble</td><td>L nibble</td></tr>
</table>
</td><td>0xFFFF</td><td></td><td>0x0</td><td>0xFFFF</td></tr>
</table>

#		Register Name	Description	Default	Unit	Min	Max
			SEQUENCE COLOR SEQUENCE COLOR **SEQUENCE** / **COLOR** 0: OFF / 0: OFF 1: ON / 1: GREEN 2: FLASH / 2: RED 3: YELLOW LOOKUP:0xFF (sequence and colour according to *COLOR_TABLE*)				

Table 3: REG_EXT_TRANSIT_A_AUTH and REG_EXT_TRANSIT_B_AUTH

S	R/W	Single Transit. When set to 1 a single transit is started, and only when it is complete is this bit is set to zero.
M	R/W	Multiple Transit. When set to 1, increments the reader stack by one. It is automatically set to zero when stack is incremented.
CS	R/W	Cancel Single Transit. When set to 1, cancel single transit. It is automatically set to zero when the transit is cancelled
CM	R/W	Cancel Multiple Transit. When set to 1, cancel reader stack. It is automatically set to zero when the stack is cancelled

Table 4: REG_PRINTER_LED

0	OFF	
1	GREEN	Printer and Gate operation both normal
2	RED	Gate out of action
3	GREEN Flashing	Printout available for passenger (also Fire Signal Received and gate open
4	RED Flashing	Gate in Alarm or Printer problem (e.g. malfunction or out of paper)

Table 5: REG_SUMMARY

PAX_DETECTED	R/W	transit being processed; set: as the passage authorization is received; clear: when door are closed and no authorization in stack.
GATE_OK	R/W	passage acknowledge; set: when the passenger frees the last sensor of the corridor or the passage timeout elapses and the door closes. This signal is given as the door is completely closed; clear: when the host PC read the register.
PAX_CANCEL	R/W	time out elapsed without passage; set: timeout elapsed, door closed, and exit corridor free; clear: when the host PC read the register.
PAX_READY	R/W	the gate is ready to process the next authorization; set: door closed and exit corridor free; clear: as the passage authorization is received or not ready.
PAX_NOT_READY	R/W	gate is in alarm because the exit zone is occupied; set: alarm event; clear: end of the alarm cause.
PAX_TAILGATE	R/W	tail-gating attempt detected; set: the unauthorized passenger frees the last sensor of the corridor; clear: when the host PC read the register.
PAX_ENTRY_FRAUD	R/W	fraud consumed from entrance; set: the unauthorized passenger frees the last sensor of the corridor; clear: when the host PC read the register.
PAX_EXIT_FRAUD	R/W	fraud consumed from exit; set: the unauthorized passenger frees the first sensor of the corridor; clear: when the host PC read the register.
PAX_CLEAR_BTN	R/W	clear button input state
PAX_PRESENCE	R/W	Passenger presence inside the aisle; set: the passenger engages at least one sensor;

clear:	all the sensors are free.

Table 6: REG_LIGHT_COLOR_TABLE_NUMBER

0	The parameters from *PAR_LIGHT_CASE00_LIGHT2_LIGHT1* to *PAR_LIGHT_CASE31_LIGHT4_LIGHT3* controls the lights.	
1	Light1:	traffic light
	Light2:	pictogram
	Light3 and Light4:	off
2	Light1:	"display" light
	Light2:	"reader" light
	Light3 and Light4:	off
3	Light1:	traffic light
	Light2:	"reader" light
	Light3 and Light4:	off

3.3 Parameters Map

The system operation is conditioned by the values given to certain parameters stored in the permanent memory of the NEP Lite board. When the control logic microprocessor executes the resident program it consults the values of the programmable parameters and sets the timings of certain actions and internal algorithms. The parameters together with their locations and functional descriptions are listed in the following map.

The value of the parameter can by changed accessing them as registers of the Modbus or directly through the human interface of the NEP Lite board. The first two columns of the following table are the location indexes to access each parameter respectively through the Modbus or the NEP Lite interface.

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
61	0	*PAR_CENTER_POSITION_MS*	Encoder value when master door is closed	*508*	*pt*	*450*	*650*
62	1	*PAR_CENTER_POSITION_SL*	Encoder value when master slave is closed	*508*	*pt*	*450*	*650*
63	2	*PAR_OPENA_POSITION_MS*	Encoder value when master door is open in direction A	*308*	*pt*	*250*	*350*
64	3	*PAR_OPENA_POSITION_SL*	Encoder value when slave door is open in direction A	*308*	*pt*	*250*	*350*
65	4	*PAR_OPENB_POSITION_MS*	Encoder value when master door is open in direction B	*708*	*pt*	*650*	*750*
66	5	*PAR_OPENB_POSITION_SL*	Encoder value when slave door is open in direction B	*708*	*pt*	*650*	*750*
67	6	*PAR_MECHANISM_TYPE*	Mechanism selection (see Table 11)	*CONFIG_PAR_MECHANISM_TYPE (Table 7)*		*0*	*7*
68	7	*PAR_CLOSE_SPEED*	Close speed (from slowest '0' to fastest '5')	*3*		*0*	*5*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
69	8	*PAR_OPEN_SPEED*	Open speed (from slowest '0' to fastest '5')	*3*		*0*	*5*
70	9	*PAR_CLOSE_OFFSET*	Close offset	*1*		*0*	*9*
71	A	*PAR_OPEN_OFFSET*	Open Offset	*1*		*0*	*9*
72	B	*PAR_FIRE_ALARM_ENABLE_NO*	Enables the Normally Open condition • 0: disabled • 1: enabled	*1*		*0*	*1*
73	C	*PAR_FIRE_ALARM_PRIORITY*	Which direction to open toward to, in case of emergency: • 0: do not open • 1: toward direction A • 2: toward direction B NB: as to minimize opening time, doors may open toward the opposite direction.	*(Table 7)*		*0*	*2*
74	D		Not used				
75	E	*PAR_FIRE_ALARM_ENABLE_BUZZER*	Enable/ Disable the buzzer	*0*		*0*	*1*
76	F	*PAR_FIRE_ALARM_ENABLE_POWERDOWN*	Enable the power down event to switch off the battery	*0*		*0*	*1*
77	10	*PAR_FIRE_ALARM_EXIT_TIMER*	Delay timer to exit from the fire alarm status	*320*	*10ms*	*0*	*65535*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
78	11	*PAR_GATE_FUNC*	Gate type: • *GATE_FUNC_UNKNOWN* Gate type unknown • *GATE_FUNC_AIRPORT* Airport Flap Lane • *GATE_FUNC_MASSTRANSIT* Mass Transit	*CONFIG_PAR_GATE_FUNC (Table 7)*		*0*	*2*
79	12		Not used				
80	13	*PAR_AISLE_MODE*	Aisle Mode: • 0: normally closed • 1: normally open	*0*		*0*	*1*
81	14	*PAR_MODE_A*	Current Transit Mode in A direction: • 1: lock mode • 2: free mode • 3: controlled mode	*3*		*0*	*3*
82	15	*PAR_MODE_B*	Current Transit Mode in B direction • 1: lock mode • 2: free mode • 3: controlled mode	*1*		*0*	*3*
83	16	*PAR_MP2000*	Enable / Disable the MP2000 device	*0*		*0*	*1*
84	17	*PAR_LOCAL_MODE*	Enable the management of the Serial Line communication: • 0: keep current modes REG_MODE_A and REG_MODE_B; • 1: switch to modes PAR_MODE_A and PAR_MODE_B; • 2: apply modes from PAR_LOCAL_MODE_A and PAR_LOCAL_MODE_B	*0*		*0*	*2*
85	18	*PAR_BUZZER_ENABLE*	Enable/Disable the use of the buzzer	*0*		*0*	*1*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
86	19	*PAR_BUZZER_ SHUT_TIMER*	Shut the buzzer after a timeout, even if the cause of the alarm is still present	*300*	*10ms*	*0*	*65535*
87	1A		Not used				
88	1B		Not used				
89	1C	*PAR_PASSAGE_ CANC_TIMEOUT*	Maximum time to pass through the gate after validation	*CONFIG_PAR_ PASSAGE_CANC_ TIMEOUT (Table 7)*	*10ms*	*0*	*65535*
90	1D	*PAR_CTRL_ CLOSE_DELAY*	Close delay time in controlled mode	*0*	*10ms*	*0*	*65535*
91	1E	*PAR_FREE_ CLOSE_DELAY*	Close delay time in free mode	*120*	*10ms*	*0*	*65535*
92	1F	*PAR_COUNTING_ PULSE*	Counting pulse width	*5*	*16ms*	*0*	*65535*
93	20	*PAR_COUNTING_ SENSITIVITY*	Sensitivity of counting algorithm: • 0: low • 1: high	*0*		*0*	*1*
94	21		Not used				
95	22	*PAR_VALIDATION_ IN_FRAUD*	Enable/Disable reader during fraud condition	*1*		*0*	*1*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
96	23	*PAR_VALIDATION_SENSORS*	Specify the zone where Validation is NOT Allowed: • 0: Zone disabled • 1: S1 | S2 | S3 | S4 | S5 | S6 | S7 • 2: S2 | S3 | S4 | S5 | S6 | S7 • 3: S3 | S4 | S5 | S6 | S7 • 4: S4 | S5 | S6 | S7 • 5: S5 | S6 | S7 • 6: S6 | S7 • 7: S7	*0*		*0*	*7*
97	24	*PAR_VALIDATION_RESET_ALARM*		*1*		*0*	*1*
98	25	*PAR_INTRUSION_TRANSIT_TIMEOUT*	Timeout for a long transit when in controlled mode	*1500*	*10ms*	*0*	*65535*
99	26	*PAR_INTRUSION_ZONE_1_TIMEOUT*	Validation Zone 1 Timeout	*500*	*10ms*	*0*	*65535*
100	27	*PAR_INTRUSION_ZONE_2_TIMEOUT*	Validation Zone 2 Timeout	*30*	*10ms*	*0*	*65535*
101	28	*PAR_WRONG_WAY_TIMEOUT*	Timeout for the exit sensors in the wrong way alarm	*20*	*10ms*	*0*	*65535*
102	29	*PAR_FRAUD_EXTEND_TIMEOUT*	Extending Time after a fraud alarm with acoustic indication	*500*	*10ms*	*0*	*65535*
103	2A	*PAR_WRONG_WAY_EXTEND_TIMEOUT*	Extending Time after a wrong way alarm with acoustic indication	*100*	*10ms*	*0*	*65535*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
104	2B	*PAR_TAILGATING_ ENABLE*	Enable tailgating algorithms: • 0: no tailgating detection • 1: enable narrow algorithm • 2: enable far algorithm • 3: enable both algorithm	*3*		*0*	*3*
105	2C	*PAR_TAILGATING_ SENSITIVITY*	Select the sensitivity for narrow and/or far tailgating algorithm: • 0: low • 1: medium • 2: high	*0*		*0*	*2*
106	2D	*PAR_TAILGATING_ EXTEND_TIMEOUT*	Extending Time after a tail alarm with acoustic indication	*500*	*10ms*	*0*	*65535*
107	2E	*PAR_MASTER_ USER_INPUT_1_CFG*	Function assigned to the User Input 1: • 0: reader A in standard mode • 1: reader A in ADA mode • 2: reader A in level mode • 3: reader B in standard mode • 4: reader B in ADA mode • 5: reader B in level mode • 6: hold-open active • 7: hold-close active • 8: fire alarm active • 9: clear button active	*0*		*0*	*9*
108	2F	PAR_MASTER_ USER_INPUT_2_CFG	Function assigned to the User Input 2 (see *PAR_MASTER_USER_INPUT_1_CFG*)	3		0	9

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
109	30	PAR_MASTER_ USER_INPUT_3_CFG	Function assigned to the User Input 3 (see *PAR_MASTER_USER_INPUT_1_CFG*)	6		0	9
110	31	PAR_MASTER_ USER_INPUT_4_CFG	Function assigned to the User Input 4 (see *PAR_MASTER_USER_INPUT_1_CFG*)	7		0	9
111	32	PAR_MASTER_ USER_INPUT_5_CFG	Function assigned to the User Input 5 (see *PAR_MASTER_USER_INPUT_1_CFG*)	8		0	9
112	33	*PAR_MASTER_ USER_INPUT_6_CFG*	Function assigned to the User Input 6 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*9*		*0*	*9*
113	34	*PAR_SLAVE_ USER_INPUT_1_CFG*	Function assigned to the User Input 1 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*3*		*0*	*9*
114	35	*PAR_SLAVE_ USER_INPUT_2_CFG*	Function assigned to the User Input 2 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*3*		*0*	*9*
115	36	*PAR_SLAVE_ USER_INPUT_3_CFG*	Function assigned to the User Input 3 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*6*		*0*	*9*
116	37	*PAR_SLAVE_ USER_INPUT_4_CFG*	Function assigned to the User Input 4 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*7*		*0*	*9*
117	38	*PAR_SLAVE_ USER_INPUT_5_CFG*	Function assigned to the User Input 5 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*8*		*0*	*9*
118	39	*PAR_SLAVE_ USER_INPUT_6_CFG*	Function assigned to the User Input 6 (see *PAR_MASTER_USER_INPUT_1_CFG*)	*9*		*0*	*9*
119	3A	*PAR_MASTER_ USER_INPUT_ 1_FILTER*	Filter Input 1	*4*	*16ms*	*1*	*255*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
120	3B	*PAR_MASTER_USER_INPUT_2_FILTER*	Filter Input 2	*4*	*16ms*	*1*	*255*
121	3C	*PAR_MASTER_USER_INPUT_3_FILTER*	Filter Input 3	*4*	*16ms*	*1*	*255*
122	3D	*PAR_MASTER_USER_INPUT_4_FILTER*	Filter Input 4	*4*	*16ms*	*1*	*255*
123	3E	*PAR_MASTER_USER_INPUT_5_FILTER*	Filter Input 5	*4*	*16ms*	*1*	*255*
124	3F	*PAR_MASTER_USER_INPUT_6_FILTER*	Filter Input 6	*4*	*16ms*	*1*	*255*
125	40	*PAR_SLAVE_USER_INPUT_1_FILTER*	Filter Input 1	*4*	*16ms*	*1*	*255*
126	41	*PAR_SLAVE_USER_INPUT_2_FILTER*	Filter Input 2	*4*	*16ms*	*1*	*255*
127	42	*PAR_SLAVE_USER_INPUT_3_FILTER*	Filter Input 3	*4*	*16ms*	*1*	*255*
128	43	*PAR_SLAVE_USER_INPUT_4_FILTER*	Filter Input 4	*4*	*16ms*	*1*	*255*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
129	44	*PAR_SLAVE_USER_INPUT_5_FILTER*	Filter Input 5	*4*	*16ms*	*1*	*255*
130	45	*PAR_SLAVE_USER_INPUT_6_FILTER*	Filter Input 6	*4*	*16ms*	*1*	*255*
131	46	*PAR_RELAY_MASTER_K1*	Flag linked to the relay K1 Master side: • 0x0001: reader A enable/disable • 0x0002: reader B enable/disable • 0x0004: counting pulse in direction A • 0x0008: counting pulse in direction B • 0x0010: reader A acknowledge • 0x0020: reader B acknowledge • 0x0040: priority in direction A • 0x0080: priority in direction B • 0x0100: doors are closed • 0x0200: doors are open • 0x0400: fraud alarm • 0x0800: out of service • 0x1000: fire alarm echo • 0x2000: power fail alarm	*0x0400*		*0x0*	*0xffff*
132	47	*PAR_RELAY_MASTER_K2*	Flag linked to the relay K2 Master side (see *PAR_RELAY_MASTER_K1*)	*0x0004*		*0x0*	*0xFFFF*
133	48	*PAR_RELAY_MASTER_K3*	Flag linked to the relay K3 Master side (see *PAR_RELAY_MASTER_K1*)	*0x0010*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
134	49	*PAR_RELAY_MASTER_K4*	Flag linked to the relay K4 Master side (see *PAR_RELAY_MASTER_K1*)	*0x0040*		*0x0*	*0xFFFF*
135	4A	*PAR_RELAY_SLAVE_K1*	Flag linked to the relay K1 Slave side (see *PAR_RELAY_MASTER_K1*)	*0x1000*		*0x0*	*0xFFFF*
136	4B	*PAR_RELAY_SLAVE_K2*	Flag linked to the relay K2 Slave side (see *PAR_RELAY_MASTER_K1*)	*0x0008*		*0x0*	*0xFFFF*
137	4C	*PAR_RELAY_SLAVE_K3*	Flag linked to the relay K3 Slave side (see *PAR_RELAY_MASTER_K1*)	*0x0020*		*0x0*	*0xFFFF*
138	4D	*PAR_RELAY_SLAVE_K4*	Flag linked to the relay K4 Slave side (see *PAR_RELAY_MASTER_K1*)	*0x0080*		*0x0*	*0xFFFF*
139	4E	*PAR_READER_A_MAX_STACK*	Max number of stack validation in reader A	*(Table 7)*		*0*	*10*
140	4F	*PAR_READER_B_MAX_STACK*	Max number of stack validation in reader B	*(Table 7)*		*0*	*10*
141	50		Not used				
142	51	*PAR_PITTO_ACK_TIME*	Sets the timeout for pictogram acknowledge	*320*	*10ms*	*0*	*65535*
143	52	*PAR_READER_EXT_ACK_TIME*	Sets the pulse time for the external validation acknowledge	*65*	*10ms*	*0*	*65535*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
144	53	*PAR_SAFETY_OPEN_SPEED*	Open Speed in Safety operations	*1*		*0*	*5*
145	54	*PAR_SAFETY_CLOSE_SPEED*	Close Speed in Safety Operations	*1*		*0*	*5*
146	55	*PAR_SAFETY_SENSORS_STD*	This is the safety sensors pattern for standard passage: • 0x0001: S1A • 0x0002: S2A • 0x0004: S3A • 0x0008: S4A • 0x0010: S5A • 0x0020: S6A • 0x0040: S7A • 0x0080: S8A • 0x8000: S8B • 0x4000: S7B • 0x2000: S6B • 0x1000: S5B • 0x0800: S4B • 0x0400: S3B • 0x0200: S2B • 0x0100: S1B	*0x0*		*0x0*	*0xFFFF*
147	56	*PAR_SAFETY_SENSORS_ADA*	This is the safety sensors pattern for ADA passage (see *PAR_SAFETY_SENSORS_STD*)	*0x5BDA*		*0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
148	57	*PAR_SAFETY_ DELAY_CLOSE*	Delay time before restart to close after safety	*1*	*16ms*	*0*	*65535*
149	58	*PAR_OBSTACLE_ TIMEOUT*	Delay time before an obstacle detection before to restart	*300*	*10ms*	*0*	*65535*
150	59	*PAR_POWERDOWN_ DOOR_STATUS*	Final status during power down: • 0: close • 1: open A • 2: open B • 3: stop • 4: disabled	*1*		*0*	*4*
151	5A	*PAR_POWERDOWN_ TIMEOUT*	Timeout waiting the final status	*1000*	*10ms*	*0*	*65535*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
152	5B	*PAR_FIX_SENSOR_FAULT_ENABLE*	Enable / Disable the diagnostic on the gate's sub-systems • 0: - • 1: enable fraud detection • 2: enable wrong way detection • 3: - • 4: - • 5: - • 6: - • 7: - • 8: enable encoder faults • 9: enable safety device faults • 10: enable relay faults • 11: enable sensor faults • 12: enable motor faults • 13: - • 14: - • 15: -	*(Table 7)*		*0*	*65535*
153	5C	*PAR_BATTERY_FAUL_ENABLE*	Enable the Battery Fault	*1*		*0*	*1*
154	5D	*PAR_DRIVER_FAULT_ENABLE*	Enable the Driver Fault	*1*		*0*	*1*
155	5E	*PAR_FAULT_ENABLE_BUZZER*	Enable/Disable the Buzzer during a fault condition	*1*		*0*	*1*
156	5F	*PAR_FAULT_DOOR_STATUS*	Status of the door when in fault (see *PAR_POWERDOWN_DOOR_STATUS*)	*4*		*0*	*4*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
157	60	*PAR_FAULT_OBSCURED_SENSOR_TIMEOUT*	Time of continuous obscuration of a faulty sensor	*30*	*1000ms*	*10*	*65535*
158	61	*PAR_RS232_PROTOCOL*	Select which protocol to use on RS232 port: • 0: none • 1: Italdis protocol • 2: Modbus protocol	*(Table 7)*		*0*	*2*
159	62	*PAR_RS232_BAUD_RATE*	Select the baud rate for RS232 port: • 0: 0 Bd • 1: 1200 Bd • 2: 2400 Bd • 3: 4800 Bd • 4: 9600 Bd • 5: 19200 Bd • 6: 38400 Bd • 7: 57600 Bd • 8: 115200 Bd • 9: 230400 Bd	*(Table 7)*		*0*	*9*
160	63	*PAR_RS485_PROTOCOL*	Select which protocol to use on RS485 port	*(Table 7)*		*0*	*2*
161	64	*PAR_RS485_BAUD_RATE*	Select the baud rate for RS485 port (see *PAR_RS232_BAUD_RATE*)	*(Table 7)*		*0*	*9*
162	65	*PAR_MODBUS_ID*	Board identifier on Modbus network	*1*		*1*	*32*
163	66	*PAR_MODBUS_TIMEOUT*	Modbus timeout connection	*1*	*1000ms*	*1*	*256*
164	67	*PAR_MODBUS_EOF*	End of frame time	*2*	*10ms*	*1*	*256*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
165	68	*PAR_MODBUS_TDA*	Delay answer	*1*	*10ms*	*1*	*256*
166	69	*PAR_MODBUS_ MODIFY_ENABLE*	Enable / Disable the registers modification	*1*		*0*	*1*
167	6A	*PAR_MODBUS_ PSW_ENABLE*	Enable / Disable the use of a password to connect	*0*		*0*	*1*
168	6B	*PAR_MODBUS_ PSW_ATTEMP*	Max number of attempts when the password is enabled	*20*		*4*	*65535*
169	6C	*PAR_MODBUS_ PSW_0*	Modbus code 0	*0*		*0*	*65535*
170	6D	*PAR_MODBUS_ PSW_1*	Modbus code 1	*1*		*0*	*65535*
171	6E	*PAR_MODBUS_ PSW_2*	Modbus code 2	*2*		*0*	*65535*
172	6F	*PAR_MODBUS_ PSW_3*	Modbus code 3	*3*		*0*	*65535*
173	70	*PAR_ITALDIS_ID*	Address of the Italdis device	*1*		*1*	*32*
174	71	*PAR_ITALDIS_ POLLING_TIMEOUT*	Italdis polling timeout	*70*	*100ms*	*0*	*256*
175	72	*PAR_ITALDIS_ ANSWER_TIMEOUT*	Italdis answer timeout	*4*	*10ms*	*0*	*256*
176	73		Not used				
177	74		Not used				
178	75		Not used				
179	76		Not used				
180	77		Not used				

Modbus offset	NEP Lite interface	Parameter name	Description	Default	Unit	Min	Max
181	78		Not used				
182	79		Not used				
183	7A		Not used				
184	7B		Not used				
185	7C		Not used				
186	7D		Not used				
187	7E		Not used				
188	7F	*PAR_LIGHT_ COLOR_TABLE_ NUMBER*	Color table for light management	*(Table 7)*		*0*	*3*
189	80	*PAR_LIGHT_ CASE00_ LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #0 (see Table 10) (see *REG_CUSTOM_COLOR_MASTER_LIGHT2_LIGHT1*)	*0x0*		*0x0*	*0xFFFF*
190	81	*PAR_LIGHT_ CASE00_ LIGHT4_LIGHT3*	Color and sequence for lights #4 and #3 in case #0 (see Table 10) (see *REG_CUSTOM_COLOR_MASTER_LIGHT4_LIGHT3*)	*0x0*		*0x0*	*0xFFFF*
191	82	*PAR_LIGHT_ CASE01_ LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #1 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
192	83	*PAR_LIGHT_ CASE01_ LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #1 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
193	84	*PAR_LIGHT_ CASE02_ LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #2 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
194	85	*PAR_LIGHT_CASE02_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #2 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
195	86	*PAR_LIGHT_CASE03_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #3 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
196	87	*PAR_LIGHT_CASE03_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #3 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
197	88	*PAR_LIGHT_CASE04_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #4 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
198	89	*PAR_LIGHT_CASE04_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #4 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
199	8A	*PAR_LIGHT_CASE05_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #5 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
200	8B	*PAR_LIGHT_CASE05_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #5 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
201	8C	*PAR_LIGHT_CASE06_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #6 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
202	8D	*PAR_LIGHT_CASE06_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #6 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
203	8E	*PAR_LIGHT_CASE07_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #7 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
204	8F	*PAR_LIGHT_CASE07_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #7 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
205	90	*PAR_LIGHT_CASE08_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #8 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
206	91	*PAR_LIGHT_CASE08_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #8 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
207	92	*PAR_LIGHT_CASE09_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #9 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
208	93	*PAR_LIGHT_CASE09_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #9 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
209	94	*PAR_LIGHT_CASE10_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #10 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
210	95	*PAR_LIGHT_CASE10_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #10 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
211	96	*PAR_LIGHT_CASE11_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #11 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
212	97	*PAR_LIGHT_CASE11_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #11 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
213	98	*PAR_LIGHT_CASE12_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #12 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
214	99	*PAR_LIGHT_CASE12_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #12 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
215	9A	*PAR_LIGHT_CASE13_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #13 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
216	9B	*PAR_LIGHT_CASE13_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #13 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
217	9C	*PAR_LIGHT_CASE14_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #14 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
218	9D	*PAR_LIGHT_CASE14_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #14 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
219	9E	*PAR_LIGHT_CASE15_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #15 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
220	9F	*PAR_LIGHT_CASE15_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #15 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
221	A0	*PAR_LIGHT_CASE16_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #16 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
222	A1	*PAR_LIGHT_CASE16_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #16 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
223	A2	*PAR_LIGHT_CASE17_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #17 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
224	A3	*PAR_LIGHT_CASE17_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #17 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
225	A4	*PAR_LIGHT_CASE18_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #18 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
226	A5	*PAR_LIGHT_CASE18_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #18 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
227	A6	*PAR_LIGHT_CASE19_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #19 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
228	A7	*PAR_LIGHT_CASE19_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #19 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
229	A8	*PAR_LIGHT_CASE20_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #20 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
230	A9	*PAR_LIGHT_CASE20_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #20 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
231	AA	*PAR_LIGHT_CASE21_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #21 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
232	AB	*PAR_LIGHT_CASE21_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #21 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
233	AC	*PAR_LIGHT_CASE22_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #22 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
234	AD	*PAR_LIGHT_CASE22_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #22 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
235	AE	*PAR_LIGHT_CASE23_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #23 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
236	AF	*PAR_LIGHT_CASE23_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #23 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
237	B0	*PAR_LIGHT_CASE24_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #24 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
238	B1	*PAR_LIGHT_CASE24_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #24 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
239	B2	*PAR_LIGHT_CASE25_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #25 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
240	B3	*PAR_LIGHT_CASE25_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #25 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
241	B4	*PAR_LIGHT_CASE26_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #26 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
242	B5	*PAR_LIGHT_CASE26_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #26 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
243	B6	*PAR_LIGHT_CASE27_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #27 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Modbus offset	NEP Lite interface	*Parameter name*	*Description*	*Default*	*Unit*	*Min*	*Max*
244	B7	*PAR_LIGHT_CASE27_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #27 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
245	B8	*PAR_LIGHT_CASE28_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #28 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
246	B9	*PAR_LIGHT_CASE28_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #28 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
247	BA	*PAR_LIGHT_CASE29_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #29 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
248	BB	*PAR_LIGHT_CASE29_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #29 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
249	BC	*PAR_LIGHT_CASE30_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #30 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
250	BD	*PAR_LIGHT_CASE30_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #30 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
251	BE	*PAR_LIGHT_CASE31_LIGHT2_LIGHT1*	Color and sequence for lights #1 and #2 in case #31 (see Table 10)	*0x0*		*0x0*	*0xFFFF*
252	BF	*PAR_LIGHT_CASE31_LIGHT4_LIGHT3*	Color and sequence for lights #3 and #4 in case #31 (see Table 10)	*0x0*		*0x0*	*0xFFFF*

Table 7: Default parameter values for different Firmwares

	FW0051A	*FW0052A*	*FW0054A*	*FW0055A*				
PAR_FIRE_ALARM_PRIORITY	*2*	*1*						
CONFIG_PAR_MECHANISM_TYPE	*3*	*3*	*1*	*1*				
CONFIG_PAR_GATE_FUNC	*1*	*1*	*2*	*2*				
PAR_FIX_SENSOR_FAULT_ENABLE				*7942*				
CONFIG_PAR_PASSAGE_CANC_TIMEOUT	*500*	*1500*	*625*	*625*				
PAR_PASSAGE_CANC_TIMEOUT	*300*	*938*						
PAR_READER_A_MAX_STACK	*1*	*4*						
PAR_READER_B_MAX_STACK	*1*	*4*						
PAR_RS232_PROTOCOL	*2*	*1*						
PAR_RS232_BAUD_RATE	*5*	*4*						
PAR_RS485_PROTOCOL	*1*	*0*						
PAR_RS485_BAUD_RATE	*4*	*0*						
PAR_LIGHT_COLOR_TABLE_NUMBER	*3*	*2*						

Table 8: Fault Registers

When faults occur, error codes are reported in a set of 4 read-only registers named *REG_FAULT_x* (see REG_FAULT_1, REG_FAULT_2, REG_FAULT_3 or REG_FAULT_4)

Each register is a 16-bit word which contains the following information:

high byte high nibble	high byte low nibble	low byte
addr	cat	code

Where:

- **addr** is the board address which originates the fault;
- **cat** is a single letter which represents the fault category;
- **code** is a number from 0 to 255 which is specific to the fault category (see *Table* 9).

cat	*description*
1	*Sensors*
2	*Motor driver*
3	*Encoder*
4	*Relays*
5	*Safety device*

Table 9: Fault Codes

Dec code	***Hex code***	***Description***
"dec code" shown on the display.		
		Motor Driver Fault Codes (d)
1	*1*	Mechanism *PAR_MECHANISM_TYPE* is out of range
2	*2*	Motor driver cannot be initialized
3	*3*	Encoder cannot initialized
4	*4*	Encoder module fails
		Encoder (e)
1	*01*	Channel #1 too low **or** Position below min
2	*02*	Channel #1 too high **or** Position above max
4	*04*	Encoder channel cannot be initialized
8	*08*	Channel difference too low
9	*09*	Channel #1 too low and channel difference too low
10	*0A*	Channel #1 too high and channel difference too low
16	*10*	Channel #2 too low
17	*11*	Channels #1 and #2 too low
18	*12*	Channel #1 too high and channel #2 too low
24	*18*	Channel difference too low and channel #2 too low
32	*20*	Channel #2 too high

33	*21*	Channel #1 too low and channel #2 too high
34	*22*	Channels #1 and #2 too high
40	*28*	Channel difference too low and channel #2 too high
41	*29*	Channel #1 too low, channel #2 too high, and channel difference too low
42	*2A*	Channels #1 and #2 too high and channel difference too low
128	*80*	Channel difference too high
129	*81*	Channel #1 too low and channel difference too high
130	*82*	Channel #1 too high and channel difference too high
144	*90*	Channel #2 too low and channel difference too high
145	*91*	Channels #1 and #2 too low and channel difference too high
146	*92*	Channel #1 too high, channel #2 too low, and channel difference too high
160	*A0*	Channel #2 too high and channel difference too high
161	*A1*	Channel #1 too low, channel #2 too high, and channel difference too high
162	*A2*	Channels #1 and #2 too high and channel difference too high
		NOTE: *Channel #1* is connected to SD's pin J5.2 *Channel #2* is connected to SD's pin J5.5
Safety Device (h)		
17	*11*	CAN: faults from underlying hardware
18	*12*	CAN: protocol version mismatch
20	*14*	CAN: parameter mismatch
24	*18*	CAN: peer not alive
33	*21*	Door: invalid door type

34	*22*	Door: wrong door limits
35	*23*	Door: door position is out-of-range
65	*41*	Flash: CRC mismatch
81	*51*	Parameter: wrong JP1/JP2 setting or EEPROM corrupted
97	*61*	RAM: the two RAM copies don't match
98	*62*	RAM: location lock timed out
Sensor (p)		
1	*1*	At least one passage sensor failed test #1
32	*20*	Passage sensor #1 failed test #2
33	*21*	Passage sensor #2 failed test #2
34	*22*	Passage sensor #3 failed test #2
35	*23*	Passage sensor #4 failed test #2
36	*24*	Passage sensor #5 failed test #2
37	*25*	passage sensor #6 failed test #2
38	*26*	passage sensor #7 failed test #2
39	*27*	passage sensor #8 failed test #2
40	*28*	passage sensor #9 failed test #2
41	*29*	passage sensor #10 failed test #2
42	*2A*	passage sensor #11 failed test #2
43	*2B*	passage sensor #12 failed test #2
44	*2C*	passage sensor #13 failed test #2
45	*2D*	passage sensor #14 failed test #2
46	*2E*	passage sensor #15 failed test #2

47	*2F*	passage sensor #16 failed test #2
48	*30*	passage sensor #17 failed test #2
49	*31*	passage sensor #18 failed test #2
50	*32*	passage sensor #19 failed test #2
51	*33*	passage sensor #20 failed test #2
52	*34*	passage sensor #21 failed test #2
53	*35*	passage sensor #22 failed test #2
54	*36*	passage sensor #23 failed test #2
55	*37*	passage sensor #24 failed test #2
56	*38*	passage sensor #25 failed test #2
57	*39*	passage sensor #26 failed test #2
58	*3A*	passage sensor #27 failed test #2
59	*3B*	passage sensor #28 failed test #2
60	*3C*	passage sensor #29 failed test #2
61	*3D*	passage sensor #30 failed test #2
62	*3E*	passage sensor #31 failed test #2
63	*3F*	passage sensor #32 failed test #2
		NOTE: *Test #1* a passage sensor is faulty if it does not pass SD tests *Test #2* a passage sensor is faulty when it has been obscured for more than PAR_FAULT_OBSCURED_SENSOR_TIMEOUT.
Relays (r)		
1	*01*	Relay #1 is not close
2	*202*	Relay #1 is not open
16	*10*	Relay #2 is not close

17	*11*	Relays #1 and #2 are not close
32	*20*	Relay #2 is not open
34	*22*	Relays #1 and #2 are not open
		NOTE: *relay #1* is connected to SD's pin J11.1 *relay #2* is connected to SD's pin J11.4.

Table 10: Color Table

The customizable colour table is a set of 64 parameters *named PAR_LIGHT_CASExx_LIGHT2_LIGHT1* and *PAR_LIGHT_CASExx_LIGHT4_LIGHT3*, where *xx* is a number from 00 to 31.

Each these parameters allows to specify the colours of lights #1 to #4 when the gate is operating in one of the 32 defined cases that are the output explained in the following table.

For each direction ***dir*** A and B, a case is chosen when all its input conditions are satisfied at the same time.

Output	**Inputs**					**Description**
PAR_LIGHT_CASE#	**REG_RUNNING_MODE**	**REG_MODE_A/B**	**REG_PRIORITY**	**REG_FRAUD**	**Reader A/B**	
00	(1)	(1)	(1)	(1)	(1)	Fall-back case when no other case matches.
01	*FAULT_MODE*	(1)	(1)	(1)	(1)	The gate is in fault mode.
02	*FIRE_ALARM_MODE*	(1)	not equal to *dir*	(1)	(1)	The gate is in fire alarm mode, *dir* has no precedence.
03	*FIRE_ALARM_MODE*	(1)	equal to *dir*	(1)	(1)	The gate is in fire alarm mode, *dir* has precedence.
04	*HOLD_CLOSE_MODE*	(1)	(1)	(1)	(1)	Hold close mode.
05	*HOLD_OPEN_MODE*	(1)	(1)	(1)	(1)	Hold open mode.
06	*POWERDOWN*	(1)	(1)	(1)	(1)	The gate I in power down mode.
07	*TRANSIT_MODE*	(1)	(1)	any	(1)	The gate is signalling a fraud.

08	*TRANSIT_MODE*	*CONTROLLED_MODE*	not equal to *dir*	*FRAUD_NONE*	(1)	No fraud in progress and *dir* is in controlled mode and the gate is not engaged in this direction.
09	*TRANSIT_MODE*	*CONTROLLED_MODE*	equal to *dir*	*FRAUD_NONE*	ack	No fraud in progress and *dir* is in controlled mode and the gate is engaged in this direction and is acknowledging the received authorization.
10	*TRANSIT_MODE*	*CONTROLLED_MODE*	equal to *dir*	*FRAUD_NONE*	empty	No fraud in progress and *dir* is in controlled mode and the gate is engaged in this direction and the authorization stack is empty.
11	*TRANSIT_MODE*	*CONTROLLED_MODE*	equal to *dir*	*FRAUD_NONE*	full	No fraud in progress and *dir* is in controlled mode and the gate is engaged in this direction and the authorization stack is full.
12	*TRANSIT_MODE*	*CONTROLLED_MODE*	equal to *dir*	*FRAUD_NONE*	partial	No fraud in progress and *dir* is in controlled mode and the gate is engaged in this direction and the authorization stack is neither empty nor full.
13	*TRANSIT_MODE*	*FREE_MODE*	(1)	*FRAUD_NONE*	(1)	No fraud in progress and *dir* is in free mode.
14	*TRANSIT_MODE*	*LOCKED_MODE*	(1)	*FRAUD_NONE*	(1)	No fraud in progress and *dir* is in blocked mode.

Note: (1) don’t care

Table 11: Mechanism Types

value	Mnemonic description	Gate width	Paddle type	Paddle height	Motor driver
0	GATE_MECHANISM_UNKNOWN	unknown	unknown	unknown	Unknown
1	GATE_MECHANISM_HBRIDGE_POLY_NARROW_LOW	narrow	polycarbonate	low	H-bridge
2	GATE_MECHANISM_OMRON_GLASS_NARROW_LOW	narrow	glass	low	Omron
3	GATE_MECHANISM_OMRON_GLASS_NARROW_MID	narrow	glass	mid	Omron
4	GATE_MECHANISM_OMRON_GLASS_NARROW_HI	narrow	glass	high	Omron
5	GATE_MECHANISM_OMRON_GLASS_WIDE_LOW	wide	glass	low	Omron
6	GATE_MECHANISM_OMRON_GLASS_WIDE_MID	wide	glass	mid	Omron
7	GATE_MECHANISM_OMRON_GLASS_WIDE_HI	wide	glass	high	Omron

THIS PAGE INTENTIONALLY LEFT BLANK