

683-52 RS-485 OEM Proximity Reader

Data Sheet

Overview

The 683-52 RS-485 OEM proximity reader consists of three parts: a potted unit containing the electronics, a front cover, and an optional spacer plate. A fixed 10 way colour-coded cable protrudes from the back of the potted unit.

The reader reads the code from an RFID transponder, and stores it until polled via a simple, host-controlled RS-485 polling protocol. These poll packets simultaneously control the functions of the three LEDs and beeper.

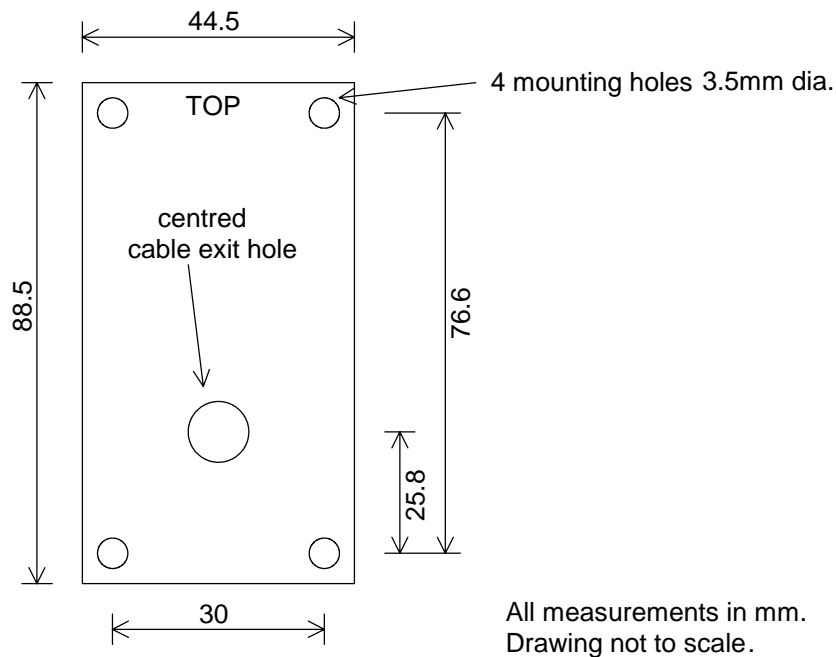
The reader also allows user control of two Open Collector Transistor (OCT) outputs and sensing of two active-low inputs.



Specifications

- Power requirements: 5.0-13.6V dc. Current consumption is 75 mA typical
- RF Frequency: 125 kHz.
- 40 bit read only transponders supported: EM4001 family, TEMIC e5550 and equivalent devices.
- Output formats supported: RS-485
- Typical reading range with supply voltage in range 5.5V-13.6V: keyring tag with 20mm coil - 85mm, ISO card with 50mm coil - 165mm.
- 3 LEDs (RED, YELLOW, GREEN), controlled by network host.
- 2 Open Collector Transistor Outputs, controlled by network host.
Specifications: $V_{CEmax} = 30V$, $I_{CEmax} = 90mA$, $P_{tot, 25^{\circ}C} = 350mW$
- 2 Active-low Inputs, sensed by network host.
Specifications: Internal 10K pull-up to +5V
- Sounder emits at 4 kHz, controlled by network host.
- Operating temperature range: $-20^{\circ}C$ - $+60^{\circ}C$.
- 10 way cable: 1m long
- Weight: 90 grams.
- Dimensions: reader 89 x 45 x 16 mm, optional spacer plate 89 x 45 x 7 mm

Physical Dimensions and Mounting Details



If the spacer plate is used the reader cable may be brought out of one of four exit points on the spacer: top, bottom, left or right. This enables the cable to be run on the surface of the wall. If no spacer plate is used a minimum hole size of 6.5mm must be drilled in the wall at the cable exit position as shown above to allow the cable to exit perpendicular to the reader.

The optional spacer plate may also be used when mounting the reader on a metal surface to reduce the negative effects of metal on the read range.

Connections

The table below details the function of each wire:

Colour	Name	Function
GREY	PROGRAM	If held low at startup, Address Programming will commence. See section on Address Programming.
WHITE	OUTPUT 1	Open Collector Output, controlled by a the host
BROWN	OUTPUT 2	Open Collector Output, controlled by a the host
GREEN	INPUT 1	Active low, passively pulled to +5V.
YELLOW	INPUT 2	Active low, passively pulled to +5V.
ORANGE	RS-485 +	RS-485 + line
BLUE	RS-485 -	RS-485 - line
PURPLE	TERMINATE	To terminate the network, link this to RS-485 + (ORANGE) See RS-485 Connections for more information.
RED	+VDC	Connect +5V - +13.6V from power supply.
BLACK	0V	Connect 0V from power supply.

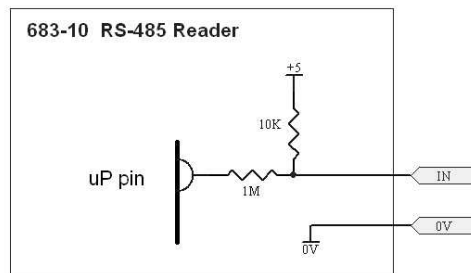
Note: INPUTS 1 & 2 are active low. The input is internally pulled high and may be pulled low by an open collector transistor or driven low by the output of a 5V CMOS or TTL gate.

Power Connections

The reader has an internal low dropout 5V regulator and so for maximum performance the input voltage must be smooth DC between 5.5V and 13.6V. The reading distance is unchanged for input voltages between 5.5V and 13.6V. For input voltages below 5.5V the read range drops off by about 20%. If 5V is supplied to the reader this should be noise-free to achieve maximum possible read ranges.

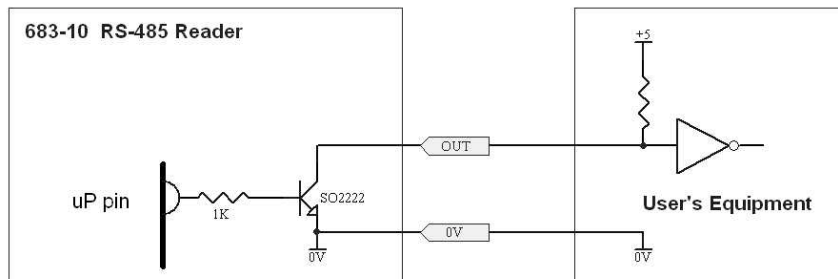
Input/Output Connections

The 683-52 is equipped with two active-low inputs, and two open collector transistor outputs. The inputs are pulled high internally (to +5V), and checking their states if no input is connected will yield a state of '1'. Grounding the input will yield a state of '0'.

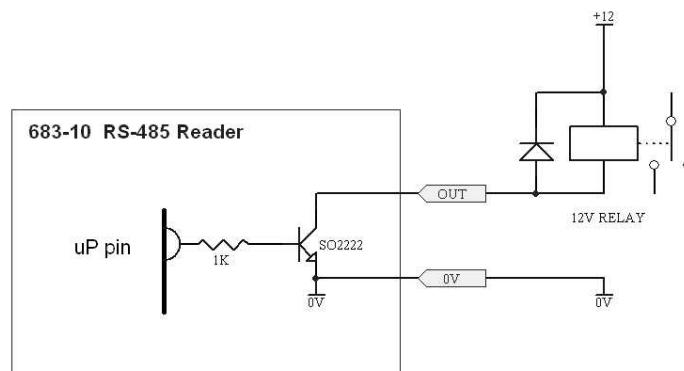


Schematic of Input

The outputs transistors are typically not conducting, corresponding to an output state of '0'. When the output is set, the transistor conducts and grounds the connection.



Schematic of Output, showing connection to TTL device



Schematic of Output, showing connection to a Relay

RS-485 Connections

An RS-485 network is typically an end-to-end bus, with a 120Ω terminating resistor on each end device, connected between the RS-485 + and the RS-485 - lines. One device on this bus (typically the host or controller device) must bias the RS-485 + to the positive rail voltage, and the RS-485 - to the negative rail voltage, using 560Ω resistors.

Additional devices are then connected to this bus.

The 683-52 provides the facility to terminate a line (ie. be the last device on the bus), but does not provide bias resistors. This must be done on a different device on the network.

To terminate the line, the TERMINATE wire (PURPLE) must be connected to the RS-485 + wire (ORANGE).

The 683-52 allows 5ms for the host device to turn its RS-485 drivers around and prepare for a reply before it starts transmitting the reply packet.

Grounding Considerations

Grounding an RS-485 network needs to be done properly to avoid ground loops and different potentials between power-supplies.

Ideally, all power-supplies used for devices need to be of a floating variety, where a transformer isolates the ground from the mains supply. The RS-485 lines should then be accompanied by a ground wire, which links all devices to the same reference point.

Alternatively, if the power-supplies are grounded, the ground component of the RS-485 should be left out.

If a combination of these options is used, ground loops may result, which may cause damage to the RS-485 ICs when transient noise is introduced.

Address Programming

To function correctly on the RS-485 network, each device requires a unique address in the range of 1-15. This can be programmed via the network, but is also programmable by using the PROGRAM control line.

To enter Programming Mode, power up the device with PROGRAM held low.

1. The beeper will emit a short tone, and the yellow LED will light. If modifying the current address, keep the PROGRAM line low. If only viewing the current address, release the PROGRAM line as soon as the yellow has lighted. Jump to Step 5 if PROGRAM was released.
2. The current address will be flashed using the green LED.
3. The yellow will then light again, warning that you are about to modify the address. PROGRAM can still be released at this point.
4. The red LED will begin to flash. Counting flashes, release the PROGRAM line when the desired address has been flashed. Return to Step 1.
5. After flashing out the current address, the unit will exit programming mode and return to normal operation.

The factory default is address 1.

Communications

The protocol has been designed to maximize the host unit's control over the 683-52 reader. The host continuously polls each device on the network for data and simultaneously sets the LED and buzzer state to that required by the system. A reply packet is typically generated within 5ms of a command.

A timeout behaviour is initiated if the device has not received communications for a configurable period of time. This can be disabled, or set to a value between 5s (default) and 15s. When in a timeout state, all three LEDs flash together.

Line Configuration

Baud Rate:	9600 bps	Start bits:	1
Data bits:	8	Stop bits:	1
Parity:	None		

Packets consist of ASCII characters, with an optional checksum for data integrity.

Packet Structure

[Start Character][Device Address][Command Code][Parameters/Data][Optional Checksum][End Character]

Note: To use the checksum functionality, the device needs to be configured appropriately. See the CHECKSUM ENABLE command ('V') for more information

Where:

[Start Character] is always '>' (0x3E) when a message is sent from the host to the 683-52. It is always '<' (0x3C) when the 683-52 replies to the host.

[Device Address] is an ASCII character between '@' (0x40) and 'O' (0x4F). '@' (0x40) represents device 0, 'A' (0x41) represents device 1, 'B' (0x42) represents device 2 etc. 'O' (0x4F) is device 15. These must be uppercase.

[Command Code] is an ASCII character in the set [K,P,Q,R,T,U,V,Y,Z]. These are discussed in more detail further in this document, as are their parameters.

[Parameters/Data] contain ASCII characters. These will be discussed as relevant to each command. It is important to note that parameters like 0 or 1 need to be sent as '0' (0x30) and '1' (0x31) respectively.

[Optional Checksum] is an additive checksum byte which is calculated by adding each character from the device address character to the last of the parameter characters. For example, if the command is ">AK00", then the checksum is calculated as $0x41 + 0x4B + 0x30 + 0x30 = 0x00EC$, therefore the checksum is 0xEC, or 'ì' (ASCII #236).

[End Character] is always the carriage return character, 0x13. It cannot be shown on paper, so will be represented as '↵'

Examples:

">BK01↵" is a poll command to device 2, setting the LEDs off and enabling the beeper.

">ARKK↵" is a re-address command to device 1, setting the new address to 11, using a checksum.

Commands

These commands are all shown without checksums. As stated above, all characters are ASCII values.

Poll Command	'K' (0x4B)
Time-out Period	'P' (0x50)
Query Configuration	'Q' (0x51)
Re-address	'R' (0x52)
Set Outputs	'T' (0x54)
Set LEDs\Beeper	't' (0x74)
Get Inputs	'U' (0x55)
Use Checksum	'V' (0x56)
RS-485 Reply Delay	'W' (0x57)
Version Request	'X' (0x58)
Tag-Reply Format	'Y' (0x59)

These are detailed below:

Poll Command **'K' (0x4B)**

This command instructs the device that it is permitted to transmit data on the network, and also sets the LED and Buzzer states on the specified device.

Parameters:

ASCII byte 1	The LED state number (0,1,2,3,4,5,6 or 7) 0=All Off, 1=Green, 2=Red, 3=Yellow, 4=Red & Green, 5=Red & Yellow, 6=Green & Yellow, 7=All On, 8=No change
ASCII byte 2	The beeper state (0 or 1) 0=Off, 1=On, 2=No change

Examples:

>AK00↵	Poll for data on device 1, Set LEDs and buzzer off
>BK11↵	Poll for data on device 2, Set LEDs green and buzzer on
>CK21↵	Poll for data on device 3, Set LEDs red and buzzer on

Comment:

On receiving this command, the device maintains the LED and buzzer state until it receives a new command with a different state. If it has not received a command for more than the configured timeout period, the 683-52 reverts to its fault condition: flashing all three LEDs, with the beeper off.

It is also important to note that RFID sensing is disabled while the beeper is being driven, resuming immediately after the beeper is turned off.

'No change' will leave the state unaltered. This is to allow control of the LEDs using the 't' command.

Reply:

If a new tag has been scanned which has not yet been requested, the unit replies with a tag number in an 'S' packet. This packet (for legacy reasons) contains two extra bytes, one before the tag number (always 3) and one afterwards (always 0). These are not currently indicative of any information.

If the device does not have a new tag to return, it simply replies with a 'K' packet.

Examples:

<AK↵	No new tag available on device 1
<BS <u>3005A4BF</u> 40	A tag was available on device 2, and was replied. The underlined portion is the tag number, while the two extra bytes appear on either side of it. The tag number is always returned in hexadecimal (converting in this case to 5917684)

Note: Depending on the configuration of the device, the tag may be returned in 32 bit or 40 bit format. In the case of 40 bit format, the header byte will be returned too. (The example above would be <BS3FF005A4BF40 if the tag header was 0xFF). See Tag-Reply Format command for more information.

Time-out Period 'P' (0x50)

The time-out period is the number of seconds that pass between the last communication and the start of timeout behaviour.

Parameters:

ASCII byte 1 The time-out constant (0,1,2 or 3)
0=Off, No timeout
1=5 seconds (Default)
2=10 seconds
3=15 seconds

Examples:

>AP2↵ Set the timeout on device 1, to 10 seconds
>FP0↵ Disable the timeout on device 6

Comment:

This timeout period should be modified when the network polling is particularly busy, or alternatively when the poll rate is very slow and the devices start to lapse into timeout behaviour between polls.

Reply:

A short beep indicates that the device has accepted the change and written it to the configuration EEPROM. The reply packet is simply a 'P'.

Examples:

<AP↵ Command was accepted on device 1

Query Configuration 'Q' (0x51)

Query the device for it's status. Configuration details and states of the INPUT and OUTPUT lines are returned.

Parameters: None

Examples:

>AQ↵ Query device 1

Reply:

The data of the reply packet is formed as follows:

[CHKSUM ON/OFF][FORMAT][TIMEOUT PERIOD][RS485DELAY][IN1][IN2][OUT1][OUT2]

where each value is an ASCII digit ('0','1'... etc)

Examples:

<CQ01230110↵ Device 3 has settings: Checksum OFF, 40-bit Format, 15s Timeout Period. RS485 replies will wait 3ms before replying, INPUT 1 is low, INPUT 2 is high, OUTPUT 1 transistor is conducting, OUTPUT 2 transistor is off.

<FQ00040001↵ Device 6 has settings: Checksum OFF, 32-bit Format, Timeout Period is off. RS485 replies will wait 4ms before replying, INPUT 1 is low, INPUT 2 is low, OUTPUT 1 transistor is off, OUTPUT 2 transistor is conducting.

Re-address 'R' (0x52)

Use this to change the address of a device via the network.

Note: This is also the only way of programming a device to address 0 ('@', 0x40)

Parameters:

ASCII byte 1 The new address ('@', 0x40 - 'O', 0x4F)
ASCII byte 2 Duplicate of the first byte

Examples:

>ARHH␣ Change device 1 to device 8
>DR@@␣ Change device 4 to device 0

Comment:

It is important to note that the reply packet will originate from the previous address (ie. the address the command was sent to, not the modified address). This is to allow the host to correctly match this reply to the command sent.

Reply:

A short beep indicates that the device has accepted the address change and written it to the configuration EEPROM. The reply packet is simply a 'R', from the previous address (as discussed above)

Examples:

<AR␣ Device 1 received an address change packet

Set Outputs 'T' (0x54)

Control the state of OUTPUT 1 and OUTPUT 2

Parameters:

ASCII byte 1 State of OUTPUT 1
 0=Output Transistor is not conducting
 1=Output Transistor is conducting
ASCII byte 2 State of OUTPUT 2
 0=Output Transistor is not conducting
 1=Output Transistor is conducting

Examples:

>@T00␣ Set both OUTPUTS on device 0 off
>HT01␣ Set OUTPUT 2 on, and OUTPUT 1 off on device 8

Comment:

The states of these outputs remain unaffected by any other polling activity. They will maintain their levels until the unit is reset.

Reply:

The reply packet is simply a 'T'.

Examples:

<AT␣ Device 1 received the output command.

Set LEDs\Beeper 't' (0x74)

Manipulate the states of the 3 LEDs and Beeper

Parameters:

ASCII Byte 1 Element Select
0=Red LED
1=Yellow LED
2=Green LED
3=Beeper

If an LED was selected (ASCII Byte 1 within 0-2)

ASCII Byte 2 Duration (in seconds)
0=Toggles the current state
1-9=Number of seconds the LED state will be in effect (1-9).
ASCII Byte 3 Flash Frequency
0=Solid light
1-6=Flash frequency, determined from the table below.

If the Beeper was selected (ASCII Byte 1 = 3)

ASCII Byte 2 and 3 Duration (in 100ms)
0=Toggles the current state
This and ASCII Byte 3 will form the number of 100ms the Beeper will be in effect. (1-99 intervals)
ASCII Byte 4 Beep Frequency
1-6=Beep frequency, determined from the table below

Frequency Table	
ASCII Byte	Frequency (Hertz)
1	5
2	2.5
3	3.5
4	1.25
5	1
6	0.55

Examples:

>At050.␣ Set Red led on for 5 seconds, no flashing, device 1
>Bt131.␣ Set Yellow led for 3 seconds, 5 Hz flashing, device 2
>Ct3652.␣ Set Beeper for 6.5 seconds, 2.5 Hz, device 3

Comment:

The states of these outputs will naturally be affected by the 'K' polling command unless the two 'No change' options are used (see the 'K' command comments for more information). Furthermore, if the device is set to enter timeout behaviour, indicated by all LEDs flashing at 2 Hz, the timeout mode will control the LEDs and beeper.

Reply:

The reply packet is simply a 't'.

Examples:

<At.␣ Device 1 received the set LEDs\Beeper command.

Get Inputs 'U' (0x55)

Read the state of INPUT 1 and INPUT 2

Parameters: None

Examples:
>AU↵ Read inputs on device 1

Reply:
The data of the reply packet consists of the two ASCII represented states, [IN1][IN2]

Examples:
<AU01↵ Device 1 reports IN1 is low and IN2 is high

Use Checksum 'V' (0x56)

Depending on your application, you may require a means of error-detection. The 683-52 can use an additive checksum to verify the contents of packets. If this is enabled, then each command requires a checksum byte to be inserted before the [End Character]. If this is absent, and the device is expecting a checksum, the CHKERR will be returned. (See Error Messages)

Parameters:
ASCII byte 1 Enable\Disable Checksum
 0=Disable (Default)
 1=Enable

Examples:
>AV0↵ Disable checksums on device 1
>BV1↵ Enable checksums on device 2

Comment:
It is important to note that by enabling the checksum, the change takes effect immediately and the reply will return with a checksum byte. Similarly, by disabling the checksum, the reply will not contain a checksum byte.

Reply:
A short beep indicates that the device has accepted the checksum change and written it to the configuration EEPROM. The reply packet is simply a 'V', but if enabled a checksum will be included.

Examples:
<BVÿ↵ From the above example, device 2 received an Enable Checksum packet, and subsequently replies with a checksum included (ÿ, 0x98)

RS-485 Reply Delay 'W' (0x57)

Configure the delay which occurs on the unit between receiving a command and replying to it. This can be adjusted if the RS-485 line drivers are slow to turn around. By default, this is set to 0, ie. OFF.

Parameters:

ASCII byte 1 The delay constant (0 - 9) in milliseconds.
0=Off, Reply is returned as soon as possible
1-9, processor delays for x milliseconds before sending the reply packet
Default value is 2ms.

Examples:

>AW2↵ Configure device 1 to delay 2ms before replying
>FW0↵ Configure immediate replies from device 6

Reply:

A short beep indicates that the device has accepted the change and written it to the configuration EEPROM. The reply packet is simply a 'W'.

Examples:

<AW↵ Command was accepted on device 1

Version Request 'X' (0x58)

Returns the firmware version

Parameters: None

Examples:

>AX↵ Request version on device 1

Reply:

The data of the reply packet consists of the three ASCII digits, combined to form vD1.D2D3, where D1, D2 and D3 are the corresponding digits

Examples:

<AX103↵ Device 1 returns its firmware version as v1.03

Tag-Reply Format 'Y' (0x59)

Depending on your application, you may require the full 40 bits of RFID code available.

Parameters:

ASCII byte 1 Enable\Disable Checksum
0=32 bits, No header (Default)
1=40 bits, Header byte included.

Examples:

>AY0↵ Enable 32 bit Tag Reply format on device 1
>BY1↵ Enable 40 bit Tag Reply format on device 2

Reply:

A short beep indicates that the device has accepted the format change and written it to the configuration EEPROM. The reply packet is simply a 'Y'

Examples:

<BY↵ Device 2 received a format change

Error Messages

Three error messages are generated by the 683-52. The format of an error message resembles that of a data packet, where the data portion of the packet is the error message. The packet is rejected, and no action is taken by the device.

CMDERR	Indicates that the packet contained an unknown command character.
ARGERR	Indicates that one or more of the arguments/parameters sent within the last packet were out of range (example: sending a 0 (0x00) instead of a '0' (0x30))
CHKERR	Indicates that the checksum received on the previous packet did not match the expected checksum. This implies that the packet was damaged somehow, or the checksum was incorrect. The command character for the previous packet is not included within a CHKERR packet.

Examples:

<AKARGERR.␣	Device 1 received one or more incorrect parameters for a poll command
<BXCMDERR.␣	Device 2 received an unknown command character (in this case 'X')
<BCHKERR(0x01).␣	Device B received an incorrect checksum. Since it was expecting a checksum, a checksum will obviously be included with the reply - in this case (0x01)