

Auto-Read mode in the
RIDEL5000 Long Range Reader

Application Note



EHAG

ELECTRONIC HARDWARE AG

Industriestr. 8 CH-8618 Oetwil a/S.

T: +41 43 844 94 00 info@ehag.ch

F: +41 43 844 94 01 www.ehag.ch

Code: 140.0004.0 Revision: 1

May 2001

Index

1	INTRODUCTION	3
2	AUTO READ CONFIGURATION	4
3	OPERATION	5
4	READ MEMORY COMMAND FROM THE DLL	6
5	PROTOCOL DESCRIPTION	7
5.1.	SERIAL PROTOCOL	7
5.1.1	CHARACTER DEFINITIONS	7
5.1.2	PROTOCOL DESCRIPTION	7
5.1.2.1	COMMAND SEQUENCE	8
5.1.2.2	RESPONSE SEQUENCE	8
5.1.3	DATA BLOCK FORMATS	8
5.1.3.1	HOST TO RIDEL5000 (COMMAND)	8
5.1.3.2	RIDEL5000 TO HOST (RESPONSE)	9
5.1.3.3	DESCRIPTION OF THE DATA BLOCK	9
5.2	RIDEL5000 FUNCTIONS	11
5.2.1	READ MEMORY COMMAND	11

1 INTRODUCTION

Polling is the usual way to interface an RFID reader to the host application. In this mode, the host computer will continuously send read commands to the reader, and this will trigger the reader to send the corresponding command through the air interface, to look for the tags present in the field.

This approach presents some problems described next:

- Depending on the application, the computer, the operating system, ... the time between commands may not be consistent. Some interruptions due to database or network access, other applications running or other factors could delay the read operation, resulting in tags not being read when they are in the field for an insufficient time.
- If there is more than one reader involved, in a RS485 operation, the time between commands can be even longer, so more and more tags will not be read.
- If the link with the host is temporarily broken, or the host stop working, all the tags will be lost during this time.

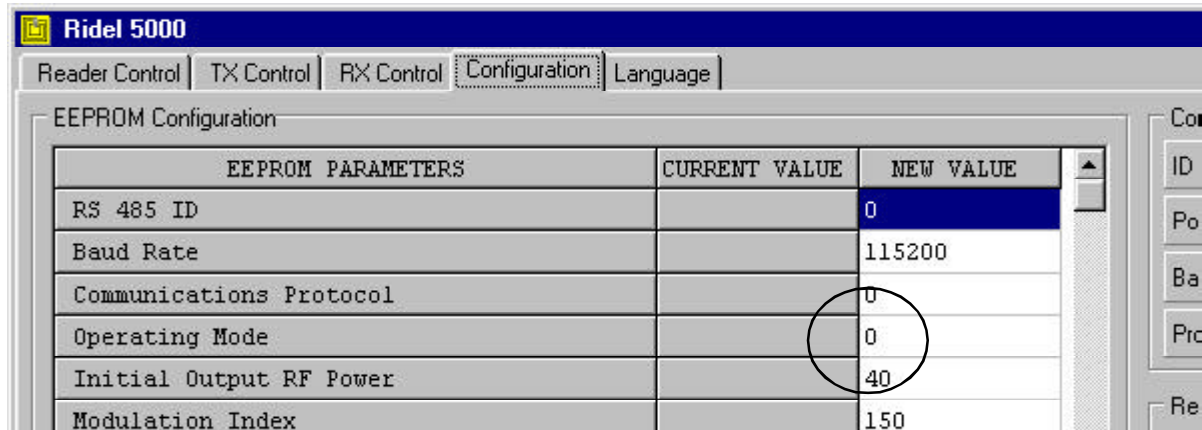
In order to solve these problems, Softrónica has introduced a new feature in their long range readers called «AUTO READ» mode. When the reader is configured in this mode of operation, it will continuously send read commands through the air interface, at a rate much higher than the polling rate of the host. Any tag in the field will be read in this way, and the contents will be stored in the internal memory of the reader.

The host application will only have to send a «READ MEMORY» command with a much lower priority, and more relaxed real-time requirements. As the internal RAM is able to store more than 500 tags, this command could be sent, depending on the application requirements, every second, 10 seconds, minute, ...

In the remainder of the document, the procedure to make the RIDEL5000 work in this mode is described, along with the setup instructions.

2 AUTO READ CONFIGURATION

To configure the RIDEL5000 in this mode of operation, it will be necessary to change the «OPERATING MODE» configuration parameter. It is possible to do it through the demo software, as described in the following drawing:



It will be necessary to set to 1 the bit 6 (add hexadecimal 40) of this operating mode, and then, store the resulting value in EEPROM, in order to make the unit start in this mode.

The following steps shall be followed to perform this operation if using the Softrónica demo program:

- Start the program, and press «UPDATE» in the main window to check the communication link.
- Go to the Configuration screen in the program
- Press the «GET EEPROM VALUES» button on the lower part of the screen
- Check the operating mode, translate it into hexadecimal, and check if the bit 4 (weight hexadecimal 10) is set. If not, add hexadecimal 10 (decimal 16) to the corresponding value, and write it in the corresponding field.
- Press the «SET EEPROM VALUES» button in the lower part of the screen.
- Reset the unit by unplugging the power supply and plugging it in again.

The unit will start in auto-read mode when powered on.

3 OPERATION

Once the RIDEL5000 is in auto-read mode, it will store all the tags correctly read. The reader will operate as follows:

- Every tag will be stored just once, provided there are no other tags read in the meantime. That is, if a tag is read 10 consecutive times, it will be stored just once. If then another tag is read, this second one will be stored, and if the first one appears again, it will be stored again.
- The memory read command will empty the reader memory contents automatically
- One time slot will be used for the operation. The reader will automatically readjust the operation if there is more than one tag in the field.
- The reader will store the whole memory contents of more than 500 tags. When the buffer is full, the reader will stop collecting tags until emptied again.

4 READ MEMORY COMMAND FROM THE DLL

The procedure to retrieve the data from the memory of the reader when using the DLL is described next:

Function

```
BYTE RFM_read_mem (BYTE *resp);
```

Arguments

***resp** Pointer to an array to contain the tag memory contents. The dimension of this array has to be enough to hold the data corresponding to the number of blocks requested and the number of timeslots. Let **tse=n**, **blnr=m** and **nobl=x**. The read data will be returned in the following format:

resp length:	4+(64*n)	n=tag number
resp[2]		Low part of the number of bytes
resp[3]		High part of the number of bytes
resp[4..]		Tag data

The status byte for every timeslot contains information of the results of the reading process for the corresponding timeslot. This byte can have the following values:

00	No error
01	No tag
02	CRC Error in the RF interface
03	Collision. 2 or more tags have answered in the same slot
08	Weak collision. There is a collision, but the answer from one tag is clear

Returned value

OK	0x00
ST_WAITING_COMAND	0x01
ST_SENDING_COMAND	0x02
ST_WAITING_STX	0x03
ST_WAITING_DATA	0x04
ST_COMAND_OK	0x05

Description

This function is used to ask the reader for the memory contents of a set of labels stored in the RIDEL5000 memory when working in stand-alone mode. In this mode, the RIDEL5000 stores all the tags correctly read in its internal memory.

All the corresponding data can be then retrieved with this command in the same format as a normal reading operation.

The **resp[2..3]** bytes indicate the length of the remaining response, that is, the number of bytes returned minus 4.

A deeper description of the use of the DLL is found on the RFID Software Development Manual.

5 PROTOCOL DESCRIPTION

The read memory command and response will be described in this section for the Philips communication protocol. This description is first found on the «RIDEL5000 Long Range I-CODE reader/encoder. TECHNICAL MANUAL».

The use of this mode in a different protocol will be described in next releases of this application note.

5.1. SERIAL PROTOCOL

For point to point communications, the RIDEL5000 incorporates a RS232 link with the controlling computer. The communication parameters are defined as follows:

- 8 Data Bits
- One Stop Bit
- No Parity bit

The communication speed is software configurable between 9600 and 115200 baud. The unit is factory configured at 115200 baud.

5.1.1 CHARACTER DEFINITIONS

Description	Char	Value
Start of text	STX	02Hex
End of text	ETX	03Hex
Data link escape	DLE	10Hex
No acknowledge	NAK	15Hex

5.1.2 PROTOCOL DESCRIPTION

To start a communication, the transmitter (at the command sequence this is the host, at the response sequence, this is the RIDEL5000) and the receiver (at the command sequence the RIDEL5000, at the response sequence the host), must be ready. The transmitter starts with STX to establish a data link.

If the receiver answers NAK (or nothing), the transmitter repeats to send STX. If this trial fails again a third (i.e. the last) STX is transmitted to the receiver. If no valid response (DLE) is returned from the receiver, an error message is generated finally, and the transmitter stops trying to establish a data link.

If the receiver answers DLE within a specified period of time (0.5s), data can be transmitted.

If DLE appears within a data block, it is transmitted twice to distinguish it from the control character DLE.

If the defined maximum character delay (0.5s) is exceeded during transmission of the data block, the receiver returns to the idle state and waits for another STX to establish a new data link.

At the end of transmission of the data block the transmitter transmits DLE and then ETX (DLE is necessary to distinguish a control character from a data byte).

If the receiver detects no error in the transmission (i.e. correct CRC), it answers DLE. If an error is detected, the receiver sends NAK, then the transmitter tries to repeat the entire transmission (maximum two times). If this is not possible it stops sending data and generates an error message.

5.1.2.1 COMMAND SEQUENCE

HOST (Transmitter)		RIDEL 5000 (Receiver)
STX	⇒	Receiver Ready?
	⇐	DLE
Data[0]	⇒	Start of data block transmission
..		
Data[n]	⇒	
DLE	⇒	Next: Control character
ETX	⇒	End of transmission
	⇐	DLE/NAK
		DLE: No error
		NAK: An error occurred

5.1.2.2 RESPONSE SEQUENCE

HOST (Receiver)		RIDEL 5000(Transmitter)
	⇐	STX
		Receiver Ready?
DLE	⇒	DLE: Yes!
	⇐	Data[0]
		Start of data block transmission
		..
	⇐	Data[n]
	⇐	DLE
		Next->Control character
	⇐	ETX
		End of transmission
DLE/NAK	⇒	DLE: No error
		NAK: An error occurred

The time in which the receiver has to transmit the control characters upon the transmitter's request is 0.5s. This is also the allowed maximum delay time between two characters during the communication.

5.1.3 DATA BLOCK FORMATS

5.1.3.1 HOST⇔RIDEL5000 (COMMAND)

<i>TxSeq</i>	<i>Command</i>	<i>Len(0)</i>	<i>Len(1)</i>	<i>Par(0) ... Par(Len-1)</i>	<i>CRC16(0)</i>	<i>CRC16(1)</i>
Data(0)	Data(1)	Data(2)	Data(3)	Data(4) Data(Len+3)	Data(Len+4)	Data (Len+5)

<i>TxSeq</i>	Sequence number of the command	1 byte
<i>Command</i>	Command code	1 byte
<i>Len</i>	Number of parameter bytes (low byte, high byte)	2 bytes
<i>Par</i>	Parameter bytes of command	<i>Len</i> bytes
<i>CRC16</i>	16 bit CRC (low byte, high byte)	2 bytes

5.1.3.2 RIDEL5000P$HOST$ (RESPONSE)

<i>RxSeq</i>	<i>Status</i>	<i>Len(0)</i>	<i>Len(1)</i>	<i>Res(0) ... Res(Len-1)</i>	<i>CRC16(0)</i>	<i>CRC16(1)</i>
Data(0)	Data(1)	Data(2)	Data(3)	Data(4) Data(Len+3)	Data(Len+4)	Data(Len+5)

<i>RxSeq</i>	Sequence number of the response	1 byte
<i>Status</i>	Status byte (serial communication Host-Ridel)	1 byte
<i>Len</i>	Number of response bytes (low byte, high byte)	2 bytes
<i>Par</i>	Response bytes	<i>Len</i> bytes
<i>CRC16</i>	16 bit CRC (low byte, high byte)	2 bytes

5.1.3.3 DESCRIPTION OF THE DATA BLOCK

- A sequence number is generated from the host sent within the data block. After a correct command/response exchange, the host increases the sequence number at the next command. The RIDEL5000 returns always the last received sequence number.

It is recommended that the host application verifies the equality of the sent and received sequence numbers after every command/sequence exchange.

- If the value 10 Hex (DLE) is transmitted within a data block, it is transmitted twice in order to distinguish it from the control character DLE.

The additionally transmitted value 10 Hex is not counted in ***Len***.

- If the value 10 Hex (DLE) is transmitted within a data block, it is transmitted twice in order to distinguish it from the control character DLE.

The additionally transmitted value 10 Hex is not counted in ***Len***.

- The 16 bit cyclic redundancy check character (CCITT-CRC16) is calculated as described in the following:

Generator Polynom: $X^{16} + X^{12} + X^5 + 1$ \Rightarrow CRC_POLYNOM = 8408 Hex

Preset Value: \Rightarrow CRC_PRESET = FFFF Hex

Calculation algorithm (C example)

```

unsigned int crc=CRC_PRESET;

for (i=0;i<cnt;i++)          /*Command:   cnt=Len+4;*/
                            /*Response:  cnt=Len+6;*/
    {
        crc^=Data[i];
        for (j=0;j<8;j++)
            {
                if (crc & 0x0001)
                    crc=(crc>>1)^CRC_POLYNOM;
                else
                    crc=(crc>>1);
            }
    }

/* Command */
Data[i]=crc & 0xFF;          /*CRC16 Low Byte*/
Data[i+1]=crc >> 8;        /*CRC16 High Byte*/

/* Response */
if (crc==0)
    {
        /* CRC claculation of response ok */
    }
else
    {
        /* CRC error occurred */
    }

```

The crc variable is set to the preset value only at the beginning of the preparation of a command/ response sequence for transmitting to the RIDEL5000/host respectively.

At the command sequence the CRC value is calculated fro the bytes Data[0] .. Data[Len+3] of the data block (including **TxSeq, Command, Len**).

At the response sequence, the CRC value is calculated for all data bytes (Data[0] ... Data[Len+5]) of any response block (including **RxSeq, Status, Len** and both CRC bytes). The resulting CRC value is 0 if no error at transmission occurred.

If a 10 Hex occurs within the data block, 10 Hex is used **once** for the CRC calculation, but transmitted **twice** to the receiver.

5.2 RIDEL5000 FUNCTIONS

In the following, the serial number of a tag is always shown as 64 bit hexadecimal value (e.g. 01 23 45 67 89 AB CD EF Hex).

The location of these eight bytes in the tag's memory are:

Block 0: EF CD AB 89 Hex (Bytes 0,1,2,3 in the first block)

Block 1: 67 45 23 01 Hex (Bytes 0,1,2,3 in the second block)

5.2.1 READ MEMORY COMMAND

Host → RIDEL5000

Command: 20 Hex

Len: 0 (No parameters)

RIDEL5000 → HOST

Len: $n \cdot 64$ (n: number of tags) (*)

Resp[0..63] First tag in memory (tag 1)

Resp[64..127]: Second tag (tag 2)

:

Resp [$64 \cdot (j-1) .. (64 \cdot j) - 1$]: Tag j

:

Resp [$(64 \cdot (n-1) .. (64 \cdot n) - 1$]: Tag n (last tag)

The number of tag calculation is based on the length data contained on the header of the message.

