Industriestr. 8 8618 Oetwil a/S. Tel: +41 43 844 94 00 Fax: +41 43 844 94 01
http://www.ehag.ch E-Mail: info@ehag.ch

ELECTRONIC HARDWARE AG

MicroEngine Tag-it 13,56 MHz

Protocol Version 00.05

Memory Organisation of the Tag-it Label

The 256 bit EEPROM memory is divided into 8 blocks. Each Block consists of 4 bytes which can be used for user data.
The serial number (32 bit) is accessible by using a special command ('i') and transmitted continuously after reset (power up). It is not mapped into the one of the pages (like page 0 at I-Code Tags).

Block 0	User Data
Block 1	User Data
Block 2	User Data
Block 3	User Data
Block 4	User Data
Block 5	User Data
Block 6	User Data
Block 7	User Data

Industriestr. 8 8618 Oetwil a/S. Tel: +41 43 844 94 00 Fax: +41 43 844 94 01
http://www.ehag.ch E-Mail: info@ehag.ch
ELECTRONIC HARDWARE AG

Reader Instruction Set, Firmware Version Tag-it V0.05

1. Modes of Operation

The reader has the following modes of operation:

A) The **Continuous Read Mode** continually reading the serial number. On power up the reader starts in this mode. After receiving some data from the host the reader turns to **Command Mode.**

B) In the **Command Mode** the reader waits for commands from the host or PC. Depending on the first character of a frame received the protocol type is selected. The reader replies in the corresponding protocol mode.

2. Protocol Modes

The following protocol modes are available:
A) **Mode C: Continuous Mode Protocol**. After power on the reader starts automatically reading continuously the serial number of the tag. In addition to that commands compatible to the previous Microengine protocol V153 are supported. This is for compatibility purposes only and is not recommended for new designs.
All requests are starting with lower case ASCII characters

B) **Mode L: Light Frame Protocol**. This protocol is intended for testing as well as for OEM configurations. All requests are starting with upper case ASCII characters. Hexdigits are coded as two ASCII characters, representing high nibble and low nibble. For example 0x15 is coded as „15“ or chr$(31)+chr$(35). Request digits are ASCII and represented without modification.

3. Physical Layer

Telegram bits and bytes are transmitted sequentially and asynchronously.
Bitlength is 1/9600 sec/Bit = 104.2 usec/Bit on a 9600 baud line.
Data Frame for each byte is:
1 Startbit (Logic 0)
8 Databits, starting with LSB, ending with MSB
1 Stopbit (Logic 1)
The bytes of a telegram package are transferred sequentially, first byte first, last byte last. If no transmission is in progress, the interface goes to idle level (logic 1).

4. Data Representation

Depending on frametype all numeric databytes to be transfered between master and tag are packaged as binary or as ASCII Hexadecimal format.
ASCII Hexadecimal format example:
167 dec = A7 hex = 10100111 bin is transmitted as ‚A7‘ (ASCII ‚A‘ and ‚7‘).
Numeric databytes are transferred without space between.

5. Lean Protocol

This protocol type is compatible with the previous MicroEngine protocol V153 and 2214. It is supported for compatibility purposes only and is not recommended for new designs.

Data format is ASCII-Hex. Data from reader is <CR><LF> terminated. All requests are starting with lower case ASCII characters

Continuous Read is the mode of operation after start or reset (reading page 00).

Reader Commands

Command	Request To Reader	Answer From Reader
Get TAG Info	I	4 hexbytes (serial number) + CR-LF 1 hex byte (manufactorer) + CR-LF 2 hexbytes (chip version) + CR-FL 1 hex byte (number of byte per page) + CR-LF 1 hex byte (number of pages) + CR-LF
Continuous read	**c**xx, xx denotes block to be read c00	4 hexbytes +CR-LF Nothing if no tag is read EA1E4D0000000001
Read	**r**xx, xx denotes block to be read r02	8 hexbytes +CR + LF ‚N'+CR+LF if no tag is read F0FFFFFF03000000
Write	**w**xxyyyyyyyy, xx denotes page to be written, yyyyyyyy denotes data to be written to block; only first 4 bytes are actually written. w08A1FF7388	W+4 hexbytes+CR+LF if successful, last 4 bytes 0-padded N+CR+LF if writing has failed
Lock block	**K**xx, xx denotes block to be locked	Locks a single block. (makes it permanently read only)
Reset	**x**	
LED Activation	dg (turn green) dr (turn red) dn (turn off)	DG | DR | DN

6. Light Frame Protocol

The light frame structure is for testing as well as for OEM configurations with good error handling provided by some external controller (f.e. handheld applications). It is restricted to point to point transmission at short distances and can be issued even manually on a terminal program without writing a line of code.

1 byte	n bytes	2 bytes
CMD or ANS	DATA	CR-LF

Where:
CMD command sent to reader
ANS return answered from reader
Data databytes to or from reader; may be empty (0 bytes)
CR-LF frame termination by 0x0D 0x0A characters (carriage return – line feed)
All data is transmitted in ASCII-Hex.

(7. --- left free)

8. Command Set (Presentation Layer)

8.1 Command Overview

Tag access	**‚R‘ Read** **‚W‘ Write**
Hardware	**‚V‘ Get Version, Serial number** **‚PW‘ Port Write** **‚A‘ Antenna RF on, off**
Initialization	**‚Z‘ Reset**

8.2 Page Size and Tag Identifier

All data from tags comes in 4 byte blocks ands is preceded by a tag identifier.

Tagtype	**Tag Identifier**
Gemwave ARIO 10	‚G‘
Gemwave ARIO 40	‚G‘
Tag-it	‚T‘
ICODE	‚I‘
EM4050	‚R‘
EM4002	‚U‘
Iso Animal	‚Z‘

Example: Serial number of an ARIO 40 tag
T00A98B53

8.3 Tag Access Commands

8.3.1 Read Pages

Command	
CMD	DATA
‚R‘	M (1 byte), A1(1 byte), A2(1 byte) Where M: trigger mode 00 Read once, timeout 100 msec 01 Read once until valid data received, no timeout 02 Continuously repeat reading, ‚N‘ if no tag 03 Continuously repeat reading, silent if no tag A1: number of first block to read A2: number of last block to read The number of blocks is limited to the buffer capability of the coupler, always 4 lines of 2 blocks. This is to ensure that all requested pages or no pages are read, avoiding fragmented blocks.

Answer	
ANS	DATA
T(1 byte) where T: Tag identifier (see below)	A (1 byte), blocks (4 bytes) Where A: number of starting block read blocks: data from tag
‚N‘: no tag	None
‚F‘ incorrect query	None

The answer frame is repeated for each page requested.

8.3.2 Write Page

Command	
CMD	DATA
‚W‘	Page (1 byte), Data (4 byte) Where Page: page to write Data: data to write

Answer	
ANS	DATA; 4 bytes
‚W‘: Write OK	None
‚N‘: no tag	None
‚F‘: no write access or writing failed	None

8.3.3 Lock Page

Command	
CMD	DATA
‚K‘	Page (1 byte) Where Page: page to lock

Answer	
ANS	DATA; 4 bytes
‚L‘: Write OK	None
‚N‘: lock failed, no tag or already locked	None

8.4.1 Get Tag Version

Command	
CMD	DATA
‚I‘	none

Answer	
ANS	DATA
	serial number (4 bytes) + CR-LF TAG manufacturer (1 byte) + CR-LF 01...Texas Instruments Chip Version (2 bytes) + CR-LF number of bytes per page (1 byte) + CR-LF number of pages (1 byte) + CR-LF
‘N’: no Tag	None

This command gets the transponder information. Only transponders with four bytes per page a supported.
The additional information (Chip Version, Manufacturer, Memory Organisation) are provided by the transponder and not tested or proved by the reader module.

8.4 Hardware Commands

8.4.1 Get Version

Command	
CMD	DATA
‚V‘	none

Answer	
ANS	DATA
‚V‘ (1 byte)	M (1 byte), P (1 byte), V (2 byte), SN (2 byte) where M: Manufacturer 01..Datatronic Kodiertechnik P: Product number V: Version number SN: Serial number or 0-padded Example V012500050000

8.4.2 Antenna Power

This command is not supported at the moment

8.5 Initialization Commands

8.5.1 Reset

Command	
CMD	DATA
‚Z‘	None

Answer	
ANS	DATA
‚Z‘ (1 byte)	M (1 byte), P (1 byte), V (2 byte), SN (4 byte) Same as in GET VERSION command.

Industriestr. 8 8618 Oetwil a/S. Tel: +41 43 844 94 00 Fax: +41 43 844 94 01
http://www.ehag.ch E-Mail: info@ehag.ch
ELECTRONIC HARDWARE AG

Frequently Asked Questions

How can I test a reader quickly ? I am using Windows 95/98/NT

1. Make sure, that your reader is RS232-interface type
2. Connect your reader to COM2 (COM1) of the PC and apply appropriate supply voltage
3. Start HyperTerminal
4. Create a new connection (FILE/NEW CONNECTION)
5. Enter name of connection as you like (f.e. 'ICODE')
6. Select connect COM2 (COM1) direct connection
7. Connection setup 9600,8,n,1,no handshake
8. Put a tag to your reader. Serial numbers should be displayed properly
9. Enter commands via keyboard. They should be transmitted to the reader

How should I implement the device driver ?

This section is providing implementation ideas only. We do not claim that the methods below are sufficient or complete nor that any rights or patents are involved when using those or similar methods. In any case testing, testing, testing is recommended for good implementations.

A. Readonly Applications

This is very simple and straightforward. If you need the tag ID, just connect the reader to power, present a tag to the reader and wait for data. If you need other pages issue an cxx (continuous read) or R02xx command to get the page automatically whenever a tag is presented to the reader.

B. Readwrite Applications

This is far more complicated, than readonly. Avoid it, if possible.
The air gap between reader and tag is a very unreliable communication channel, subject to all kind of interference. A protocol handler using read and write commands has to be implemented on the host side to deal with all those cases. Special care has to be taken when implementing device drivers to handle the following cases:

- There is no tag in the field
- Tag is moved in the field. Sometimes there is communication, sometimes not.
- There is electromagnetic interference.

The required error detection and error correction could be as follows:

- Implement read- and write commands that expect a precise number of bytes from the reader. The command should return after a timeout (no response). The read command should also return after a 'N'-response (no card) from the reader.
- Always issue a read after write.
- Provide a feedback to the user (using LED signals etc.)

C. Ticketing Applications

Implementing ticketing can not be done in an afternoon. Extreme skill in designing and testing the required communication schemes is required.
Besides the communication problems to be coped with, ticketing applications must withstand manipulation and fraud attacks. Some of the attacks to be dealt with are:

- Copying the content from one tag to another.
- Reading a tag and writing a modified content to another tag.
- Copying a previous read image to a „used“ ticket, to recharge it.
- Manipulate the reading writing process by moving quickly different tags through the field.

The following methods are by far not a complete list. Lots of thought, experimenting and testing is required to get the best combination for a specific application.

- Use readonly ticket and a central database for the value balancing whenever possible. This is by far the most simple and most reliable implementation.
- Use ciphering.
- Handle transactions using several version blocks on the write area. The most recently written version block that satisfies all integrity checks is taken to be the valid one.

The protocol does not seem to have handshakes. How can I archive a high transmission security?

The problem with handshakes is, that they need more time to implement. Our protocol was designed for the shortest way to integration (most people want to get ahead with their application as quickly as possible).
Concerning error handling we have to distinguish between the host-reader link, the reader-transponder link as well as user- or environmentally induced errors such as tags entering or leaving the field. Good system design requires considering all error cases, handling them specifically and testing, testing, testing. The following set of examples is by far not complete and has to be adapted according to your requirements:

- For highest security on reading do a double reading and compare the results.
- You can incorporate some integrity checks within your data (such as check bits)
- In read-write applications do a read after write (always recommended).
- In ticketing applications you can put your sensitive data into blocks that contain integrity checks and a version counter. When you update this information, do not overwrite the block but rather create a new version on a new space. On reading, take the latest correct version of your data block.