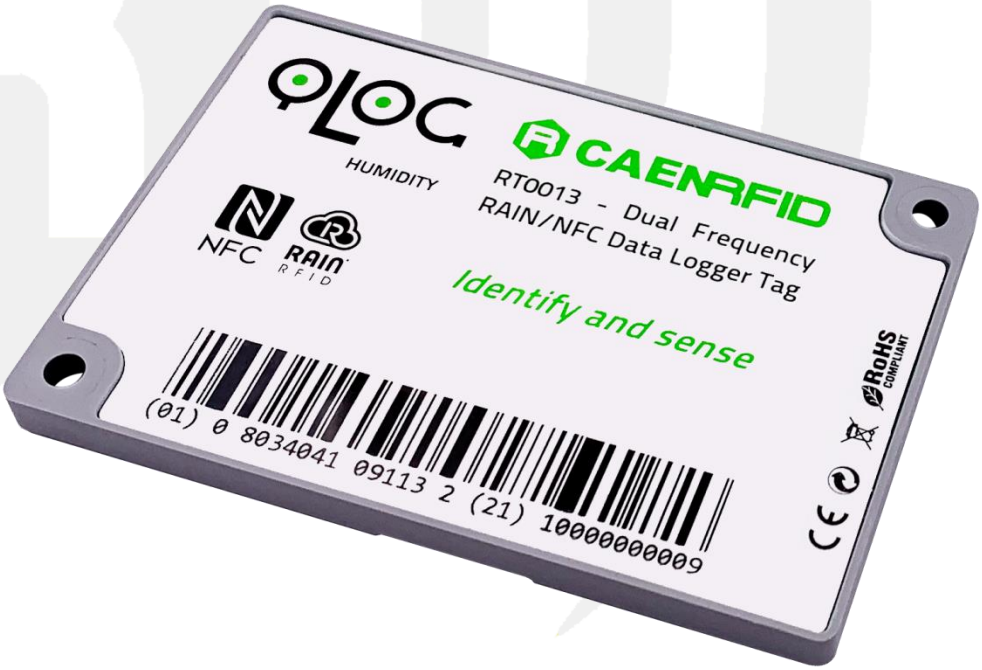
TECHNICAL INFORMATION MANUAL

Revision 0 – 10 September 2020

QLOC RT0013

HUMIDITY

Dual frequency RAIN/NFC humidity and temperature logger TAG



easy2log

CAENRFID

Visit the qLog$_{HUMIDITY}$ RT0013 web page, you will find the latest revision of data sheets, manuals, certifications, technical drawings, and software. All you need to start using your tag in a few clicks!

Scope of Manual

The goal of this manual is to provide the basic information to work with the qLog$_{HUMIDITY}$ RT0013 Dual frequency RAIN/NFC humidity and temperature logger TAG.

Change Document Record

Date	Revision	Changes	Pages
10 Sep 2020	00	Preliminary Release	-

Reference Document

[RD1] EPCglobal: EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz, Version 2.0.1 (April, 2015).

CAEN RFID srl

Via Vetraia, 11 55049 Viareggio (LU) - ITALY
Tel. +39.0584.388.398 Fax +39.0584.388.959
info@caenrfid.com
www.caenrfid.com

Disposal of the product

Do not dispose the product in municipal or household waste. Please check your local regulations for disposal/recycle of electronic products.

Index

List of Figures

List of Tables

1 INTRODUCTION

Product Description

CAEN RFID qLog$_{HUMIDITY}$ (RT0013) is a low cost, semi-passive NFC/RAIN RFID temperature and humidity logger that allows to monitor temperature and humidity sensitive products. The combination of the high resolution sensor, the large memory size and the standard NFC/RAIN RFID interfaces permit to realize effective track and trace solutions for the cold-chain.

The RAIN RFID interface is ideal for reading temperature data or alarms from distance allowing automated check-points on conveyors or through dock doors. The NFC interface permits a very easy interaction with any NFC enabled smartphone allowing the consumer to check the good condition at home.

The qLog$_{HUMIDITY}$ can be configured to store temperature and humidity samples in intervals from 5 second to 18 hours in the internal memory that can contain up to 4,096 samples. The user can define up to 16 temperature ranges with independent threshold alarms for a very accurate control of the temperature excursions.

The rugged enclosure and the compact size permit to use the logger in various applications and the passive radiofrequency behaviour does not prevent air shipments.

The qLog$_{HUMIDITY}$ RFID logger can be used for multiple shipments thanks to the long battery life and the reset function allowing to reduce the total cost of a single monitored shipment and anticipate the ROI of the solution.

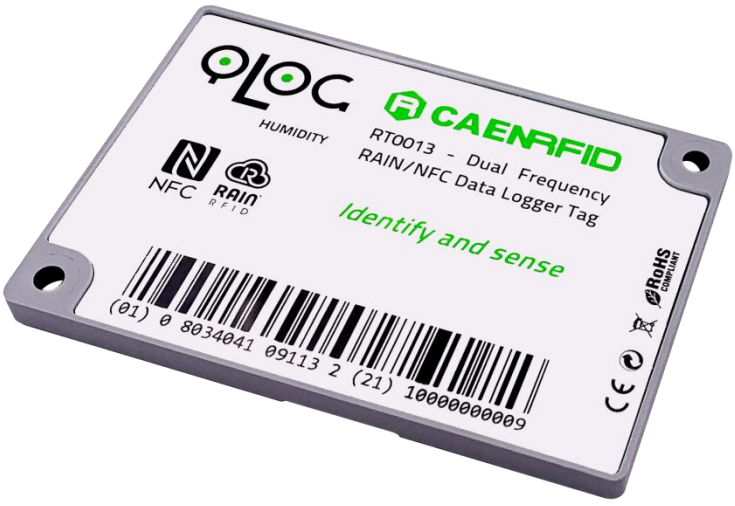


Fig. 1.1: qLog RT0013 - Dual frequency RAIN/NFC humidity and temperature logger TAG

Ordering Options

Code	Description
WRT0013XAAAA	RT0013 - Dual frequency NFC/UHF humidity and temperature logger TA

2 FUNCTIONAL DESCRIPTION

Operating states

The $qLog_{HUMIDITY}$ tag behaves according to the following operating states:

Idle. *Idl*e is the starting state of the tag. The tag is in *Idle* state when the internal clock is running, the memory is empty and the tag is waiting for a start logging command.

Active. The tag enters in *Active* state after having received the start logging command. In this state the tag wakes up at defined intervals in order to acquire temperature samples and, eventually, it stores those samples inside the log memory.

Sleep. The tag enters in the *Sleep* state at the end of a logging activity. Differently from the *Idle* state, in the *Sleep* state the tag's log memory is not empty and it is retained until a reset command is issued. When a reset command is issued, the tag enters the *Idle* state.

Logging and histogram functionality description

The $qLog_{HUMIDITY}$ RT0013 tag allows to define up to 6 different histograms bins for temperature and 6 different histograms bins for humidity . tutto da rivedere per T e H. The user is able to decide the upper and lower temperature limits for each bin: the lower limit of each bin is equal to the upper limit of the previous bin while the upper limit can defined by the user (BIN_HLIMIT_[X] where is X is the index of the histogram bin). The first bin starts always from the lower temperature limit of the tag while the upper limit of the last bin is always the higher temperature limit of the tag.

Each histogram bin has a bit that allows to enable/disable the counter for the bin (BIN_ENABLE_COUNTER_ register). If the bin is enabled, the corresponding counter (BIN_COUNTER[X]) is incremented when the sampled temperature value is greater than the lower limit and less or equal than the higher limit. If the bin is not enabled the temperature values in the interval defined by the corresponding bin are not counted.

For each bin it is possible to define a threshold value (BIN_THRESHOLD_[X]) in order to generate an alarm when the counter is greater or equal to the threshold value. The generation of the alarm can be disabled setting the threshold to 0xFFFF (default value).

It is also possible to define a different sampling interval for each different bin in order to permit a fine grained sampling in critical temperature ranges and a more relaxed sampling in standard temperature conditions. The sampling interval time is modified after the detection of the first temperature sample value in a new bin (if the value of the temperature falls in the same bin the sampling time does not change, if it falls in a new bin the sampling time is updated with the value of the BIN_SAMPLETIME[X] register where X is the index of the new bin).

Histogram bins configuration can be used also to control the storage of the samples inside the logging memory: for each bin (if enabled) the user can choose to store or not in the memory the temperature value (depending on the state of the corresponding bit in the BIN_ENA_SAMPLE_STORE register) and the time stamp (depending on the state of the corresponding bit in the BIN_ENA_TIME_STORE). Disabling the sample storage for accepted temperature ranges can provide a better handling of long term deliveries due to the memory saving. In this case it is necessary to enable the storage of absolute timestamps together with the samples out of the acceptance range in order to keep track of the chronological sequence. At the same time, the bin counter can be enabled also for the accepted temperature range in order to have a feedback on the temperature trend for the entire delivery. Due to the availability of a large number of bins, the accepted range can be divided in more than one bin in order to have finer grained information on the temperature trend.

Examples of usage

Example 1

Let suppose that the user wants to monitor the temperature trend for frozen food, we can define some histogram bins as follows:

Bin	Low Limit (°C)	High Limit (°C)	Threshold	Store Samples	Sampling Interval (min.)
0	-30	-15	0xFFFF	Yes	15
1	-15	-10	3	Yes	10
2	-10	0	2	Yes	5
3	0	70	1	Yes	1

Tab. 2.1: Example of histogram bins for frozen food

With this configuration the tag generates an alarm if the temperature has been sampled:

- between -15°C and -10°C for 3 times
- between -10 °C and 0 °C for 2 times
- greater than 0 °C

See Fig. 2.1and Fig. 2.2 for more details.

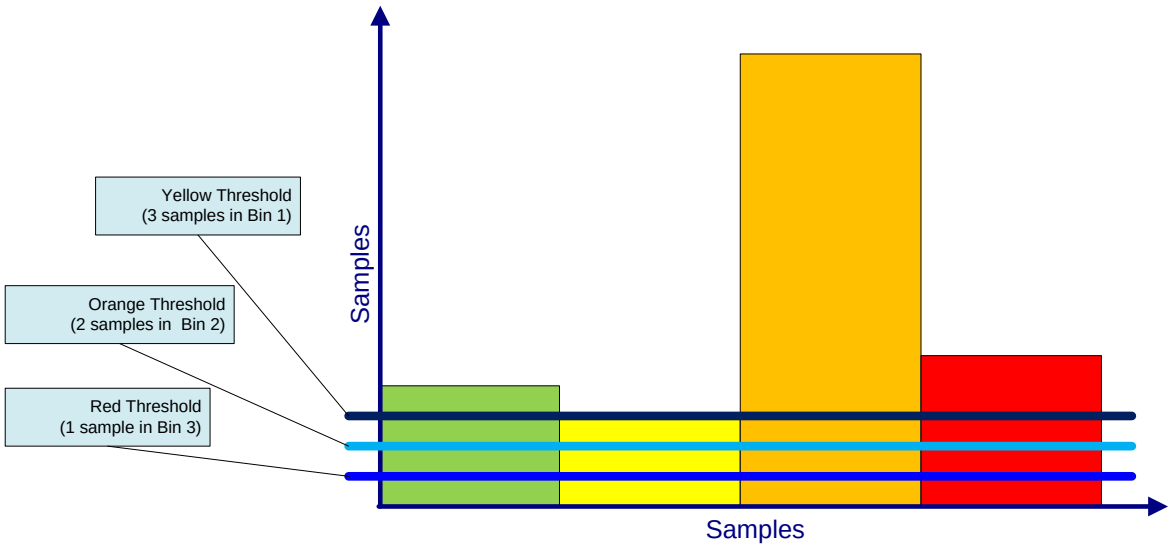


Fig. 2.1: Frozen example, time vs. histogram view

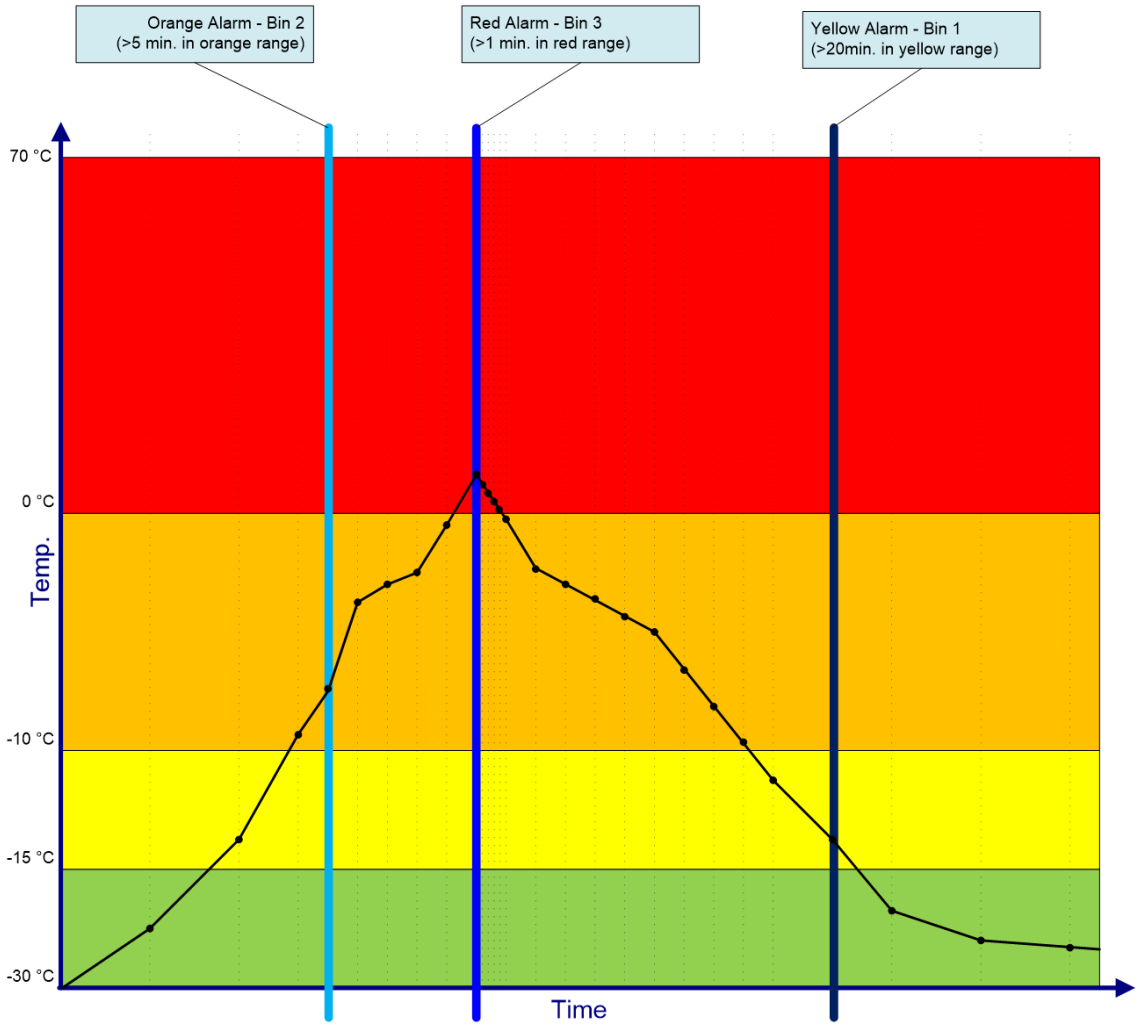


Fig. 2.2: Frozen example, time vs. temperature view

Example 2

Let suppose that the user wants to monitor the temperature trend for cooled food, we can define some histogram bins as follows:

Bin	Low Limit (°C)	High Limit (°C)	Threshold	Store Samples	Sampling Interval (min.)
0	-30	-5	5	Yes	5
1	-5	0	0xFFFF	No	10
2	0	70	3	Yes	5

Tab. 2.2: Example of histogram bins for cooled food

With this configuration the tag generates an alarm if the temperature has been sampled:

- between -30°C and -5°C for 5 times
- between 0 °C and 70 °C for 3 times
- greater than 0 °C

See Fig. 2.3 and Fig. 2.4 for more details.

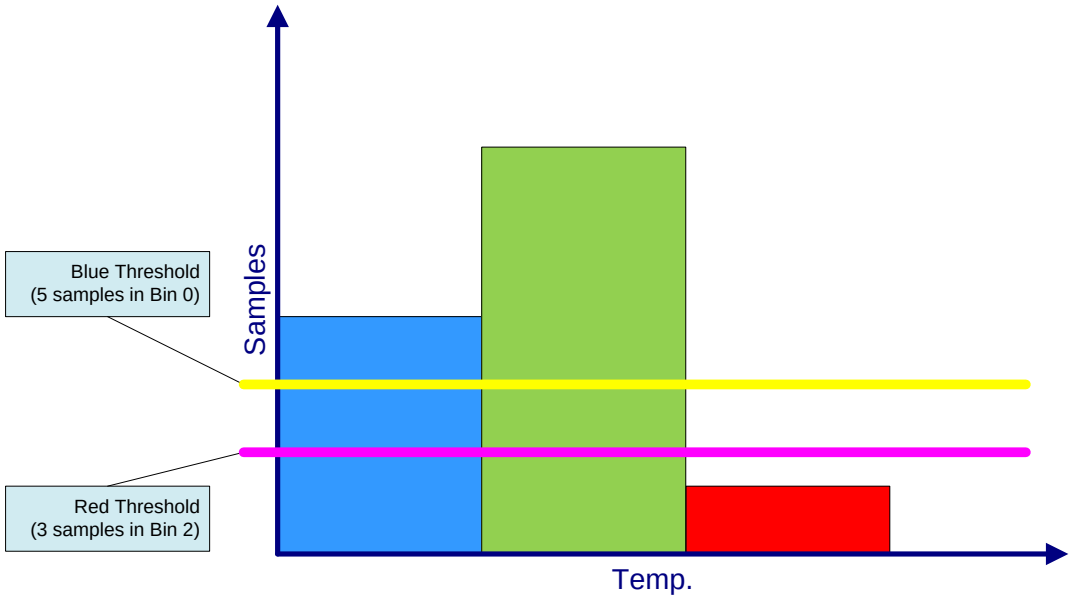


Fig. 2.3: Cooled example, histogram view

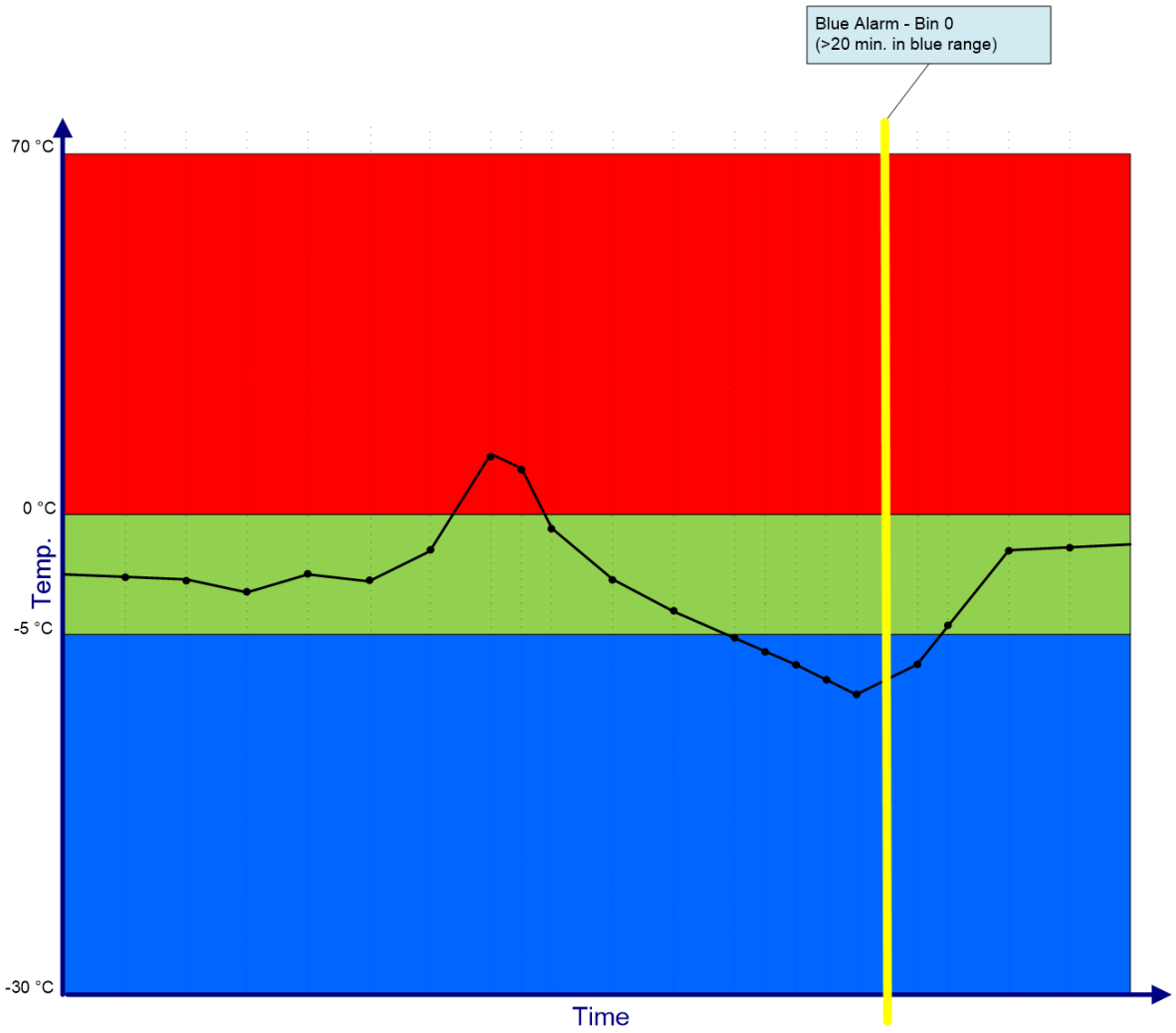


Fig. 2.4: Cooled example, time vs. temperature view

Battery measurement and behaviour

The RT0013 tag includes a battery measurement circuitry that allows to control the battery charge level.

Four battery charge levels are defined: Full (3), Normal (2), Low (1), Empty (0); values can be obtained reading the BAT_LSBIT and BAT_MSBIT in the STATUS register. The RT0013 tag behaves normally when the battery is at the Full, Normal or Low battery level, while it is automatically put in Sleep state when the level reach the Empty value.

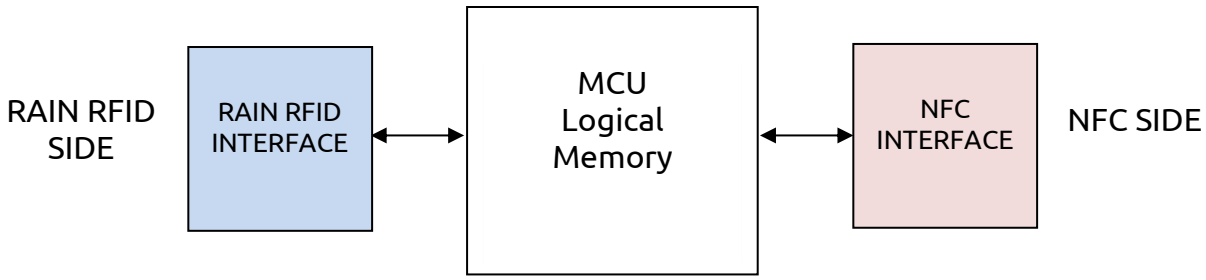
The battery level is measured every 15 minutes.

3 MEMORY DESCRIPTION

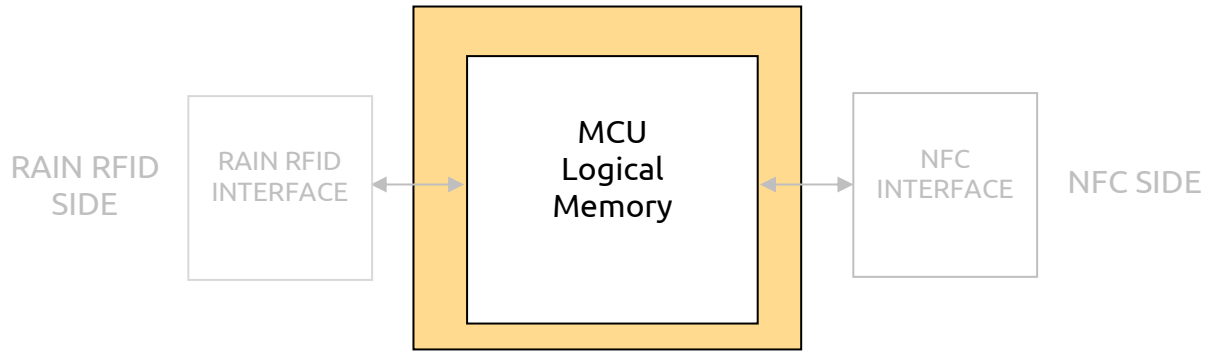
Introduction

The RT0013 logical registers and log data are located in the microcontroller memory (MCU) and are not directly accessible from the RAIN RFID (UHF EPC C1G2/ISO 18000-63) and NFC sides.

The access to these logical registers occurs indirectly: two exchange memories (RAIN RFID Interface and NFC interface) are used for the communication between the MCU and the RAIN RFID/NFC parts.



Logical Memory



The following table shows the logical memory map and its registers in the microcontroller memory (MCU):

Register	Address	Op.	Default	Reference
RFU	0x00	RO	-	Reserved for Future Use
RFU	0x01	RO	-	
RFU	0x02	RO	-	
RFU	0x03	RO	-	
RFU	0x04	RO	-	
RFU	0x05	RO	-	
RFU	0x06	RO	-	
RFU	0x07	RO	-	
FW_REVISION	0x08	RO	-	See § *FW_REVISION register (RO)* page *17*
HW_REVISION	0x09	RO	-	See § *HW_REVISION register (RO)* page *18*

Register	Address	Op.	Default	Reference
CONTROL	0x0A	RW	0x0000	See § *CONTROL register (RW)* page *18*
SAMPLING_DELAY	0x0B	RW	0x0000	See § *SAMPLING_DELAY register (RW)* page *18*
INIT_DATE_L	0x0C	RW	0x0000	See § *INIT_DATE register (RW- 2 words)* page *18*
INIT_DATE_H	0x0D	RW	0x0000	
ETA_L	0x0E	RW	0x0000	See § *ETA register (RW- 2 words)* page 19
ETA_H	0x0F	RW	0x0000	
BIN_ENABLE_COUNTER	0x10	RW	0x0041	See § *BIN_ENABLE_COUNTER register (RW)* page 19
BIN_ENA_SAMPLE_STORE	0x11	RW	0x0041	See § *BIN_ENA_SAMPLE_STORE register (RW)* page 19
BIN_ENA_TIME_STORE	0x12	RW	0x0000	See § *BIN_ENA_TIME_STORE register (RW)* page 19
BIN_HLIMIT_T_0	0x13	RW	0x08C0	See § *BIN_HLIMIT_T registers (RW – 6 words)* page 2020
BIN_HLIMIT_T_1	0x14	RW	0x0000	
...	...	..	0x0000	
BIN_HLIMIT_T_5	0x18	RW	0x0000	
BIN_HLIMIT_H_0	0x19	RW	0x0C80	See § *BIN_HLIMIT_H registers (RW – 6 words)* page 20
BIN_HLIMIT_H_1	0x1A	RW	0x0000	
...	...	..	0x0000	
BIN_HLIMIT_H_5	0x1E	RW	0x0000	
RFU	0x1F	-	-	Reserved for Future Use
RFU	0x20			
RFU	0x21			
RFU	0x22			
BIN_SAMPLETIME_T_0	0x23	RW	0x001E	See § *BIN_SAMPLETIME_T registers (RW – 6 words)* page 20
BIN_SAMPLETIME_T_1	0x24	RW	0x001E	
...	...	..	0x001E	
BIN_SAMPLETIME_T_5	0x28	RW	0x001E	
BIN_SAMPLETIME_H_0	0x29	RW	0x001E	See § *BIN_SAMPLETIME_H registers (RW – 6 words)* page 21
BIN_SAMPLETIME_H_1	0x2A	RW	0x001E	
...	...	..	0x001E	
BIN_SAMPLETIME_H_5	0x2E	RW	0x001E	
RFU	0x2F	-	-	Reserved for Future Use
RFU	0x30	-	-	
RFU	0x31	-	-	
RFU	0x32	-	-	
BIN_THRESHOLD_T_0	0x33	RW	0xFFFF	See § *BIN_THRESHOLD_T registers (RW – 6 words)* page 21
BIN_THRESHOLD_T_1	0x34	RW	0xFFFF	
...	...	..	0xFFFF	
BIN_THRESHOLD_T_5	0x38	RW	0xFFFF	
BIN_THRESHOLD_H_0	0x39	RW	0xFFFF	See § *BIN_THRESHOLD_H registers (RW – 6 words)* page 21
BIN_THRESHOLD_H_1	0x3A	RW	0xFFFF	
...	...	..	0xFFFF	
BIN_THRESHOLD_H_5	0x3E	RW	0xFFFF	
RFU	0x3F	-	-	Reserved for Future Use
RFU	0x40	-	-	
RFU	0x41	-	-	
RFU	0x42	-	-	
RFU	0x43	-	-	
RFU	0x44	-	-	
RFU	0x45	-	-	
RFU	0x46	-	-	
RFU	0x47	-	-	
RFU	0x48	-	-	
RFU	0x49	-	-	
RFU	0x4A	-	-	
RFU	0x4B	-	-	
RFU	0x4C	-	-	
RFU	0x4D	-	-	

Register	Address	Op.	Default	Reference
RFU	0x4E	-	-	
RFU	0x4F	-	-	
RFU	0x50	-	-	
STATUS	0x51	RO	-	See § *STATUS register (RO)* page *21*
RFU	0x52	RO	-	Reserved for Future Use
RFU	0x53	RO	-	
RFU	0x54	RO	-	
BIN_ALARM	0x55	RO	0x0000	See § *BIN_ALARM register (RO)* page 22
BIN_COUNTER_T_0	0x56	RO	0x0000	See § *BIN_COUNTER_T registers (RO - 6 words)* page 22
BIN_COUNTER_T_1	0x57	RO	0x0000	
...	...	..	0x0000	
BIN_COUNTER_T_5	0x5B	RO	0x0000	
BIN_COUNTER_H_0	0x5C	RO	0x0000	See § *BIN_COUNTER_H registers (RO - 6 words)* page 22
BIN_COUNTER_H_1	0x5D	RO	0x0000	
...	...	..	0x0000	
BIN_COUNTER_H_5	0x61	RO	0x0000	
LAST_SAMPLE_VALUE_T	0x62	RO	0x0000	See § *LAST_SAMPLE_VALUE_T register (RO)* page 22
LAST_SAMPLE_VALUE_H	0x63	RO	0x0000	See § *LAST_SAMPLE_VALUE_H register (RO)* page23
SAMPLES_NUM_T	0x64	RO	0x0000	See § *SAMPLES_NUM_T register (RO)* page 23
SAMPLES_NUM_H	0x65	RO	0x0000	See § *SAMPLES_NUM_H register (RO)* page 23
RFU	0x66	RO	-	Reserved for Future Use
RFU	0x67	RO	-	
RFU	0x68	RO	-	
RFU	0x69	RO	-	
SHIPPING_DATE_L	0x6A	RO	0x0000	See § *SHIPPING_DATE register (RO- 2 words)* page 23
SHIPPING_DATE_H	0x6B	RO	0x0000	
STOP_DATE _L	0x6C	RO	0x0000	See § *STOP_DATE register (RO- 2 words)* page 23
STOP_DATE _H	0x6D	RO	0x0000	
USER_AREA_0	0x6E	RW	-	See § *USER_AREA registers (RW - 28 words)* page 23
USER_AREA_1	0x6F	RW	-	
...	...	..	-	
USER_AREA_27	0x89	RW	-	
LOG_AREA_T_0	0x8A	RO	-	Logging area for temperature samples (2018 sample available, 4096 bytes)
LOG_AREA_T_1	0x8B	RO	-	
...	...	..	-	
LOG_AREA_T_2047	0x889	RO		
LOG_AREA_H_0	0x88A	RO	-	Logging area for humidity samples (2018 sample available, 4096 bytes)
LOG_AREA_H_1	0x88B	RO	-	
...	...	..	-	
LOG_AREA_H_2047	0x1089	RO		

Tab. 3.1: Internal Memory map

Register description

FW_REVISION register (RO)

Register	Address	Op.	Default
FW_REVISION	0x08	RO	-

This register contains the firmware release of the tag in hexadecimal format (Byte 1)(Byte 2)

where Byte 1 is the major version and Byte 2 is the minor version (e.g. 0x0103 corresponds to firmware release 1.3).

HW_REVISION register (RO)

Register	Address	Op.	Default
HW_REVISION	0x09	RO	-

This register contains the hardware release of the tag in hexadecimal format (Byte 1)(Byte 2)

where Byte 1 is the major version and Byte 2 is the minor version (e.g. 0x0103 corresponds to hardware release 1.3).

CONTROL register (RW)

Register	Address	Op.	Default
CONTROL	0x0A	RW	0x000

This register is composed by the following fields:

15..5	4	3	2	1	0
RFU	RFSL	DE	LE	RFU	RST

RST (RW) - Reset bit: Reset bit is used to erase all the log memory and reset all the counters included the histogram bins. The value of the bit remain set until the reset operation is performed then it returns to 0. Default value is 0.

LE (RW) - Logging_Enable bit: Logging_Enable bit is used to enable(1)/disable(0) the logging activity. Once stopped, the logging activity can be restarted only after the tag has been reset using the RST bit. Default value is 0.

DE (RW) - Delay_Enable bit: Delay_Enable bit is used to enable(1)/disable(0) the usage of the SAMPLING_DELAY register. If enabled, the first sample's acquisition is done SAMPLING_DELAY seconds after the start of the logging activity. Default value is 0.

RFSL (RW) - RF_Sensitivity_Level: Sensitivity_Level bit is used to enable(1)/disable (0) the High RF sensitivity option. Default value is 0 (Low RF sensitivity).

Warning: To increase the performance in terms of reading distance and read-write speed on the tag, set the *RF_Sensitivity_Level* to 1

SAMPLING_DELAY register (RW)

Register	Address	Op.	Default
SAMPLING_DELAY	0x0B	RW	0x0000

This register defines the delay, in seconds, of the first acquisition after the start of the logging activity. Register value range is 0 to 65535; default value is (seconds). The sampling delay is activated if the corresponding bit on the CONTROL register (DE) is enabled.

Warning: The minimum sampling unit is 5 seconds, so if the sampling delay is not a multiple of 5, it will be approximated to the next multiple of 5 (e.g. if sampling delay is set to 6 seconds, sampling occurs every 10 seconds).

INIT_DATE register (RW- 2 words)

Register	Address	Op.	Default
INIT_DATE_L	0x0C	RW	0x0000
INIT_DATE_H	0x0D	RW	0x0000

The *Init Date* register is used to keep track of time and it is expressed in Unix time format. Default register's value is 0 that corresponds to midnight (UTC) January, 1 1970. The INIT_DATE register MUST be set AFTER the RESET operation on the tag. INIT_DATE_L must be written before INIT_DATE_H. The INIT_DATE register is set to 0 after a reset operation.

ETA register (RW- 2 words)

Register	Address	Op.	Default
ETA_L	0x0E	RW	0x0000
ETA_H	0x0F	RW	0x0000

The *Estimated Time of Arrival* register (expressed in seconds) defines the maximum time, computed from shipping date, required to arrive at destination. If the logging is not stopped before the ETA is reached an alarm is generated. If the value of this register is 0 the ETA alarm feature is disabled. Default value of ETA register is 0.

BIN_ENABLE_COUNTER register (RW)

Register	Address	Op.	Default
BIN_ENABLE_COUNTER	0x10	RW	0x0041

The register is composed by the following fields:

15..12	11	…	6	5	…	0	
RFU	BIN05_H_EN	…	BIN00_H_EN	BIN05_T_EN	…	BIN00_T_EN	
--	0	0	1	0	0	1	**Default**

BINX_T_EN Enable(1)/Disable(0) the temperature counter for the corresponding histogram bin.

BINX_H_EN Enable(1)/Disable(0) the humidity counter for the corresponding histogram bin.

BIN_ENA_SAMPLE_STORE register (RW)

Register	Address	Op.	Default
BIN_ENA_SAMPLE_STORE	0x11	RW	0x0041

The register is composed by the following fields:

15..12	11	…	6	5	…	0	
RFU	BIN05_H_EN	…	BIN00_H_EN	BIN05_T_EN	…	BIN00_T_EN	
--	0	0	1	0	0	1	**Default**

BINX_T_EN: if the BINX_T_EN bit (X=0..5) is set to 1 all the temperature sampled data values which are in the range of bin X are saved into the logging memory.

BINX_H_EN: if the BINX_H_EN bit (X=0..5) is set to 1 all the humidity sampled data values which are in the range of bin X are saved into the logging memory.

BIN_ENA_TIME_STORE register (RW)

Register	Address	Op.	Default
BIN_ENA_TIME_STORE	0x12	RW	0x0000

The register is composed by the following fields:

15..12	11	…	6	5	…	0	
RFU	BIN05_H_EN	…	BIN00_H_EN	BIN05_T_EN	…	BIN00_T_EN	
--	0	0	0	0	0	0	**Default**

BINX_T_EN: if the BINX_T_EN bit (X = 0..5) is set to 1 the timestamps (2 words - Unix time format) of temperature sampled values which are in the range of bin X are saved into the logging memory. Default condition is 0 for all flags.

BINX_H_EN: if the BINX_H_EN bit (X = 0..5) is set to 1 the timestamps (2 words - Unix time format) of humidity sampled values which are in the range of bin X are saved into the logging memory. Default condition is 0 for all flags.

Example: In case both the sample value and the sample time need to be stored in the log area the sequence of the logged data in the memory is the following:

Address	Data (16 bit)
0x08A	Sample 1: Temperature value
0x08B	Sample 1: Time (low)(*)
0x08C	Sample 1: Time (high)(*)
0x08D	Sample 2: Temperature value
….	….

(*)Time is expressed in Unix format and need 2 words for a single sample.
Tab. 3.2: Temperature/Time memory organization

BIN_HLIMIT_T registers (RW – 6 words)

Register	Address	Op.	Default
BIN_HLIMIT_T_0	0x13	RW	0x08C0
BIN_HLIMIT_T_1	0x14	RW	0x0000
…	…	..	0x0000
BIN_HLIMIT_T_5	0x18	RW	0x0000

These registers are used to set the temperature upper limit for the corresponding histogram bin. The lower level corresponds to the upper level of the previous bin. The lower level of the first bin is fixed to the lower limit of the sensor.

The temperature T in °C is expressed in fixed point 8.5 notation $T_{fixedpoint}$ using the formula:

$$T_{fixedpoint} = \begin{cases} T * 32 & if\ 0°C \leq T \leq 70°C \\ 8192 + T * 32 & if\ -30°C \leq T \leq 0°C \end{cases}$$

Some sample values are shown in the table below:

T(°C)	$T_{fixedpoint}$
-30	7232
-0,25	8184
0	0
70	2240

BIN_HLIMIT_H registers (RW – 6 words)

Register	Address	Op.	Default
BIN_HLIMIT_H_0	0x19	RW	0x0C80
BIN_HLIMIT_H_1	0x1A	RW	0x0000
…	…	..	0x0000
BIN_HLIMIT_H_5	0x1E	RW	0x0000

These registers are used to set the humidity upper limit for the corresponding histogram bin. The lower level corresponds to the upper level of the previous bin. The lower level of the first bin is fixed to the lower limit of the sensor.

The humidity H in % is expressed in fixed point 8.5 notation $H_{fixedpoint}$ using the formula:

$$H_{fixedpoint} = H_{\%} * 32 \quad (0 \leq H_{\%} \leq 100)$$

BIN_SAMPLETIME_T registers (RW – 6 words)

Register	Address	Op.	Default
BIN_SAMPLETIME_T_0	0x23	RW	0x001E
BIN_SAMPLETIME_T_1	0x24	RW	0x001E
…	…	..	0x001E
BIN_SAMPLETIME_T_5	0x28	RW	0x001E

These registers are used to define the sample interval to apply for the next acquisitions in the corresponding bin. The value is expressed in seconds.

Warning: The minimum sampling unit is 5 seconds, so if the sample interval is not a multiple of 5, it will be approximated to the next multiple of 5 (e.g. if sample interval is set to 6 seconds, sampling occurs every 10 seconds).

BIN_SAMPLETIME_H registers (RW – 6 words)

Register	Address	Op.	Default
BIN_SAMPLETIME_H_0	0x29	RW	0x001E
BIN_SAMPLETIME_H_1	0x2A	RW	0x001E
...	...	..	0x001E
BIN_SAMPLETIME_H_5	0x2E	RW	0x001E

These registers are used to define the sample interval to apply for the next acquisitions in the corresponding bin. The value is expressed in seconds.

Warning: The minimum sampling unit is 5 seconds, so if the sample interval is not a multiple of 5, it will be approximated to the next multiple of 5 (e.g. if sample interval is set to 6 seconds, sampling occurs every 10 seconds).

BIN_THRESHOLD_T registers (RW – 6 words)

Register	Address	Op.	Default
BIN_THRESHOLD_T_0	0x33	RW	0xFFFF
BIN_THRESHOLD_T_1	0x34	RW	0xFFFF
...	...	..	0xFFFF
BIN_THRESHOLD_T_5	0x38	RW	0xFFFF

These registers represent the threshold value for the alarm (expressed in counts) on the corresponding bin. If the bin counter value is greater or equal than the threshold value the alarm bit for the corresponding bin is set.

BIN_THRESHOLD_H registers (RW – 6 words)

Register	Address	Op.	Default
BIN_THRESHOLD_H_0	0x39	RW	0xFFFF
BIN_THRESHOLD_H_1	0x3A	RW	0xFFFF
...	...	..	0xFFFF
BIN_THRESHOLD_H_5	0x3E	RW	0xFFFF

These registers represent the threshold value for the alarm (expressed in counts) on the corresponding bin. If the bin counter value is greater or equal than the threshold value the alarm bit for the corresponding bin is set.

STATUS register (RO)

Register	Address	Op.	Default
STATUS	0x51	RO	--

The register is composed by the following fields:

15..12	11	10	9..5	4	3	2	1	0
RFU	BIN_ALRM_H	MEMFULL_H	RFU	BIN_ALRM_H	ETA_ALRM	MEMFULL_T	BAT_MSBIT	BAT_LSBIT

BAT_LS/MS_BIT - Battery_Level: Battery Level is a 2 bit register representing the remaining battery capacity. Value of 3 means full capacity, value 0 means minimum capacity. The measurement is performed every 15 minutes. If the Battery Level value is 0(almost discharged) the RT0013 is put automatically in Sleep state.

MEMFULL_T - Temperature_Memory_Full bit: Temperature Memory Full bit represents the status of the temperature log memory, its value is 0 if the temperature memory is not yet full, it is 1 if there is no more space in the temperature log memory. If both temperature and humidity memory become full the RT0013 is put automatically in Sleep state. Default value is 0. MEMFULL_T bit is set to 0 after a reset operation.

ETA_ALRM - ETA_Alarm bit: the ETA_Alarm bit is set to 1 when the current time exceeds the value of the SHIPPING_DATE register plus the ETA delay register. Default value is 0. ETA_ALRM bit is set to 0 after a reset operation.

BIN_ALRM_T - Temperature_Histogram_Alarm bit: the Temperature Histogram Alarm bit is set to 1 when at least one of the temperature histogram bin counters is greater than the corresponding threshold. Default value is 0. BIN_ALRM bit is set to 0 after a reset operation.

MEMFULL_H - Humidity_Memory_Full bit: Humidity Memory Full bit represents the status of the Humidity log memory, its value is 0 if the Humidity memory is not yet full, it is 1 if there is no more space in the Humidity log memory. If both temperature and humidity memories become full the RT0013 is put automatically in Sleep state. Default value is 0. MEMFULL_T bit is set to 0 after a reset operation.

BIN_ALRM_H - Humidity_Histogram_Alarm bit: the Humidity Histogram Alarm bit is set to 1 when at least one of the Humidity histogram bin counters is greater than the corresponding threshold. Default value is 0. BIN_ALRM_H bit is set to 0 after a reset operation.

BIN_ALARM register (RO)

Register	Address	Op.	Default
BIN_ALARM	0x55	RO	0x0000

The register is composed by the following fields:

15	…	2	1	0	
BIN15_EN	…	BIN02_EN	BIN01_EN	BIN00_EN	
0	0	0	0	0	**Default**

BINX_EN bit (X = 0..15) is set to 1 if the value of the BIN_COUNTER_X register is greater than the corresponding BIN_THRESHOLD_X register value. BIN_ALRM register is set to 0 after a reset operation.

BIN_COUNTER_T registers (RO - 6 words)

Register	Address	Op.	Default
BIN_COUNTER_T_0	0x56	RO	0x0000
BIN_COUNTER_T_1	0x57	RO	0x0000
...	...	..	0x0000
BIN_COUNTER_T_5	0x5B	RO	0x0000

These registers represent the number of temperature samples in the range of the corresponding bin. Default value is 0 for each counter. All the BIN_COUNTER registers are set to 0 after a reset operation.

BIN_COUNTER_H registers (RO - 6 words)

Register	Address	Op.	Default
BIN_COUNTER_H_0	0x5C	RO	0x0000
BIN_COUNTER_H_1	0x5D	RO	0x0000
...	...	..	0x0000
BIN_COUNTER_H_5	0x61	RO	0x0000

These registers represent the number of temperature samples in the range of the corresponding bin. Default value is 0 for each counter. All the BIN_COUNTER registers are set to 0 after a reset operation.

LAST_SAMPLE_VALUE_T register (RO)

Register	Address	Op.	Default
LAST_SAMPLE_VALUE_T	0x62	RO	0x0000

It represents the last temperature value read from the sensor (even if not stored in the log memory). LAST_SAMPLE_VALUE_T register is set to 0 after a reset operation.

LAST_SAMPLE_VALUE_H register (RO)

Register	Address	Op.	Default
LAST_SAMPLE_VALUE_H	0x63	RO	0x0000

It represents the last humidity value read from the sensor (even if not stored in the log memory). LAST_SAMPLE_VALUE_H register is set to 0 after a reset operation.

SAMPLES_NUM_T register (RO)

Register	Address	Op.	Default
SAMPLES_NUM_T	0x64	RO	0x0000

It represents the number of temperature samples present inside the log memory. SAMPLES_NUM_T register is set to 0 after a reset operation.

SAMPLES_NUM_H register (RO)

Register	Address	Op.	Default
SAMPLES_NUM_H	0x65	RO	0x0000

It represents the number of humidity samples present inside the log memory. SAMPLES_NUM_H register is set to 0 after a reset operation.

SHIPPING_DATE register (RO- 2 words)

Register	Address	Op.	Default
SHIPPING_DATE_L	0x6A	RO	0x0000
SHIPPING_DATE_H	0x6B	RO	0x0000

The Shipping Date register (expressed in the Unix time format) is used to store the time when logging was enabled (by RF or, if enabled, by the button). Default value of SHIPPING_DATE register is 0. SHIPPING_DATE register is set to 0 after a reset operation.

STOP_DATE register (RO- 2 words)

Register	Address	Op.	Default
STOP_DATE _L	0x6C	RO	0x0000
STOP_DATE _H	0x6D	RO	0x0000

The STOP_DATE register (expressed in the Unix time format) is used to store the time when logging was stopped (by RF or, if enabled, by the button). Default value of STOP_DATE register is 0. STOP_DATE register is set to 0 after a reset operation.

USER_AREA registers (RW - 28 words)

Register	Address	Op.	Default
USER_AREA_0	0x6E	RW	--
USER_AREA_1	0x6F	RW	--
...	..	..	--
USER_AREA_27	0x89	RW	--

User area is a set of 28 words registers for user defined data storage.

Registers use example

Consider the example 1 in *Examples of usage* paragraph page 10: in the table below is presented again the scenario that shall be realized:

Bin	Low Limit (°C)	High Limit (°C)	Threshold	Store Samples	Sampling Interval (min.)
0	-30	-15	0xFFFF	Yes	15
1	-15	-10	3	Yes	10
2	-10	0	2	Yes	5
3	0	70	1	Yes	1

First of all let's set the upper limit of each histogram bin. The upper limits in the example are -15, -10, 0 and 70 °C. The data need to be converted in fixed point 8.5 notation format using the formula (3):

T(°C)	T_{fixed}	$Register_{hex}$
-15	7712	HLIMIT[0] = 0x1E20
-10	7872	HLIMIT[1] = 0x1EC0
0	0	HLIMIT[2] = 0x0000
70	2240	HLIMIT[3] = 0x08C0

All the 4 Bins used in the example need to be enabled in the BIN_ENABLE_COUNTER register, the bits corresponding shall be set to 1:

BIN_ENABLE_COUNTER = 0x000F

The Threshold value register shall be filled with the example's value:

BIN_THRESHOLD[0] = 0x0000

BIN_THRESHOLD[1] = 0x0003

BIN_THRESHOLD[2] = 0x0002

BIN_THRESHOLD[3] = 0x0001

The sampling interval time, expressed in minutes, must be converted in seconds for the correct setting of the BIN_SAMPLETIME register:

Min.	Sec.	Register
15	900	BIN_SAMPLETIME[0] = 0x0384
10	600	BIN_SAMPLETIME[1] = 0x0258
5	300	BIN_SAMPLETIME[2] = 0x012C
1	60	BIN_SAMPLETIME[3] = 0x003C

In this example, the samples have to be stored in the user memory hence the BIN_SAMPLE_STORE register flags have to be set for the corresponding bins:

BIN_SAMPLE_STORE = 0x000F

The samples' timestamps do not need to be saved (it is not required by the example) hence the BIN_SAMPLE_TIME register's flags shall be disabled:

BIN_SAMPLE_TIME = 0x0000

In order to activate the tag and put it in *Idle state* the tag shall be reset using the dedicated bit in the CONTROL register and then the initial date configuration register (INIT_DATE) shall be written with a non-zero value. In this state the internal clock is active and the tag is waiting for a start logging command (LE in the STATUS register).

Once the LE bit toggle to 1 (start logging) the tag waits for the SAMPLING_DELAY expiration (if the Delay Enable flag is enabled and SAMPLING_DELAY is different than 0) then it starts the acquisition of the temperature samples.

During the acquisition, the tag samples the temperature each BIN_SAMPLETIME[x] seconds (where x is the bin index corresponding to the last temperature sample).

Let now suppose that the temperature has been sampled more than 3 times in the first bin; in this case an alarm must be generated. The status of the register will be:

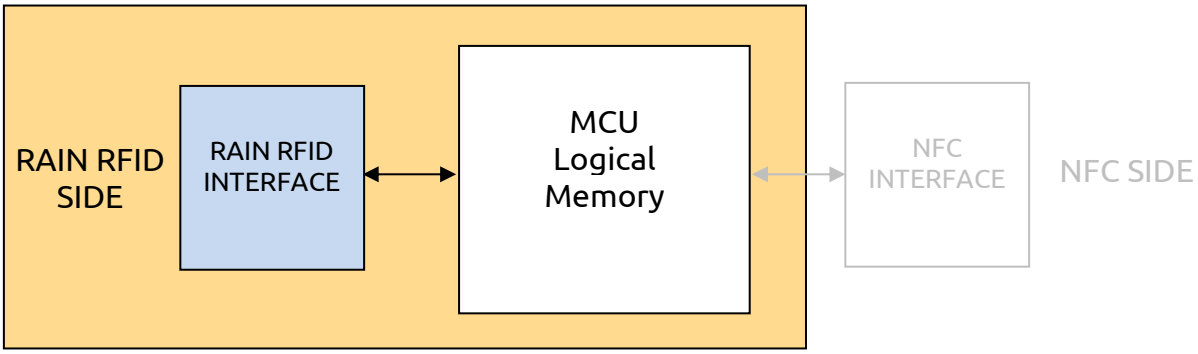
STATUS	**= 0x0013**	(BIN_ALARM flag active and battery status Full)
BIN_ALARM	**= 0x0002**	(alarm flag set corresponding to the first bin).
BIN_COUNTER[1]	**= 0x0003**	(3 samples counted in the first bin)

4 RAIN RFID COMMUNICATION PROTOCOL

Introduction

The RT0013 logical registers and log data are located in the microcontroller memory (MCU) and are not directly accessible from the RAIN RFID side using the standard RAIN RFID read/write commands.

The communication with the MCU occurs using an exchange memory that is the RAIN RFID INTERFACE for the RAIN RFID side:



This exchange memory allows the user to access the logical registers using a communication protocol described in this chapter.

RAIN RFID Memory

The RAIN RFID Interface is the UHF Memory. The RT0013 tag implements a standard Gen2 memory map: it has 4 memory banks, RESERVED, EPC, TID and USER.

Memory Banks are defined as follows:

Bank 11b	USER
Bank 10b	TID
Bank 01b	EPC
Bank 00b	RESERVED

Tab. 4.1: Standard Gen2 memory map

Below a brief explanation of memory banks content (all the address are specified 16bit word bounded).

RESERVED

This bank contains the 32 bit *Kill password* and the 32 bit *Access password* according to the EPC Class 1 Gen 2 standard.

EPC

Word Address	Contents
0x00	CRC-16
0x01	16 bit PC
0x02 – 0x07	96 bit EPC*
0x08 - 0x1E	RFU
0x1F	(interface) TRIGGER REGISTER
0x20	RFU

* The EPC bank contains the EPC of the RT0013 product that is composed by the CAEN RFID company prefix, the RT0013 object type and a serial number. The EPC is formatted as a SGTIN-96 code.

TID

Word Address	Contents
0x00-0x01	TID Header
0x02	XTID Header
0x03-0x05	TID serial number

USER

Word Address	Contents
0x00	(interface) COMMAND REGISTER
0x01	(interface) ADDRESS REGISTER
0x02	(interface) SIZE REGISTER
0x03	(interface) REPLY REGISTER
0x04 - 0xCB	(interface) DATA REGISTERs
0xCC - 0xCF	RFU

RAIN RFID Registers description

Using the interface registers (COMMAND, ADDRESS, SIZE, REPLAY, DATA and TRIGGER) of the RAIN RFID exchange memory, the user can perform any operation on the tag:

1. Using the COMMAND register (user memory) the user specifies the operation to perform (READ or WRITE) and assigns an identifier (ID) to this operation.
2. Using the ADDRESS register (user memory) the user specifies at which logical address it wants to operate.
3. Using the SIZE register (user memory) the user indicates the number of registers affected by the command, starting from the one specified in the ADDRESS register.
4. In the DATA register the user specifies the values to write in the logical registers (in case of WRITE operation).
5. Using the TRIGGER register (EPC memory) the user warns the microcontroller that there are operations to perform.
6. The reader waits for the replies and then searches for the ID of the required operation inside the REPLY register and reads the content of the DATA register to retrieve the required information in case of a READ operation.

For more details, refer to the registers description below and to the *Communication Protocol RAIN RFID* Side paragraph page 29.

Warning: To increase the efficiency of read and write operations on the tag, it is possible to select several registers simultaneously using the SIZE register

COMMAND register (RW)

Register	Address	Op.	Bank
COMMAND	0x00	RW	USER

COMMAND is a 16 bit word register used to select between a READ or a WRITE operation of the tag internal registers or log area. This register is composed by the ID and CMD fields.

15...8	7...0
ID	CMD

ID: Message Identifier. This value, chosen by the user, will be echoed by the tag in its subsequent reply.

CMD: The command to be executed. Allowed values are the following:

Command	Value	Description
CMD_READ	0x12	performs a read operation from internal registers or from the log area.
CMD_WRITE	0x13	performs a write operation to the tag internal registers.

ADDRESS register (RW)

Register	Address	Op.	Bank
ADDRESS	0x01	RW	USER

ADDRESS is a 16-bit register used to store the address of the internal register or log data the READ/WRITE operation is to be performed from/on. If the READ/WRITE operation comprises two or more consecutive internal registers or data, the READ/WRITE will start from the register address stored in ADDRESS.

SIZE register (RW)

Register	Address	Op.	Bank
SIZE	0x02	RW	USER

SIZE is a 16-bit register used to store the number of registers or 16-bit word data to be read/written from the tag.

Warning: To increase the efficiency of read and write operations on the tag, it is possible to select several registers simultaneously using the SIZE register

REPLY register (RW)

Register	Address	Op.	Bank
REPLY	0x03	RW	USER

REPLY is a 16-bit register used by the tag to return the result of the operation it performed after receiving a read/write command. This register is composed by following fields:

15...8	7...0
ID	REPLY

ID: Last message identifier. It should match the COMMAND register ID value.

REPLY: Command return value. Allowed values are:

- ACK (0xAC): Command was successfully executed.
- NACK (0xFC): Command failed/Invalid parameter.

DATA Area (RW)

Register	Address	Op.	Bank
DATA	0x04 - 0xCB	RW	USER

The DATA 16-bit registers contain either the data/internal registers read from the tag or the values to be written:

- when performing a CMD_WRITE command of *size* (value stored in SIZE register) 16-bit words, starting from address *address* (value stored in ADDRESS register), the user will have to load size words in the DATA area (starting from USER bank address DATA_AREA_START = 0x04 – the start of DATA area). When performing the WRITE, the word written at address DATA_AREA_START of the USER bank will be written by the tag at address *address* of its internal registers, the word written at address (DATA_AREA_START + 1) of the USER bank will be written by the tag at address (*address* + 1) of its internal registers,...., the word written at address (DATA_AREA_START + size - 1) of the USER bank will be written by the tag at address (*address* + *size* - 1) of its internal registers.
- when executing a CMD_READ command of *size* (value stored in SIZE register) 16-bit words, starting from address *address* (value stored in ADDRESS register), the tag will load in the DATA area the values of the read internal registers/data. After a successful command execution, the word at address DATA_AREA_START of the USER memory bank will contain the value of the internal register/data found at address *address*,...,the word at address (DATA_AREA_START + *size* - 1) of the USER bank will contain the value of the internal register/data found at address (*address* + *size* – 1).

RFU registers (R)

Register	Address	Op.	Bank
RFU	0xCC-0xCF	R	USER

The RFU registers are 16-bit Reserved registers and should not be written.

TRIGGER register (R)

Register	Address	Op.	Bank
TRIGGER	0x1F	R	EPC

The 16-bit TRIGGER register is used to start the execution of a command. After loading the COMMAND, ADDRESS, SIZE, and optionally DATA area registers, in order to start the command execution the user must perform an EPCC1G2 READ operation of the TRIGGER address in the EPC memory bank.

Communication Protocol RAIN RFID Side

To read/write data into the tag's logical registers the user will have to perform the following steps:

1. Initialize the COMMAND, ADDRESS, SIZE register as required to perform the desired read/write operation. In case of a *CMD_WRITE* operation fill the DATA area with the values to be written into the tag's internal memory.
2. Start the command execution by a single read operation of the TRIGGER register in the EPC memory bank.
3. Wait for the tag to execute command.
4. Read the REPLY register. If the ID field in the reply register matches the ID field of the COMMAND register then command execution is completed. If the ID is different return to step 3.
5. Check the value of the REPLY field of the REPLY register. If the REPLY field value is an ACK, the tag performed successfully the requested operation: in case of a CMD_READ command move to next step, otherwise command execution is completed, tag is ready to receive another command. If the REPLY field value is a NACK, either one or more parameters are invalid, or the operation failed.
6. If a CMD_READ command was performed, read the DATA area of the USER bank to retrieve the values of the requested internal registers/log data.

Warning: The RAIN RFID shared interface memory cannot be accessed by the user (RAIN RFID side) and by the tag at the same time. In fact, the RAIN RFID reader and the tag attempting to access the interface memory at the same time could cause the failure of read/write operations to the interface, and to subsequent command executions. For this reason, there are timing constraints and general rules that must be adhered to when performing a communication with the tag:

1. Avoid performing EPCC1G2 reads to the TRIGGER register when not needing to trigger a command execution.
2. Avoid to attempt performing EPCC1G2 writes to the TRIGGER register.
3. To trigger a command execution, a single EPCC1G2 read of the TRIGGER register is enough. Do not perform more consecutive reads.
4. After triggering a command execution, before checking the REPLY register for a tag reply give the tag enough time to perform the desired operation. Usually:
   - for a CMD_WRITE command the waiting time is constant T ~ 400 msec
   - for a CMD_READ command the waiting time depends on the size of internal registers/ 16-bit word data to be read.

     A good approximation is T ~ 100 msec + 7 msec * (size/2 + 1)
5. If, after waiting enough time for an operation execution, a check of the REPLY register results in the tag not having yet completed the required operation and replied, avoid re-triggering the command execution right away. Please wait again and check again the REPLY register afterwards.

Due to the wait timing constraints, when needing to perform read or write operations on internal registers/data whose addresses are consecutive, the user can speed up significantly the execution by accessing internal registers in blocks and not individually, through the use of the SIZE register.

Warning: The use of the *Communication Protocol* on the RAIN RFID side is the same as the one on the NFC side, with the difference that **to trigger a command execution, a READ operation is to be performed on the TRIGGER register instead of a WRITE.**

CAEN RFID SDK UHF Protocol Sample Functions

This section shows a C# example of protocol functions implementation using the CAENRFID SDK.

```
public static byte[] TagReadRegisters(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag, short byteaddress, short
numreg)
{
	const int TIME_WAITTAG_CMDREADBASE = 100;
	const int TIME_WAITTAG_WRITEPAGE = 7;
	byte idmsg = 0;
	short command;
	short numbytes = (short) (numreg * 2);
	byte reply = INTERFACEMEM.REPLY_NACK;

	//check parameters
	if (byteaddress % 2 != 0)
	{
		throw new Exception("Access at byte level not allowed");
	}
	if (numbytes > INTERFACEMEM.MAXBYTESIZEDATA)
	{
	throw new Exception("Invalid size");
	}

	//check current idmsg value written in reply word
	//and adjust idmsg of next command accordingly
	if (LS_0.ReadTagData_EPC_C1G2(tag, INTERFACEMEM.CMDBANK, INTERFACEMEM.ADDR_REPLY, 2)[0] == idmsg) idmsg++;
	command = (short)(idmsg << 8 | INTERFACEMEM.CMD_READ);

	//load command parameters in user memory
	INTERFACEMEM.interfacemem_writeparams(LS_0, tag, command, (short)(byteaddress / 2), (short)(numbytes /
	2));
	//trigger tag command reception+execution
	INTERFACEMEM.interfacemem_trigger(LS_0, tag);

	//wait for tag to parse command, execute it, and reply
	Thread.Sleep(TIME_WAITTAG_CMDREADBASE + TIME_WAITTAG_WRITEPAGE * (numbytes / 4 + 1));

	//check if tag replied
	byte[] buff = INTERFACEMEM.interfacemem_readreply(LS_0, tag);
	if (buff[0] == idmsg)
	{
		reply = buff[1];
		if (!((reply == INTERFACEMEM.REPLY_ACK) || (reply == INTERFACEMEM.REPLY_NACK)))
		{
			throw new Exception("Tag reply time out");
		}
	}
	else
	{
		throw new Exception("Tag reply time out");
	}
	//check reply
	if (reply == INTERFACEMEM.REPLY_NACK)
	{
		throw new Exception("Tag replied NACK");
	}

	//tag replied ACK, now we can read the data
	byte[] data = INTERFACEMEM.interfacemem_readdata(LS_0, tag, 0, numbytes);
	return data;
}

public static void TagWriteRegisters(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag, short byteaddress, short
numreg, byte[] data)
{
	const int TIME_WAITTAG_CMDWRITE = 400;
	byte idmsg = 0;
	short command;
	short numbytes = (short)(numreg * 2);
	byte reply = INTERFACEMEM.REPLY_NACK;

	//check parameters
	if (byteaddress % 2 != 0)
	{
		throw new Exception("Access at byte level not allowed");
	}
	if (numbytes > INTERFACEMEM.MAXBYTESIZEDATA)
	{
		throw new Exception("Invalid size");
	}

	//check current idmsg value written in reply word
	//and adjust idmsg of next command accordingly
	if (LS_0.ReadTagData_EPC_C1G2(tag, INTERFACEMEM.CMDBANK, INTERFACEMEM.ADDR_REPLY, 2)[0] == idmsg) idmsg++;
```

```
            command = (short)(idmsg << 8 | INTERFACEMEM.CMD_WRITE);

            //load command parameters in user memory
            INTERFACEMEM.interfacemem_writeparams(LS_0, tag, command, (short)(byteaddress / 2), (short)(numbytes /
            2));

            //load data to be written
            INTERFACEMEM.interfacemem_writedata(LS_0, tag, 0, numbytes, data);

            //trigger tag command reception+execution
            INTERFACEMEM.interfacemem_trigger(LS_0, tag);

            //wait for tag to parse command, execute it, and reply
            Thread.Sleep(TIME_WAITTAG_CMDWRITE);

            //check if tag replied
            byte[] buff = INTERFACEMEM.interfacemem_readreply(LS_0, tag);
            if (buff[0] == idmsg)
            {
                    reply = buff[1];
                    if (!((reply == INTERFACEMEM.REPLY_ACK) || (reply == INTERFACEMEM.REPLY_NACK)))
                    {
                            throw new Exception("Tag reply time out");
                    }
            }
            else
            {
                    throw new Exception("Tag reply time out");
            }
            //check reply
            if (reply == INTERFACEMEM.REPLY_NACK)
            {
                    throw new Exception("Tag replied NACK");
            }
    }

    private static class INTERFACEMEM
    {
            public readonly static short CMDBANK = 3;
            public readonly static short TRIGBANK = 1;
            public readonly static short ADDR_TRIGGER = 0x001F * 2; //byte address
            public readonly static short ADDR_COMMAND = 0x0000 * 2;
            public readonly static short ADDR_ADDRESS = (short)(ADDR_COMMAND + 2);
            public readonly static short ADDR_SIZE = (short)(ADDR_ADDRESS + 2);
            public readonly static short ADDR_REPLY = (short)(ADDR_SIZE + 2);
            public readonly static short ADDR_DATA = (short)(ADDR_REPLY + 2);
            public readonly static short MAXBYTESIZEDATA = 200 * 2;
            public readonly static byte CMD_READ = 0x12;
            public readonly static byte CMD_WRITE = 0x13;
            public readonly static byte REPLY_ACK = 0xAC;
            public readonly static byte REPLY_NACK = 0xFC;

            /* This function loads the command, address, and size parameters in the corresponding registers
            * of tag memory interface. */
            public static void interfacemem_writeparams(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag, short command,
            short wordaddress, short words)
            {
                    byte[] data = new byte[6];

                    data[0] = (byte)(command >> 8);
                    data[1] = (byte)(command & 0xFF);
                    data[2] = (byte)(wordaddress >> 8);
                    data[3] = (byte)(wordaddress & 0xFF);
                    data[4] = (byte)(words >> 8);
                    data[5] = (byte)(words & 0xFF);

                    try
                    {
                            LS_0.WriteTagData_EPC_C1G2(tag, CMDBANK,ADDR_COMMAND, (short)data.Length, data);
                    }
                    catch (Exception ex)
                    {
                            throw (ex);
                    }
            }

            /* This function triggers the tag parsing and execution of a command */
            public static void interfacemem_trigger(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag)
            {
                    try
                    {
                            LS_0.ReadTagData_EPC_C1G2(tag, TRIGBANK, ADDR_TRIGGER, (short)4);
                    }
                    catch (Exception ex)
                    {
                            throw (ex);
                    }
```

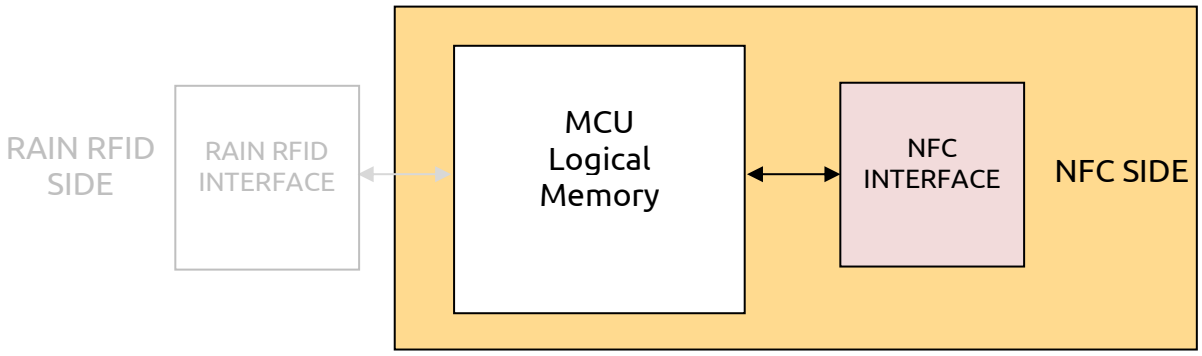
```
	}

	/* This function loads the data parameters in the DATA area of the tag memory interface. */
	public static void interfacemem_writedata(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag, short address,
	short bytes, byte[] data)
	{
		try
		{
			LS_0.WriteTagData_EPC_C1G2(tag, CMDBANK, (short)(ADDR_DATA + address), bytes, data);
		}
		catch (Exception ex)
		{
			throw (ex);
		}
	}

	/* This function reads the DATA area of the tag memory interface. */
	public static byte[] interfacemem_readdata(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag, short address,
	short bytes)
	{
		byte[] data;

		try
		{
			data = LS_0.ReadTagData_EPC_C1G2(tag, CMDBANK, (short)(ADDR_DATA + address), bytes);
			return data;
		}
		catch (Exception ex)
		{
			throw (ex);
		}
	}

	/* This function reads the REPLY register of the tag memory interface */
	public static byte[] interfacemem_readreply(CAENRFIDLogicalSource LS_0, CAENRFIDTag tag)
	{
		byte[] data;

		try
		{
			data = LS_0.ReadTagData_EPC_C1G2(tag, CMDBANK, ADDR_REPLY, 2);
			return data;
		}
		catch (Exception ex)
		{
			throw (ex);
		}
	}
}
```

5 NFC COMMUNICATION PROTOCOL

Introduction

The RT0013 logical registers and log data are located in the microcontroller memory (MCU) and are not directly accessible from the NFC side.

The communication with the MCU occurs using an exchange memory that is the NFC INTERFACE for the NFC side:



This exchange memory allows the user to access the logical registers using a communication protocol described in this chapter.

NFC Memory

The following figure shows the structure of the RT0013 tag NFC memory (as seen from the RF side):

SEGMENT 0 (COMPATIBILITY SEGMENT)	UID CL 1	
	UID CL 2	
	STATIC LOCK BITS	
	OTP BITS	
SEGMENT 1 (OPEN USER DATA)	OPEN USER DATA	AUTH_DATA
SEGMENT 2 (PROTECTED USER DATA)	PROTECTION USER DATA	DATA_END
SEGMENT 3 (DYN. LOCK BITS)	DYNAMIC LOCK BITS	
SEGMENT 4 (USER CONFIGURATION)	USER CONFIGURATION	END_MEM
EMPTY SEGMENT	EMPTY	
SEGMENT 5 (HIDDEN)	NXP CONFIGURATION AND INTERNAL DATA	

←32 bits wide→

Legend:

R ACCESS

RW ACCESS

PASSWORD NEEDED

NO ADDRESS

Tab. 5.1: NFC memory map as seen from RF interface

The memory is structured in 32-bit pages accessible through the READ/WRITE Mifare Ultralight commands.

The USER DATA area is the relevant part of memory and is structured as follows:

Address	Contents	Op.	Memory Type
0x04	(interface) COMMAND REGISTER	RW	Volatile
0x05	(interface) ADDRESS REGISTER	RW	Volatile
0x06	(interface) SIZE REGISTER	RW	Volatile
0x07	(interface) REPLY REGISTER	RW	Volatile
0x08 - 0x6B	(interface) DATA REGISTERs	RW	Volatile
0x85	(interface) TRIGGER REGISTER	W	Volatile
0x87 - 0x89	Tag identifier 96 bit EPC*		Not Volatile

Tab. 5.2: Mapping of protocol registers to NFC memory

* The EPC bank contains the EPC of the RT0013 product that is composed by the CAEN RFID company prefix, the RT0013 object type and a serial number. The EPC is formatted as a SGTIN-96 code.

Warning: written values inside volatile memory are kept as long as the tag detects the NFC field and are lost once the NFC field is turned off or lost.

Using the interface registers (COMMAND, ADDRESS, SIZE, REPLAY, DATA and TRIGGER) of the NFC exchange memory, the user can perform any operation on the tag:

1. Using the COMMAND register the user specifies the operation to perform (READ or WRITE) and assigns an identifier (ID) to this operation.
2. Using the ADDRESS register the user specifies at which logical address it wants to operate.
3. Using the SIZE register the user indicates the number of registers affected by the command, starting from the one specified in the ADDRESS register.
4. In the DATA register the user specifies the values to write in the logical registers (in case of WRITE operation).
5. Using the TRIGGER register the user warns the microcontroller that there are operations to perform.
6. The reader waits for the replies and then searches for the ID of the required operation inside the REPLY register and reads the content of the DATA register to retrieve the required information in case of READ operation.

For details, refer to the registers description below and to the *Communication Protocol NFC Side* paragraph page 37.

Warning: To increase the efficiency of read and write operations on the tag, it is possible to select several registers simultaneously using the SIZE register

NFC Registers Description

COMMAND register (RW)

Register	Page	Op.
COMMAND	0x04	RW

COMMAND is a 32 bit word register used to select between a READ or a WRITE operation of the tag internal registers or log area. This register is composed by the following fields:

Byte[0](MSB)	Byte[1]	Byte[2]Byte[3](LSB)
ID	CMD	RFU

ID: Message Identifier. This value, chosen by the user, will be echoed by the tag in its subsequent reply.

CMD: The command to be executed. Allowed values are the following:

Command	Value	Description
CMD_READ	0x12	performs a read operation from internal registers or from the log area.
CMD_WRITE	0x13	performs a write operation to the tag internal registers.

RFU: Not used.

ADDRESS register (RW)

Register	Page	Op.
ADDRESS	0x05	RW

ADDRESS is a 32-bit register. It is composed by the following fields:

Byte[0](MSB)Byte[1]	Byte[2]Byte[3](LSB)
ADDRESS	RFU

ADDRESS: the address of the internal register or log data the read/write operation is to be performed from/on. If the read/write operation comprises two or more consecutive internal registers or data, the read/write will start from the register address stored in ADDRESS.

RFU: Not used.

SIZE register (RW)

Register	Page	Op.
SIZE	0x06	RW

SIZE is a 32-bit register. It is composed by the following field:

Byte[0](MSB)Byte[1]	Byte[2]Byte[3](LSB)
SIZE	RFU

SIZE: the number of registers or 16-bit word data to be read/written from the tag.

RFU: Not used.

Warning: To increase the efficiency of read and write operations on the tag, it is possible to select several registers simultaneously using the SIZE register

REPLY register (RW)

Register	Page	Op.
REPLY	0x07	RW

REPLY is a 32-bit register. used by the tag to return the result of the operation it performed after receiving a read/write command. This register is composed by following fields:

Byte[0](MSB)	Byte[1]	Byte[2]Byte[3](LSB)
ID	REPLY	RFU

ID: Last message identifier. It should match the COMMAND register ID value.

REPLY: Command return value. Allowed values are:

- ACK (0xAC): Command was successfully executed.
- NACK (0xFC): Command failed/Invalid parameter.

RFU: Not used.

DATA Area (RW)

Register	Page	Op.
DATA	0x08 - 0x6B	RW

The DATA 32-bit registers contain either the data/internal registers read from the tag or the values to be written.

The use is the same as the DATA area on the UHF side, except for the mapping of 16-bit word internal registers to the 32-bit interface register.

Example:

If the user wants to write the following 3 16-bit words to the internal registers (starting from the address specified in the ADDRESS register):

0x1234 0x5678 0xABCD

he should write in the DATA_AREA_START page (page 0x08) the following 32-bit word:

0x12345678

and in DATA_AREA_START + 1 page the following word:

0xABCDFFFF

where the last two bytes are not parsed/used.

TRIGGER register (W)

Register	Page	Op.
TRIGGER	0x85	W

The 32-bit TRIGGER register is used to start the execution of a command. After loading the COMMAND, ADDRESS, SIZE, and optionally DATA area registers, in order to start the command execution the user must perform a WRITE operation of the TRIGGER register.

Communication Protocol NFC Side

To read/write data into the tag's logical registers the user will have to perform the following steps:

1. Initialize the COMMAND, ADDRESS, SIZE register as required to perform the desired read/write operation. In case of a *CMD_WRITE* operation fill the DATA area with the values to be written into the tag's internal memory.
2. Start the command execution by a single write operation of the TRIGGER register in the USER DATA area.
3. Wait for the tag to execute command.
4. Read the REPLY register. If the ID field in the reply register matches the ID field of the COMMAND register then command execution is completed. If the ID is different return to step 3.
5. Check the value of the REPLY field of the REPLY register. If the REPLY field value is an ACK, the tag performed successfully the requested operation: in case of a CMD_READ command move to next step, otherwise command execution is completed, tag is ready to receive another command. If the REPLY field value is a NACK, either one or more parameters are invalid, or the operation failed.
6. If a CMD_READ command was performed, read the DATA area of the USER DATA memory to retrieve the values of the requested internal registers/log data.

Warning: The NFC shared interface memory cannot be accessed by the user (NFC side) and by the tag at the same time. In fact, the NFC reader and the tag attempting to access the interface memory at the same time could cause the failure of read/write operations to the interface, and to subsequent command executions. For this reason, there are timing constraints and general rules that must be adhered to when performing a communication with the tag:

1. Avoid performing writes to the TRIGGER register when not needing to trigger a command execution.
2. Avoid to attempt performing reads to the TRIGGER register.
3. To trigger a command execution, a single write of the TRIGGER register is enough. Do not perform more consecutive writes.
4. After triggering a command execution, before checking the REPLY register for a tag reply give the tag enough time to perform the desired operation. Usually a good approximation for both CMD_READ and CMD_WRITE commends is T \sim 100 msec.
5. If, after waiting enough time for an operation execution, a check of the REPLY register results in the tag not having yet completed the required operation and replied, avoid re-triggering the command execution right away. Please wait again and check again the REPLY register afterwards.

Due to the wait timing constraints, when needing to perform READ or WRITE operations on internal registers/data whose addresses are consecutive, the user can speed up significantly the execution by accessing internal registers in blocks and not individually, through the use of the SIZE register.

Warning: The use of the *Communication Protocol* on the NFC side is the same as the one on the RAIN RFID side, with the difference that **to trigger a command execution, a WRITE operation is to be performed on the TRIGGER register instead of a READ.**

6 TECHNICAL SPECIFICATIONS

Technical Specification Table

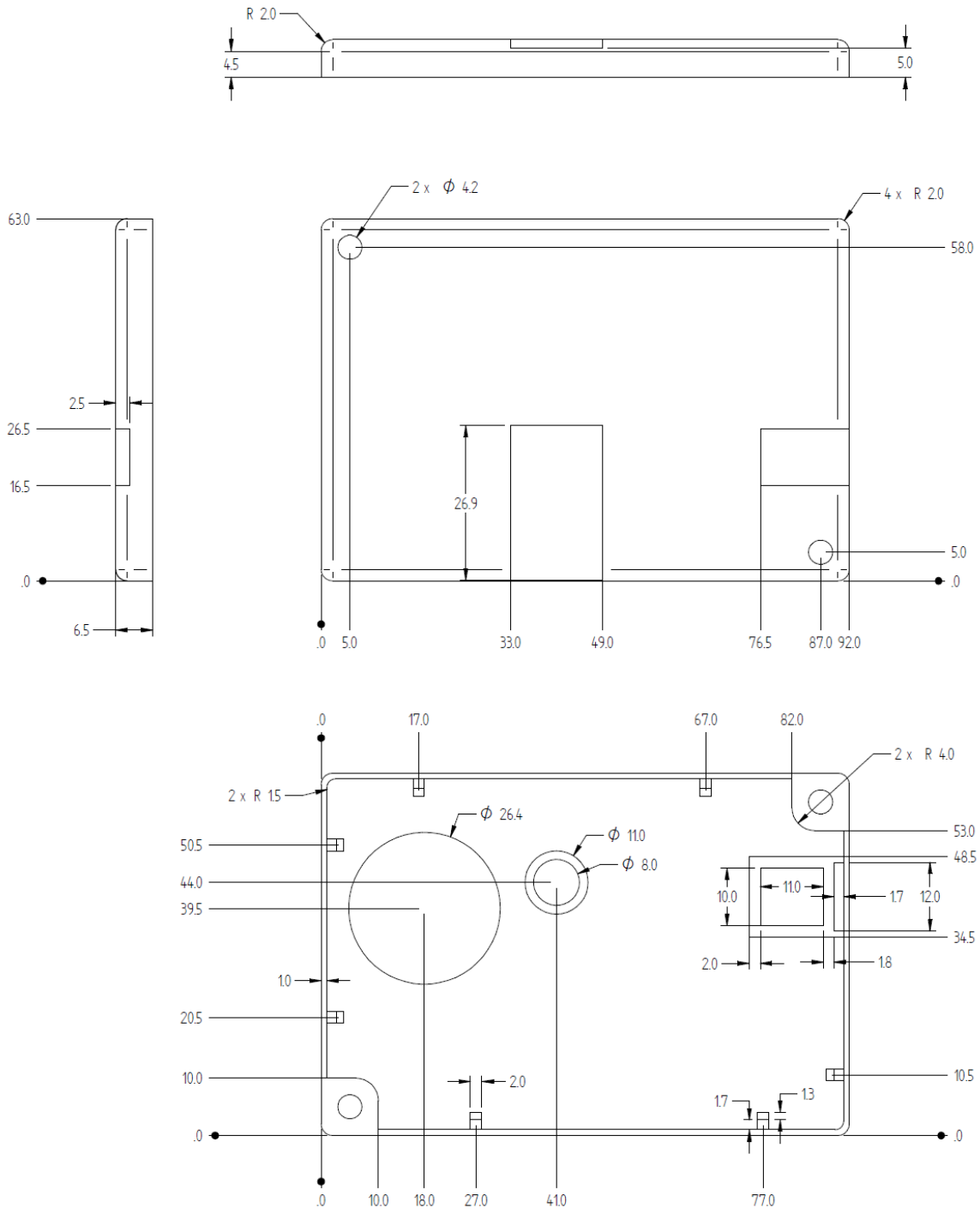
Frequency Range	– – NFC/HF: 13.56 MHz – – RAIN/UHF: 860÷930 MHz
RFID Protocols	– – NFC/RFID ISO 14443 Type A Interface – – RAIN : EPC Class 1 Gen 2 – ISO18000-63
Tag Type	Semipassive
Data Points	Up to 4096 samples
Operating Temperature	-30 °C to +70 °C
Temperature Accuracy	±0.5 °C
Humidity Range	0 to 100% relative humidity range
Humidity Accuracy	± 3.5% rH, 20 to +80% rH
Monitoring Time Span	Up to 5 years
Time Accuracy	<0.02% error (TBC)
Read Range	– NFC/HF: up to 5 cm. – RAIN/UHF: up to 5 mt. In free air @ 2W ERP
Available Memory	Up to 160 bits in EPC memory bank and up to 448 bits available for user
Monitoring Delay Option	Up to 18 hours
Features	– Multiple configurable sampling interval – Temperature and humidity histogram function – Configuration and start accessible both from NFC and RAIN interface – Samples download accessible both from NFC and RAIN interface – User accessible memory shared between NFC and RAIN
Alarms	– Multiple configurable high and low temperature/humidity thresholds – Estimated Time of Arrival – Battery Level
Battery Life	1 year typical (depending on usage and operating temperature)
Battery Type	Li/MnO2 Model Renata CR2430SN
IP rating	IP68
Enclosure Material	ABS
Dimensions	– (W)92 x (L)63 x (H)6.5 mm3 max. – 3.62 x 2.48 x 0.25 inches3
Weight	42 g.

Tab. 6.1: RT0013 qLog Technical Specifications Table

Warnings:

Do not incinerate, the product contains lithium battery.

Mechanical Specification



Dimensions are in mm