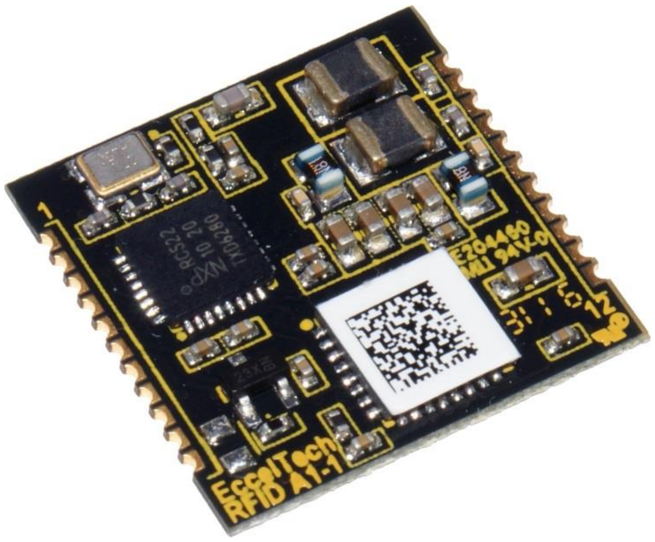
RFID A1 Module User Manual

V2.5
21/05/2018

Table of Contents

1 Introduction

1.1 Device Overview



Features

- Low cost RFID Reader with MIFARE Classic®, MIFARE Ultralight® and NTAG2xx® support
- Multiple Tag support
- Polling functionality (Stand-alone mode)
- Command interface via I2C and SPI
- I2C speed up to 400 kHz
- SPI speed up to 500 kHz
- Compact form factor
- Castellated SMT pads for easy and reliable PCB mounting
- High transponder read and write speed
- Low power design
- Single operating voltage: 2.5V to 3.6V
- -25°C to 85°C operating range
- AES-128 encryption engine
- RoHS compliant

Applications

- Access control
- Monitoring goods
- Approval and monitoring consumables
- Pre-payment systems
- Managing resources
- Connection-less data storage systems

Description

The RFID A1 module is the first in an evolving family of 13.56MHz sub assemblies from Eccel Technology Ltd (IB Technology). The product is designed with embedded applications in mind. This product is an ideal design choice if the user wishes to add RFID capability to their design quickly and without requiring extensive RFID and embedded software expertise and time. An on board low power ARM microcontroller handles the RFID configuration setup and provides the user with a powerful yet simple command interface to facilitate fast and easy read/write access to the memory and features of the various transponders supported by this module.

A polling option with configurable TPI outputs that are dependent upon the UID of the tag detected, makes it simple to build a stand-alone device that can control door locks, etc. and additionally send the signal to the host controller.

The module simply requires a single power and GND connection from the user PCB, along with two connections to an antenna. Eccel Technology Ltd (IB Technology) provide a range of suitable antennas designed for use with this module.

1.2 Pinout

Pin Number	Symbol	Type	Description
1	GND	Ground	
2	nRESET	Digital input with pull-up	Reset input signal (active low). This pin requires no external pull-up / down resitor unless the module is used in noisy environments, in which case connection of an external pull-up resistor combined with HF filter is recommended.
3	nBUSY	Digital push-pull output	Busy signal output indicating the device is working and the user should wait with any communication (active low) except status reading.
4	nPWRDN	Digital input with no pull resistors	Power Down Request input signal (active low). This pin has no pull-up/down resistor and should NOT be left floating. For power optimization it is recommended that the user drives this pin with a push-pull GPIO or similar.
5	I2C SDA	Digital input-outut with no pull resistors	Data input / output for the I2C Interface Bus. This pin has no pull-up / down resistor and should not be left floating. When using I2C communication external pull-up resistor is required matching the I2C bus requirements.
6	I2C SCL	Digital input with no pull resistors	Clock input for the I2C Interface Bus. This pin has no pull-up/down resistor and should not be left floating. External pull-up resistor is required matching the I2C bus requirements.
7	SPI MISO	Digital push-pull output	Master In Slave Out output for the Serial Peripheral Interface Bus.
8	SPI MOSI	Digital input with no pull resistors	Master Out Slave In input for for the Serial Peripheral Interface Bus.
9	SPI CS	Digital input with no pull resistors	Chip Select input for the Serial Peripheral Interface Bus. This pin has no pull-up / down resistor and should not be left floating.
10	SPI CLK	Digital input with no pull resistors	Clock input for the Serial Peripheral Interface Bus. This pin has no pull-up / down resistor and should not be left floating.
11	GND	Ground	
12	GND	Ground	
13	I2C AD2	Digital input with no pull resistors	Digital input for configuring the I2C Address bit 2. Value is sampled after the reset or power up. This pin has no pull-up / down resistor and should not be left floating.
14	I2C AD1	Digital input with no pull resistors	Digital input for configuring the I2C Address bit 1. Value is sampled after the reset or power up. This pin has no pull-up / down resistor and should not be left floating.
15	I2C AD0	Digital input with no pull resistors	Digital input for configuring the I2C Address bit 0. Value is sampled after the reset or power up. This pin has no pull-up / down resistor and should not be left floating.
16	TPI0	Digital push-pull output	Tag Presence Indicator 0 used in Polling mode.
17	TPI1	Digital push-pull output	Tag Presence Indicator 1 used in Polling mode.
18	V_{DD}	Power supply pin	Power Supply pin. Low ESR capacitor with capacitance 10uF or higher should be connected close to this pin.
19	GND	Ground	
20	ANT2	Analog output	Antenna output.
21	ANT1	Analog output	Antenna output.
22	GND	Ground	

Table 1-1

1.3 Application

The RFID A1 module is specifically designed for embedded applications, where a fast or low pin count connection to a host microcontroller is required. Fastest data exchange speed can be obtained by using the Serial Peripheral Interface Bus. The lowest pin count connection can be achieved by using the I2C Bus. For the module to be a fully functional RFID reader/writer it only requires connection to a power supply and antenna. Eccel Technology Ltd manufactures a wide range of such antennas designed to give optimal performance with this module. Please contact us for further details of our range of antennas.

1.4 Typical application schematic

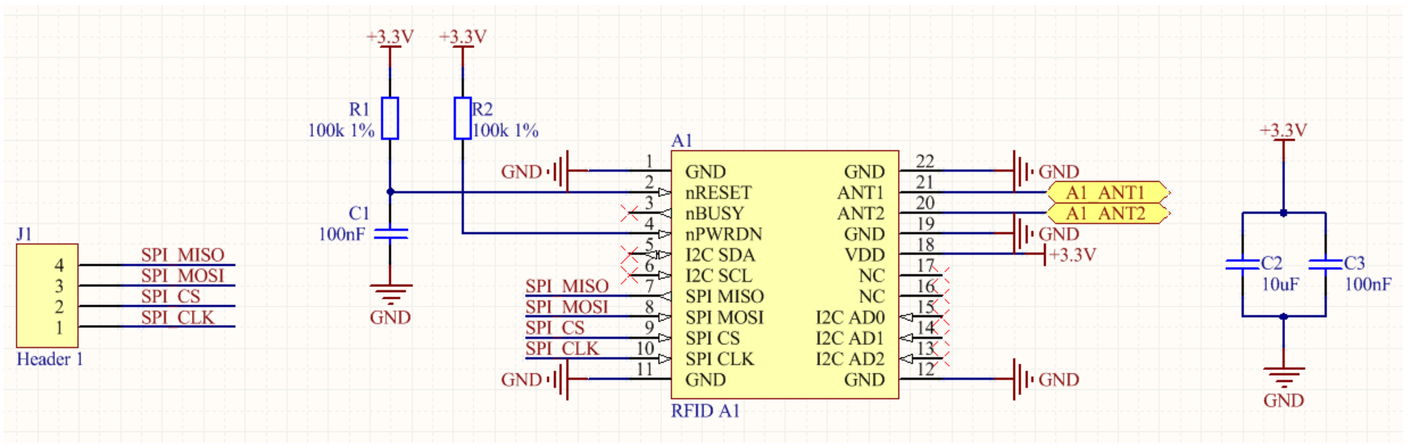


Figure 1.4-1 Typical application with SPI interface

2 Electrical Characteristics

2.1 Test Conditions

Typical device parameters have been measured at ambient temperature 22°C ±3°C and a power supply of 3.3V ±5%.

2.2 Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit	Notes
T_S	Storage Temperature	-40	150	°C	Tested for 10'000 hours at 150°C.
V_{DDMAX}	Supply Voltage	0	3.8	V	
V_{IOMAX}	Input Pin Voltage	-0.3	V_{DD} + 0.3	V	
I_{IOMAX}	Output Pin Current	0	6	mA	
I_{ANT}	ANT1 and ANT2 Current	0	100	mA	Maximum continuous current. This depends upon the impedance of the circuit between ANT1 and ANT2 at 13.56MHz.

Table 2-1

2.3 Operating Conditions

Symbol	Parameter	Min	Max	Unit
T_O	Ambient Temperature	-25	85	°C
V_{DD}	Supply Voltage	2.5	3.6	V

Table 2-2

2.4 Current Consumption

Symbol	Parameter	Typ	Max	Unit	Comment
I_{IDLE}	Idle Current	0,9	1,1	µA	T = 25°C, Vdd = 3.3V.
			1,8	µA	Full range of temperature and power supply voltage.
I_{PWRDN}	Power Down State Current	20	40	nA	T = 25°C, Vdd = 3.3V.
			400	nA	Full range of temperature and power supply voltage.
I_{TX}	Power Up Current	14	24	mA	T = 25°C, Vdd = 3.3V, 50Ω antenna connected between ANT1 and ANT2.
I_{TXMAX}	Maximum Current		120	mA	Maximum current consumed by the module in the worst conditions.

Table 2-3

Current measurement was taken place with all digital input pins connected to Vdd. Pins number 16 and 17 was left floating. Antenna was not connected except for the Power Up Current measurement and Maximum Current measurement.

2.5 GPIO

Symbol	Parameter	Min	Typ	Max	Unit	Notes
V_{IOIL}	Input Low Voltage			$0.3V_{DD}$	V	
V_{IOIH}	Input High Voltage	$0.7V_{DD}$			V	
I_{IOMAX}	Output Pin Current			± 6	mA	
I_{IOLEAK}	Input Leakage Current		± 0.1	± 40	nA	High impedance IO connected to V_{DD} or GND.
R_{IOESD}	Internal ESD Series Resistor		200		Ω	
V_{IOHYST}	IO Pin Histeresis	$0.1V_{DD}$			V	

Table 2-4

2.6 Antenna Output

Symbol	Parameter	Min	Typ	Max	Unit	Notes
f_{ANT}	Antenna Signal Frequency		13.56		MHz	±30 ppm (-20°C - 70°C).
f_{ANTAG}	Antenna Signal Frequency Aging	0		3	ppm	At 25°C.
V_{ANTH}	Antenna High Level Output Voltage	V_{DD} - 0.64			V	V_{DD} = 2.5V, I_{ANT} = 80mA.
V_{ANTL}	Antenna Low Level Output Voltage			0.64	V	V_{DD} = 2.5V, I_{ANT} = 80mA.
I_{ANT}	ANT1 and ANT2 Current	0	60	100	mA	Maximum continuous current. This depends upon the impedance of the circuit between ANT1 and ANT2 at 13.56MHz.

Table 2-5

2.7 Communication Buses

Symbol	Parameter	Min	Typ	Max	Unit	Notes
f_{I2CMAX}	I2C Maximum Clock Frequency			400	kHz	
f_{SPIMAX}	SPI Maximum Clock Frequency			500	kHz	
T_{CSCLK}	SPI CS To CLK Time Delay	2			µs	Minimum time delay between falling edge of Chip Select and first clock edge.
T_{CSHMIN}	SPI CS high Minimum Time	50			µs	Minimum time when CS is in HIGH state between SPI transmissions.

Table 2-6

2.8 Flash

Symbol	Parameter	Min	Typ	Max	Unit	Notes
C_{FE}	Flash Erase Cycles Before Failure	20000			cycles	
T_{FDR}	Flash Data Retention Time	10			years	For ambient temperature < 85°C
		20			years	For ambient temperature < 70°C

Table 2-7

3 System

3.1 Overview

The general overview of system components is shown in Figure 3.1-1. The system internally consists of four main parts:

- CORE – the main processing part of the microcontroller firmware responsible for managing all system tasks, parsing and the execution of commands received from the user's master controller.
- RFID – dedicated RFID IC together with its firmware drivers responsible for communication with the RFID tag.
- COMMUNICATION SYSTEM – microcontroller's peripherals (I2C and SPI) and its firmware drivers providing communication functionality to the user.
- POWER MANAGER – a subsystem responsible for managing power states and clocks in the module to minimise power consumption during operation.

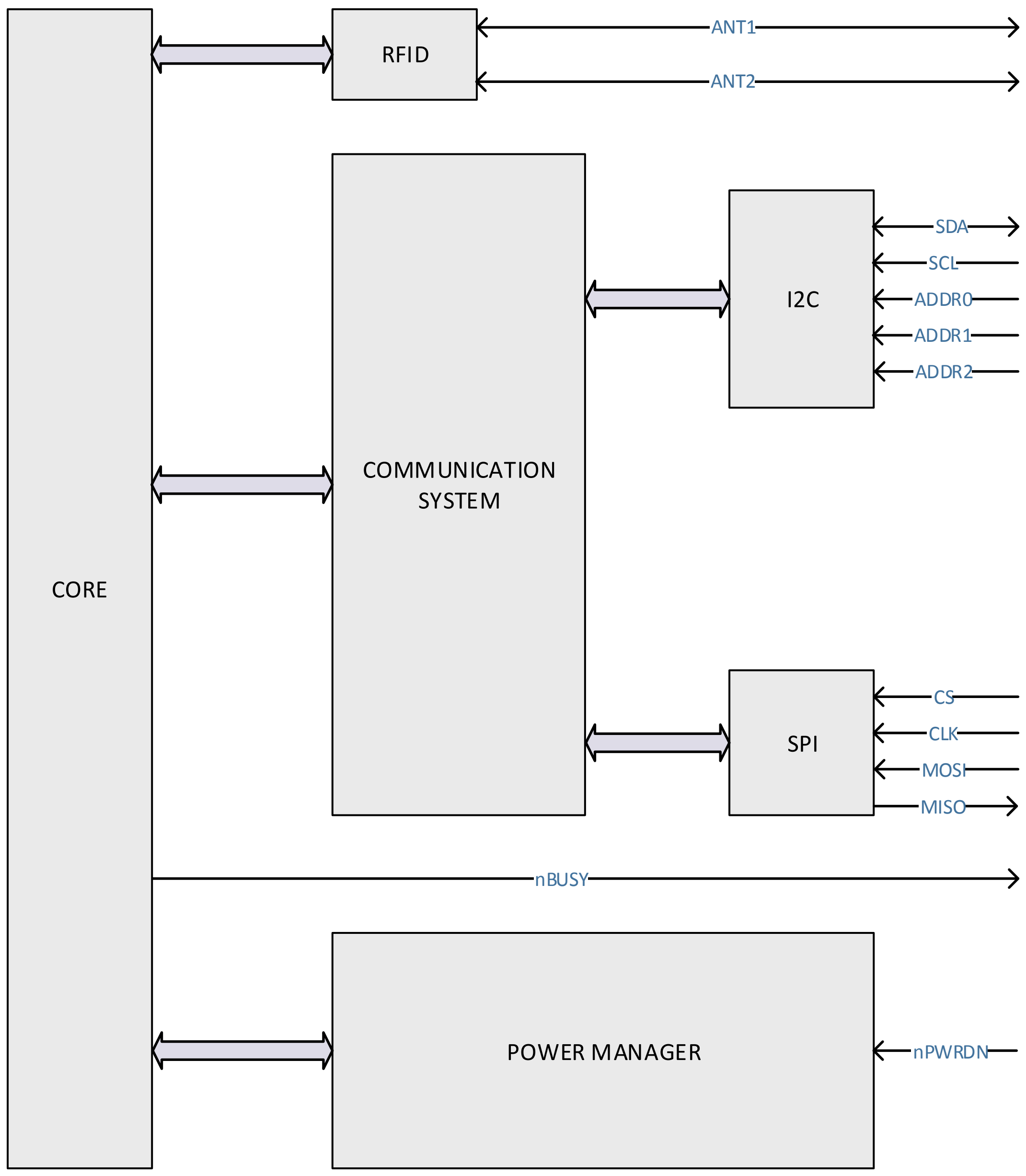

RFID
ANT1
ANT2
COMMUNICATION SYSTEM
I2C
SDA
SCL
ADDR0
ADDR1
ADDR2
SPI
CS
CLK
MOSI
MISO
CORE
nBUSY
POWER MANAGER
nPWRDN

Figure 3.1-1

3.2 Modules

3.2.1 Core

The core of the system shown above in figure 3.1, is the part of the modules' hardware and firmware responsible for managing tasks, prioritizing them and allocating resources to other components of the system. When the user sends a command, the core is the part of the system which ensures that the command is executed properly, that the outcome is saved and that a response is sent back to the user's master controller.

3.2.2 RFID

The RFID component is the part of the hardware and module firmware responsible for the complete communication interaction with the transponder via RF. In the RFID A1 module this part currently provides communication capabilities with MIFARE Classic®, MIFARE Ultralight® and NTAG2xx® transponder types. The key features of this RFID section are both fastest speed of transmission and system responsiveness, and optimal low power consumption.

3.2.3 Communication System

The Communication System component of the RFID A1 module is responsible for the entire communication with the user master controller, together with parsing commands for subsequent execution by the module core. Two communication interfaces are available: SPI and I2C. Both have equivalent access to the Communication System. It is possible to communicate with the module using all two interfaces at the same time, but it is strongly recommended to select and use only one at a time.

Communication peripherals are always active unless the nPWRDN signal is asserted. Doing so will put the module into Deep Sleep Mode and turn off entire Communication System.

3.2.4 Power Manager

The Power Manager is the part of the firmware, together with hardware support, that manages the power states of the system and tries to minimize power consumption as much as possible. The user has the option to put the module into Power Down Mode by pulling low the nPWRDN pin.

When the system enters Power Down Mode, all clocks and submodules are disabled and no response to user communication commands will occur until the system is re-enabled by the nPWDN being taken high by the user's system.

3.3 Memory Map

The device memory layout is shown below in Table 3.1.

In the RFID A1 module there are 728 bytes of user accessible memory. Each byte of the memory has a defined factory default value and these values can be recovered by using the 'Reset to Factory Defaults' command. The first 288 bytes are volatile memory which also have a default reset state that is the same as the factory default value. The other memory is buffered non-volatile memory and can be modified and stored using 'Unlock' and 'Lock' commands. The RFID A1 module is automatically returned to factory default state if after power up there is no valid configuration stored in non-volatile memory. The first 288 bytes of this user accessible memory are not stored in non-volatile memory and are always reset to factory defaults after a power-up sequence or after exit from the Power Down Mode.

Reading and writing to the registers can be done via one of two interfaces available on the module – I2C or SPI. The rest of the memory is accessible only when the module is unlocked. The default state of the module after power up is locked.

Address [DEC]	Address [HEX]	Size [bytes]	Description	Access	Locked and Stored in non-volatile memory
0	0x0000	1	Result	Read Only	No
1	0x0001	1	Command	R / W	No
2	0x0002	18	Command Parameters	R / W	No
20	0x0014	10	Tag UID	Read Only	No
30	0x001E	1	Tag Type	Read Only	No
31	0x001F	1	Tag UID Size	Read Only	No
32	0x0020	256	Data Buffer	R / W	No
288	0x0120	8	Password	R / W when unlocked	Yes
296	0x0128	16	AES Initialization Vector 0	R / W when unlocked	Yes
312	0x0138	16	AES Initialization Vector 1	R / W when unlocked	Yes
328	0x0148	16	AES Key 0	R / W when unlocked	Yes
344	0x0158	16	AES Key 1	R / W when unlocked	Yes
360	0x0168	6	Authentication Key / Password 0	R / W when unlocked	Yes
366	0x016E	6	Authentication Key / Password 1	R / W when unlocked	Yes
372	0x0174	6	Authentication Key / Password 2	R / W when unlocked	Yes
378	0x017A	6	Authentication Key / Password 3	R / W when unlocked	Yes
384	0x0180	6	Authentication Key / Password 4	R / W when unlocked	Yes
390	0x0186	6	Authentication Key / Password 5	R / W when unlocked	Yes
396	0x018C	6	Authentication Key / Password 6	R / W when unlocked	Yes
402	0x0192	6	Authentication Key / Password 7	R / W when unlocked	Yes
408	0x0198	6	Authentication Key / Password 8	R / W when unlocked	Yes
414	0x019E	6	Authentication Key / Password 9	R / W when unlocked	Yes
420	0x01A4	6	Authentication Key / Password 10	R / W when unlocked	Yes
426	0x01AA	6	Authentication Key / Password 11	R / W when unlocked	Yes
432	0x01B0	6	Authentication Key / Password 12	R / W when unlocked	Yes
438	0x01B6	6	Authentication Key / Password 13	R / W when unlocked	Yes
444	0x01BC	6	Authentication Key / Password 14	R / W when unlocked	Yes
450	0x01C2	6	Authentication Key / Password 15	R / W when unlocked	Yes
456	0x01C8	6	Authentication Key / Password 16	R / W when unlocked	Yes
462	0x01CE	6	Authentication Key / Password 17	R / W when unlocked	Yes
468	0x01D4	6	Authentication Key / Password 18	R / W when unlocked	Yes
474	0x01DA	6	Authentication Key / Password 19	R / W when unlocked	Yes
480	0x01E0	6	Authentication Key / Password 20	R / W when unlocked	Yes
486	0x01E6	6	Authentication Key / Password 21	R / W when unlocked	Yes
492	0x01EC	6	Authentication Key / Password 22	R / W when unlocked	Yes
498	0x01F2	6	Authentication Key / Password 23	R / W when unlocked	Yes
504	0x01F8	6	Authentication Key / Password 24	R / W when unlocked	Yes
510	0x01FE	6	Authentication Key / Password 25	R / W when unlocked	Yes
516	0x0204	6	Authentication Key / Password 26	R / W when unlocked	Yes
522	0x020A	6	Authentication Key / Password 27	R / W when unlocked	Yes
528	0x0210	6	Authentication Key / Password 28	R / W when unlocked	Yes
534	0x0216	6	Authentication Key / Password 29	R / W when unlocked	Yes
540	0x021C	6	Authentication Key / Password 30	R / W when unlocked	Yes
546	0x0222	6	Authentication Key / Password 31	R / W when unlocked	Yes
552	0x0228	6	Authentication Key / Password 32	R / W when unlocked	Yes
558	0x022E	6	Authentication Key / Password 33	R / W when unlocked	Yes
564	0x0234	6	Authentication Key / Password 34	R / W when unlocked	Yes
570	0x023A	6	Authentication Key / Password 35	R / W when unlocked	Yes
576	0x0240	6	Authentication Key / Password 36	R / W when unlocked	Yes
582	0x0246	6	Authentication Key / Password 37	R / W when unlocked	Yes
588	0x024C	6	Authentication Key / Password 38	R / W when unlocked	Yes
594	0x0252	6	Authentication Key / Password 39	R / W when unlocked	Yes
600	0x0258	128	User Memory	R / W when unlocked	Yes

Table 3-1

3.3.1 Result Register

The Result Register is 1-byte long, with located at address 0x0000, with both read and write access. Writing to this register has no effect. The register contains the result (error code) of the last executed command. The list of all possible results is shown in Table 3-2.

Result Register Values		
Value	**Type**	**Description**
0x00	No Error	Command was executed successfully and results were stored in the registers.
0x01	Invalid Command	Value written to command register is invalid.
0x02	Invalid Command Parameter	One of the parameters taken by the command is invalid.
0x03	Indexes Out Of Range	Indexes passed as command parameters exceed limit.
0x04	Error When Writing To Non Volatile Memory	There was an internal error during writing to the non-volatile memory.
0x05	System Error	Internal system error. Shall be considered as fatal.
0x06	Tag CRC Error	During communication with the tag a CRC was not correct.
0x07	Tag Collision	Reserved for future use.
0x08	Tag is not present	There is no tag within range.
0x09	Tag Authentication Error	Authentication failed due to incorrect Autentication Key or Password.
0x0A	Tag Value Block Corrupted	At least one value block is corrupted in the tag memory.
0x0B	Module Overheated	A overheat was detected.
0x0C	Tag Not Supported	There is a tag in the field which is not supported.
0x0D	Tag Communication Error	There was an error during communication with the tag.
0x0E	Invalid Password	The Password used in the Unlock command string was invalid.
0x0F	Already Locked	You are trying to lock a module that is already locked.
0xFF	Module Busy	Your command was ignored because the module is busy. Retry later.

Table 3-2

3.3.2 Command Register

The Command Register is 1-byte, located at address 0x0001, with both read and write access. Writing to this register is recognized by the module as a command execution request. Depending upon the command (value written to the register) the Command Parameters Register is parsed to extract arguments for the command. Whilst commands are executing, the Result Register value is set to 0xFF and the nBUSY line is driven low. When command execution is complete the memory is updated and the nBUSY line returns to the high state indicating that the module is ready to receive another command.

3.3.3 Command Parameters Register

Register Name	Command Parameters																
Register Address	0x0002																
Byte Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F	0x10
Byte Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Factory Default Value	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Read Function	Parameters taken when executing commands. Each command has various number and order of parameters.																
Write Function																	

Table 3-3

The Command Parameters Register is 18-bytes long, located at addresses from 0x0002 to 0x0013, with both read and write access. This is the place from where the system parses the arguments necessary to perform requested operation when a command is executed. Depending upon the command, this register is parsed and interpreted in different ways. The details of interpretation of the data stored in this register can be found in chapter 4.2. The module never changes the values inside this register except after power-up or exit from Deep Sleep Mode.

3.3.4 Tag UID Register

Register Name	Tag UID									
Register Address	0x0014									
Byte Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09
Byte Position	0	1	2	3	4	5	6	7	8	9
Access	R	R	R	R	R	R	R	R	R	R
Factory Default Value	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Read Function	UID[0]	UID[1]	UID[2]	UID[3]	UID[4]	UID[5]	UID[6]	UID[7]	UID[8]	UID[9]

Table 3-4

Tag UID Register is 10-bytes long, located at addresses from 0x0014 to 0x001D, with read only access. In most cases the tag UIDs are 4 bytes, 7 bytes or 10 bytes long, thus this register can cover the longest UID but not necessarily all bytes available in the register will be used. Bytes within the register are ordered from the least significant byte to the most significant byte. This register is updated whenever Get UID and Type commands are used. The Tag UID Size Register contains information detailing how many bytes of the ten available represent the tag UID.

3.3.5 Tag Type Register

The Tag Type Register is 1-byte long located at address 0x001E, with read only access. This register contains information about the type of tag which was last seen in the field. Possible tag types are shown in Table 3-5.

Returned value	Tag type
0x00	No Tag
0x01	Incomplete Type
0x02	Ultralight
0x03	Ultralight EV1 80B
0x04	Ultralight EV1 164B
0x05	Classic Mini
0x06	Classic 1K
0x07	Classic 4K
0x08	NTAG203F
0x09	NTAG210
0x0A	NTAG212
0x0B	NTAG213F
0x0C	NTAG216F
0x0D	NTAG213
0x0E	NTAG215
0x0F	NTAG216
0x10	Unknown

Table 3-5

3.3.6 Tag UID Size Register

The Tag UID Size Register is 1-byte long, located at address 0x001F, with read only access. It contains the information of what the UID size in bytes was of the last tag in the field.

3.3.7 Data Buffer

The Data Buffer is a 256-byte long, located at address 0x0020 to 0x011F, with both read and write access. This buffer is used for data transfers between the tag and the user of the module.

3.3.8 Password Register

Register Name	Password							
Register Address	0x0120							
Byte Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Factory Default Value	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Read Function / Write Function	PASS[0]	PASS[1]	PASS[2]	PASS[3]	PASS[4]	PASS[5]	PASS[6]	PASS[7]

Table 3-6

The Password Register is 8-bytes long, located at address 0x0120 to 0x0127, with both read and write access but only after first unlocking the device. This register is inaccessible when the module is locked. It contains an 8-byte long password which must be used with the Unlock Command to unlock protected memory. This password can be changed when the device is unlocked. The password change will only be updated and become valid after executing the Lock command.

3.3.9 AES Initialization Vector Register

Register Name	AES Initialization Vector															
Register Address	0x0128,0x0138															
Byte Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
Byte Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Factory Default Value	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Read Function	InVec[0]	InVec[1]	InVec[2]	InVec[3]	InVec[4]	InVec[5]	InVec[6]	InVec[7]	InVec[8]	InVec[9]	InVec[10]	InVec[11]	InVec[12]	InVec[13]	InVec[14]	InVec[15]
Write Function	InVec[0]	InVec[1]	InVec[2]	InVec[3]	InVec[4]	InVec[5]	InVec[6]	InVec[7]	InVec[8]	InVec[9]	InVec[10]	InVec[11]	InVec[12]	InVec[13]	InVec[14]	InVec[15]

Table 3-7

The AES Initialization Vector Registers are two 16-bytes long registers, located at address 0x0128 to 0x0147, with both read and write access but only after first unlocking the device. These registers are inaccessible when the module is locked. They can be used as an initialization vector for the first encrypted data block. The byte order in the memory is from the least significant byte. Their role during data encryption and decryption is described in detail in chapter 4.2.8 and 4.2.9.

3.3.10 AES Key Register

Register Name	AES Key															
Register Address	0x0148, 0x0158															
Byte Offset	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
Byte Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Factory Default Value	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
Read Function	Key[0]	Key[1]	Key[2]	Key[3]	Key[4]	Key[5]	Key[6]	Key[7]	Key[8]	Key[9]	Key[10]	Key[11]	Key[12]	Key[13]	Key[14]	Key[15]
Write Function	Key[0]	Key[1]	Key[2]	Key[3]	Key[4]	Key[5]	Key[6]	Key[7]	Key[8]	Key[9]	Key[10]	Key[11]	Key[12]	Key[13]	Key[14]	Key[15]

Table 3-8

The AES Key Registers are two 16-bytes long registers, located at address 0x0148 to 0x0167, with both read and write access but only after first unlocking the device. These registers are inaccessible when the module is locked. Both registers contain an AES encryption key which can be used for encryption of the Data Buffer. Their role during encryption and decryption of the data in the buffer is described in detail in chapter 4.2.8 and 4.2.9.

3.3.11 Authentication Key / Password Register

Register Name	Authentication Key / Password					
Register Address	0x0168, 0x016E … 0x0252					
Byte Offset	0x00	0x01	0x02	0x03	0x04	0x05
Byte Position	0	1	2	3	4	5
Access	R/W	R/W	R/W	R/W	R/W	R/W
Factory Default Value	0x00	0x00	0x00	0x00	0x00	0x00
Read Function	KEY[0] / PASS[0]	KEY[1] / PASS[1]	KEY[2] / PASS[2]	KEY[3] / PASS[3]	KEY[4]	KEY[5]
Write Function	KEY[0] / PASS[0]	KEY[1] / PASS[1]	KEY[2] / PASS[2]	KEY[3] / PASS[3]	KEY[4]	KEY[5]

Table 3-9

The Authentication Key and Password Registers are forty 6-bytes long registers, located at address 0x0168 to 0x0257, with both read and write access but only after first unlocking the device. These registers are inaccessible when the module is locked. When working with MIFARE Classic® tags these registers contain the password keys used for block authentication in the tag. When working with Ultralight and NTAG transponders, these registers contain 4-byte passwords. There are forty Authentication Key and Password registers numbered from 0 to 39. The number of the key register to be used is passed as an argument in some commands.

3.3.12 User Memory

There are 128 bytes of memory available for the user as a protected memory space from address 0x0258 to 0x02D7. This memory is inaccessible when the device is locked. This data is stored into non-volatile memory when the Lock command is executed.

4 Functional Description

4.1 Overview

The RFID A1 module is designed to provide a simple interface to communicate and manage RFID tags from the MIFARE Classic®, MIFARE Ultralight® and NTAG2xx® families. The interface is provided via SPI and I2C busses. From the user perspective, the module is a memory region where he can read from and write to. Depending upon the memory address, the role of registers located in the memory differ as per the Memory Map description given in Chapter 3.3. By writing to and reading from these registers, the user can achieve the desired result.

4.2 Commands

The RFID A1 Module has a special register, the Command Register. Writing to this register is interpreted as a command execution request from the user. The command type corresponds to the value written into the register. Execution of a command starts when the communication is finished. Depending upon the bus used, definition of the communication endpoint varies. When using the I2C bus, communication is finished when a STOP condition is sent. In case of the SPI bus, communication is finished on the rising edge of the CS signal.

The nBUSY line goes from the high state to the low state and indicates that the module is busy (it parses data stored in the registers and executes commands, if any command was sent). A more detailed description of the signal behavior is given in chapter 5.1.

The full RFID A1 command list is shown below in Table 4-1.

Value	Command Type	Arguments Taken	Memory Modified
0x00	No action	-	-
0x01	Get UID and Type	-	Result Register, Tag Type, Tag UID, Tag UID Size
0x02	Read Block	Block Address, Read Length, Authentication Key Number, Authentication Key, Buffer Offset	Result Register, Data Buffer
0x03	Write Block	Block Address, WriteLength, Authentication Key Number, Authentication Key, Buffer Offset	Result Register
0x04	Read Data Block	Block Address, Read Length, Authentication Key Number, Authentication Key, Buffer Offset	Result Register, Data Buffer
0x05	Write Data Block	Block Address, Read Length, Authentication Key Number, Authentication Key, Buffer Offset	Result Register
0x06	Read Page	Page Address, Read Length, Buffer Offset	Result Register, Data Buffer
0x07	Write Page	Page Address, Read Length, Buffer Offset	Result Register
0x08	Encrypt Data	Encryption Key Number, Initialization Vector Number, Buffer Offset, Data Length	Result Register, Data Buffer
0x09	Decrypt Data	Encryption Key Number, Initialization Vector Number, Buffer Offset, Data Length	Result Register, Data Buffer
0x0A	Read Value	Block Address, Authentication Key Number, Authentication Key, Buffer Offset	Result Register, Data Buffer
0x0B	Write Value	Block Address, Authentication Key Number, Authentication Key, 32-bit signed Value, Stored Block Address	Result Register
0x0C	Increment Value	Block Address, Authentication Key Number, Authentication Key, 32-bit signed Delta Value	Result Register
0x0D	Decrement Value	Block Address, Authentication Key Number, Authentication Key, 32-bit signed Delta Value	Result Register
0x0E	Restore Value	Block Address, Authentication Key Number, Authentication Key	Result Register
0x0F	Transfer Value	Block Address, Authentication Key Number, Authentication Key	Result Register
0x10	Recover Value	Block Address, Buffer Offset, Authentication Key Number, Authentication Key	Result Register
0x11	Get Version	-	Result Register, Data Buffer
0x12	Read Signature	-	Result Register, Data Buffer
0x13	Configure UID	Authentication Key Number, Authentication Key, UID Type	Result Register
0x14	Read Counter	Counter Number, Buffer Offset	Result Register, Data Buffer
0x15	Increment Counter	Counter Number, 24-bit signed Increment Value	Result Register
0x16	Check Tearing Event	Counter Number, Buffer Offset	Result Register, Data Buffer
0x17	Password Authentication	Buffer Offset, Password Number, Password	Result Register, Data Buffer
0x18	Halt	-	Result Register
0x19	Calculate CRC	Memory Address, Data Legth, Buffer Offset	Data Buffer
0x1A	Copy Data	Destination Address, Source Address, Data Length	All
0x1B	Unlock	Password	Result Register
0x1C	Lock	-	Result Register, Non-volatile memory
0x1D	Get Module Version	-	Data Buffer
0x1E	Reset to Defaults	-	All
0x1F	Enumerate Tags UID	-	Result Register, Data Buffer
0x20	Enumerate Tags UID and Type	-	Result Register, Data Buffer
0x21	Select Tag	UID Size, UID	Result Register

Table 4-1

Depending upon the command, different arguments are taken as parameters and various registers are modified. After each command execution, the Result Register is updated.

4.2.1 Get UID and Type (0x01)

The ‘Get UID and Type’ command takes no arguments. After receiving this command, the RFID A1 Module checks for any tag presence in the field. If there is no tag in the field, it returns ‘No Tag’ value in the ‘Tag Type Register’. If there is a tag in the field, it reads its UID and type and writes it to Tag UID and Tag Type registers. The ‘UID Size Register’ together with the ‘Result Register’ are updated also. This command must always be executed first before any other command to turn on and initialize the tag. Additionally, this command must be executed after any error when doing any operations on a tag to reset the tag and to enable it to be able to respond to further command requests. Additionally, via this command one can discover serial numbers from MIFARE Plus®, MIFARE® DESfire® and any transponder compatible with the ISO14443A standard.

4.2.2 Read Block (0x02)

Command Number	0x02				
Command Name	Read Block				
Valid Tag Types	Mifare Classic				
Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04
Argument Name	Absolute Block Number	Read Length	Data Buffer Offset	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory that is to be read. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04.	Number of blocks to read. Value has to be larger than 0.	Data Buffer offset in bytes of where the data from the tag is to be stored. The total read size and the offset must not exceed the Data Buffer length (number of bytes).	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to decrypt the data read. If the most significant bit (bit 7) is set, then the key will be used as key B. If bit 7 is zero it will be used as key A. If bit 6 is set, then the key will be taken from the following argument list (next 6 bytes).	Optional parameter used when bit 6 in Authentication Key Number argument is set. This Key will be used as Key B if bit 7 in Authentication Key Number is set or as Key A if bit 7 is not set. The byte order is the least significant byte first.

Table 4-2

The Read Block command takes as arguments block number of the first block to read (Absolute Block Number), the number of blocks to read (Read Length), the byte offset in the Data Buffer (Data Buffer Offset), the Authentication Key Number and (optionally) the Authentication Key. If the Authentication Key is set-up, the command first tries to authenticate the tag memory sector which contains the first block. If authentication fails, the 'Result Register' is updated with the appropriate error value and command execution finishes. After a successful authentication, the first block is read and copied to the Data Buffer. The command continues to read and copy following blocks if the Read Length is higher than 1. If the next blocks fall into another tag memory sector, this sector is authenticated using the same key. If everything goes well all block bytes from all blocks are copied to the Data Buffer starting from Data Buffer Offset, and the Result Register is updated with the 'No Error' value. If the RFID Module is unable to read tag memory, an error value is stored in the Result Register.

4.2.3 Write Block (0x03)

Command Number	0x03				
Command Name	Write Block				
Valid Tag Types	Mifare Classic				
Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04
Argument Name	Absolute Block Number	Write Length	Data Buffer Offset	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory that is to be written. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04.	Number of blocks to write. Value has to be larger than 0.	Data Buffer offset in bytes from where the data is to be taken to write to the tag. The total write size and the offset must not exceed the Data Buffer length (number of bytes).	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to encrypt the data that is to be written. If the most significant bit (bit 7) is set, then the key will be used as key B. If bit 7 is zero it will be used as key A. If bit 6 is set, then the key will be taken from the following argument list (next 6 bytes).	Optional parameter used when bit 6 in Authentication Key Number argument is set. This Key will be used as Key B if bit 7 in Authentication Key Number is set or as Key A if bit 7 is not set. The byte order is the least significant byte first.

Table 4-3

The Write Block command takes as arguments block number of the first block to write (Absolute Block Number), the number of blocks to write (Write Length), the byte offset in the Data Buffer (Data Buffer Offset), the Authentication Key Number and (optionally) the Authentication Key. If the Authentication Key is set-up, the command first tries to authenticate the tag memory sector which contains the first block. If authentication fails, the Result Register is updated with the appropriate error code and command execution finishes. After a successful authentication, the data is copied from the Data Buffer to the first block. The command continues to copy data from the Data Buffer to the following blocks if the Write Length is higher than 1. If subsequent blocks fall into another tag memory sector, this sector is authenticated using the same key. If everything goes well all block bytes are copied from the Data Buffer starting from Data Buffer Offset, and the Result Register is updated with the 'No Error' code. If the RFID Module is unable to write to the blocks an appropriate error code is stored in the Result Register.

4.2.4 Read Data Block (0x04)

Command Number	0x04				
Command Name	Read Data Block				
Valid Tag Types	Mifare Classic				
Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04
Argument Name	Absolute Block Number	Read Length	Data Buffer Offset	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory that is to be read. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04.	Number of blocks to read. Value has to be larger than 0.	Data Buffer offset in bytes of where the data from the tag is to be stored. The total read size and the offset must not exceed the Data Buffer length (number of bytes).	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to decrypt the data read. If the most significant bit (bit 7) is set, then the key will be used as key B. If bit 7 is zero it will be used as key A. If bit 6 is set, then the key will be taken from the following argument list (next 6 bytes).	Optional parameter used when bit 6 in Authentication Key Number argument is set. This Key will be used as Key B if bit 7 in Authentication Key Number is set or as Key A if bit 7 is not set. The byte order is the least significant byte first.

Table 4-4

The Read Data Block command takes as arguments block number of the first block to read (Absolute Block Number), the number of blocks to read (Read Length), the byte offset in the Data Buffer (Data Buffer Offset), the Authentication Key Number and (optionally) the Authentication Key. If the Authentication Key is set-up, the command first tries to authenticate the tag memory sector which contains the first block. If authentication fails, the Result Register is updated with the appropriate error value and command execution finishes. After successful authentication, the first block is read and copied to the Data Buffer. The command continues to read and copy following blocks if the Read Length is higher than 1. The difference between the Read Data Block command and the Read Block command is that the first one omits blocks containing authentication keys and lock bits. It only reads the data blocks from tag memory. If the next blocks fall into another tag memory sector, this sector is authenticated using the same key. If everything goes well all block bytes from all blocks are copied to the Data Buffer starting from the Data Buffer Offset, and the Result Register is updated with the 'No Error' value. If the RFID Module is unable to read tag memory, an error value is stored in the Result Register.

4.2.5 Write Data Block (0x05)

Command Number	0x05				
Command Name	Write Data Block				
Valid Tag Types	Mifare Classic				
Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04
Argument Name	Absolute Block Number	Write Length	Data Buffer Offset	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory that is to be written. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04.	Number of blocks to write. Value has to be larger than 0.	Data Buffer offset in bytes from where the data is to be taken to write to the tag. The total write size and the offset must not exceed the Data Buffer length (number of bytes).	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to encrypt the data that is to be written. If the most significant bit (bit 7) is set, then the key will be used as key B. If bit 7 is zero it will be used as key A. If bit 6 is set, then the key will be taken from the following argument list (next 6 bytes).	Optional parameter used when bit 6 in Authentication Key Number argument is set. This Key will be used as Key B if bit 7 in Authentication Key Number is set or as Key A if bit 7 is not set. The byte order is the least significant byte first.

Table 4-5

The Write Data Block command takes as arguments block number of the first block to write (Block Address), the number of blocks to write (Write Length), the byte offset in the Data Buffer (Data Buffer Offset), the Authentication Key Number and (optionally) the Authentication Key. If the Authentication Key is set-up, the command first tries to authenticate the tag memory sector which contains the first block. If authentication fails, the 'Result Register' is updated with the appropriate error value and command execution finishes. After a successful authentication, the data is copied from the Data Buffer to the first block. The command continues to copy data from the Data Buffer to the following blocks if the Write Length is higher than 1. The difference between Write Data Block command and Write Block command is that the first one omits blocks containing authentication keys and lock bits. It only writes to the data blocks in tag memory. If subsequent blocks fall into another tag memory sector, this sector is authenticated using the same key. If everything goes well all block bytes are copied from the Data Buffer starting from the Data Buffer Offset, and the Result Register is updated with a 'No Error' value. If the RFID Module is unable to write to the blocks, an appropriate error code is stored in the Result Register..

4.2.6 Read Page (0x06)

Command Number	0x06		
Command Name	Read Page		
Valid Tag Types	Mifare Ultralight, Mifare Ultralight EV1, NTAG		
Argument Offset [bytes]	0x00	0x01	0x02
Argument Name	Absolute Page Number	Read Length	Data Buffer Offset
Argument Size [bytes]	0x01	0x01	0x01
Argument Description	Absolute order number of the first page in the memory that is to be read. It is counted from 0. It is not a byte address.	Number of pages to read. Value has to be larger than 1.	Data Buffer offset in bytes of where in memory the data from the tag is to be stored. Total read size and the offset cannot exceed the Data Buffer length.

Table 4-6

The Read Page command takes as arguments page number of the first page to read (Absolute Page Number), the number of pages to read (Read Length) and the byte offset in the Data Buffer (Data Buffer Offset). If any of the pages to be read are password protected a Password Authentication is necessary before doing a read operation. If read fails, the Result Register is updated with an error value and command execution finishes. After successful communication, the first page is read and copied to the Data Buffer. Command continues to read and copy subsequent pages if the Read Length is higher than 1. If everything goes well all bytes from all pages are copied to the Data Buffer starting from Data Buffer Offset, and the Result Register is updated with a 'No Error' value.

4.2.7 Write Page (0x07)

Command Number	0x07		
Command Name	Write Page		
Valid Tag Types	Mifare Ultralight, Mifare Ultralight EV1, NTAG		
Argument Offset [bytes]	0x00	0x01	0x02
Argument Name	Absolute Page Number	Write Length	Data Buffer Offset
Argument Size [bytes]	0x01	0x01	0x01
Argument Description	Absolute order number of the first page in the memory to which the data is to be written. It is counted from 0. It is not a byte address.	Number of pages to write. Value has to be larger than 1.	Data Buffer offset in bytes from where the data to be written to the tag is to be taken. Total write size and the offset cannot exceed the Data Buffer length.

Table 4-7

The Write Page command takes as arguments page number of the first page to be written (Absolute Page Number), the number of pages to be written (Write Length) and the byte offset in the Data Buffer (Data Buffer Offset). If any of the pages to be written are password protected, then a Password Authentication is necessary before doing a write operation. If write fails, the Result Register is updated with an error value and the command execution finishes. After successful communication, the data is copied from the Data Buffer to the first page. This Command continues to copy to subsequent pages if the Write Length is higher than 1. If everything goes well all bytes are copied from the Data Buffer starting from the Data Buffer Offset, and the Result Register is updated with a 'No Error' value.

4.2.8 Encrypt Data (0x08)

Command Number	0x08			
Command Name	Encrypt Data			
Valid Tag Types	-			
Argument Offset [bytes]	0x00	0x01	0x02	0x03
Argument Name	Encryption Key Number	Initialization Vector Number	Data Buffer Offset	Data Length
Argument Size [bytes]	0x01	0x01	0x01	0x01
Argument Description	Number of the AES encryption key to be used for encryption. Valid numbers are 0x00 and 0x01.	Number of the Initialization Vector to be used for encryption. Valid numbers are 0x00 and 0x01.	Data Buffer offset in 16-byte blocks from where encryption will start. Total data size and the offset cannot exceed the Data Buffer length.	Number of 16 byte blocks of data to be encrypted. This value can be between 1 and 16.

Table 4-8

The Encrypt Data command takes as command arguments the Encryption Key Number (0x00 or 0x01), the Initialization Vector Number (0x00 or 0x01), the Data Buffer Offset (16-bytes blocks) and the Data Length (16-bytes blocks). This command encrypts the ‘Data Length’ number of 16-byte blocks using the AES128 CBC encryption methodology. It operates only within the Data Buffer. This encryption method is shown in Figure 4.2-1.

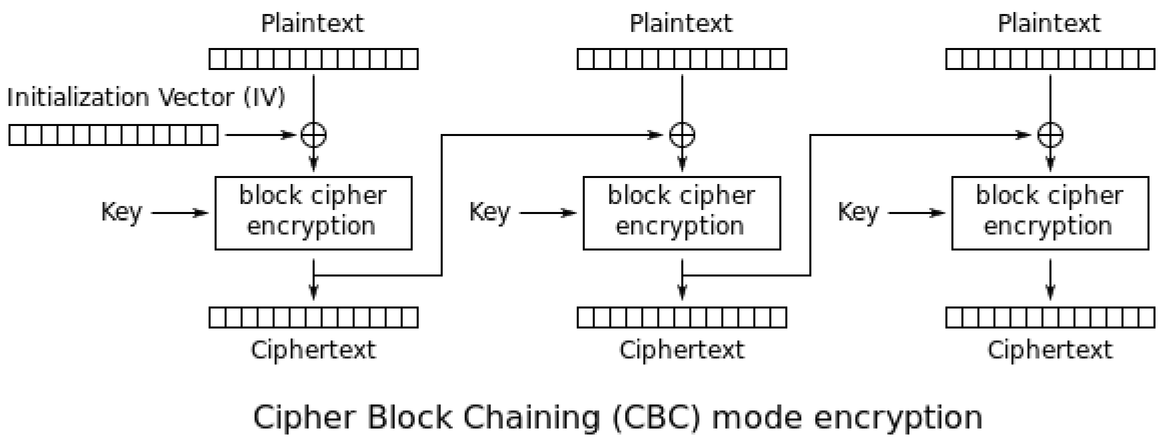


Figure 4.2-1

The Initialization Vector is used at the beginning to encrypt the first plaintext. For each subsequent 16-byte block instead of the Initialization Vector, the algorithm uses the ciphertext output of the previous 16-byte block encryption. The same AES Key is used for all blocks. When working with the Data Buffer during encryption the plaintext is substituted with the corresponding ciphertext.

4.2.9 Decrypt Data (0x09)

Command Number	0x09			
Command Name	Decrypt Data			
Valid Tag Types	-			
Argument Offset [bytes]	0x00	0x01	0x02	0x03
Argument Name	Decryption Key Number	Initialization Vector Number	Data Buffer Offset	Data Length
Argument Size [bytes]	0x01	0x01	0x01	0x01
Argument Description	The number of the AES Key to be used for decryption. Valid numbers are 0x00 and 0x01.	Number of the initialization Vector to be used for decryption. Valid numbers are 0x00 and 0x01.	Data Buffer offset in 16-byte blocks from where the decryption is to start. Total data size and the offset cannot exceed the Data Buffer length.	The number of 16-byte blocks to be decrypted. This value has to be between 1 and 16.

Table 4-9

The Decrypt Data command takes as arguments the Decryption Key Number (0x00 or 0x01), the Initialization Vector Number (0x00 or 0x01), the Data Buffer Offset (16-bytes) and the Data Length (16-bytes). This command decrypts the 'Data Length' number of 16-bytes blocks using the AES128 CBC decryption methodology. It operates only within the Data Buffer. This decryption method is shown in Figure 4.2-2.

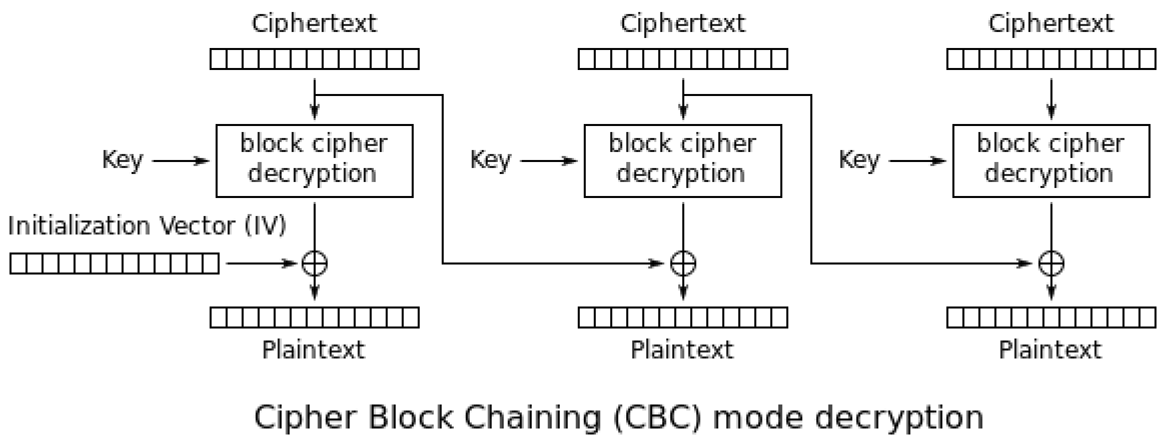


Figure 4.2-2

The Initialization Vector is used at the beginning to decrypt the first ciphertext. For each subsequent 16-byte block, instead of the Initialization Vector, the algorithm uses the ciphertext output of the previous 16-byte block decryption. The same AES Key is used for all blocks. When working with the Data Buffer during decryption the ciphertext is substituted with the corresponding plaintext.

4.2.10 Read Value (0x0A)

Command Number	0x0A			
Command Name	Read Value			
Valid Tag Types	Mifare Classic			
Argument Offset [bytes]	0x00	0x01	0x02	0x03
Argument Name	Absolute Block Number	Data Buffer Offset	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory that is to be read. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. The block has to be formatted as a value type block before reading.	Data Buffer offset in bytes from where the data read from the tag will be stored. Total read size and the offset cannot exceed the Data Buffer length.	The 6 least significant bits define the Authentication Key from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If bit 7 is zero, then it will be used as Key A. If bit 6 is set, then the key will be taken from the following argument list (next 6 bytes).	This optional parameter is only used when bit 6 of the immediately preceding Authentication Key Number argument is set. If used, this key will be used as Key A if bit 7 of the preceding Authentication number is set or as Key B if not set. The byte order is the least significant byte first.

Table 4-10

The Read Value command takes as arguments block number of the first block to read (Absolute Block Number), the offset of the Data Buffer where the value and read address will be stored (Data Buffer Offset), the Authentication Key Number and (optionally) the Authentication Key. This command has the same functionality as the Read Block command except it treats the block as a special Value type defined by the MIFARE® standard. It tries to parse the block content to this 4-byte signed value and to read a-byte address stored in last four bytes of the block. If read was successful, the value is stored in the Data Buffer at the pointed offset, with the address byte stored straight after the value. The value is stored as a signed 32-bit integer, with least significant byte first, and the address is stored as an unsigned 8-bit value.

4.2.11 Write Value (0x0B)

Command Number	0x0B				
Command Name	Write Value				
Valid Tag Types	Mifare Classic				
Argument Offset [bytes]	0x00	0x01	0x05	0x06	0x07
Argument Name	Absolute Block Number	Value	Stored Block Address	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x04	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory that is to be written. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. The block will be formatted as a value type block after writing.	Signed 32-bit value that is to be written to the value block. The least significant byte is stored first.	8-bit unsigned value to be stored in the last 4 bytes of the block memory.	The 6 least significant bits (bits 0-5) define the Authentication Key Number from 0 to 39 to be used to encrypt the data. If the most significant bit is set (bit 7), then the key will be used as Key B, if clear then the key will be used as Key A. If bit 6 is set, then the key will be taken from the following argument list (next 6 bytes).	Optional parameter used when bit 6 in Authentication Key Number argument is set. This Key will be used as Key B if bit 7 in Authentication Key Number is set or as Key A if bit 7 is not set. The byte order is the least significant byte first.

Table 4-11

The Write Value command takes as arguments block number of the first block to write (Absolute Block Number), the signed 32-bit integer value to be stored (Value), the block address value to store in the memory block (Stored Block Address), the Authentication Key Number and (optionally) the Authentication Key. This command has the same functionality as the Write Block command, except that it treats the block as a special Value type as defined by the MIFARE® standard. It tries to format the block content to this 4-byte signed value together with the address in the last 4 bytes of the memory.

4.2.12 Increment Value (0x0C)

Command Number	0x0C			
Command Name	Increment Value			
Valid Tag Types	Mifare Classic			
Argument Offset [bytes]	0x00	0x01	0x05	0x06
Argument Name	Absolute Block Number	Delta Value	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x04	0x01	0x06
Argument Description	Absolute order number of the first block in the memory from which the value is to be read and incremented. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. Block has to be formatted as a value type block before reading.	Signed 32-bit number to be added to the value read from the tag block.	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If it is zero it will be used as Key A. If bit 6 is set, then the key will be taken from the argument list (next 6 bytes).	Optional parameter used when bit 6 in the preceding Authentication Key Number argument is set. This key will be used as Key B if bit 7 of the Authentication Key Number is set, or as Key A if it is zero. The byte order is the least significant byte first.

Table 4-12

The Increment Value command is an operation on a value-type block as defined by the MIFARE® standard. It takes as arguments block number of the block where the value is stored (Absolute Block Number), the signed 32-bit integer number which will be added to the value (Delta Value), the Authentication Key Number and (optionally) the Authentication Key. The command reads the value from the block to the volatile memory on the tag and increments it by the selected Delta Value. There is no further operation done. To store the value in the same or another block (copy to non-volatile memory on the tag), the user must execute the Transfer Value command.

4.2.13 Decrement Value (0x0D)

Command Number	0x0D			
Command Name	Decrement Value			
Valid Tag Types	Mifare Classic			
Argument Offset [bytes]	0x00	0x01	0x05	0x06
Argument Name	Absolute Block Number	Delta Value	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x04	0x01	0x06
Argument Description	Absolute order number of the first block in the memory from which the value is to be read and decremented. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. Block has to be formatted as a value type block before reading.	Signed 32-bit number to be substracted from the value read from the tag block.	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If it is zero it will be used as Key A. If bit 6 is set, then the key will be taken from the argument list (next 6 bytes).	Optional parameter used when bit 6 in the preceding Authentication Key Number argument is set. This key will be used as Key B if bit 7 of the Authentication Key Number is set, or as Key A if it is zero. The byte order is the least significant byte first.

Table 4-13

The Decrement Value command is an operation on a value-type block, as defined by the MIFARE® standard. It takes as arguments block number of the block where the value is stored (Absolute Block Number), the signed 32-bit integer number which will be subtracted from the value (Delta Value), the Authentication Key Number and (optionally) the Authentication Key. The command reads the value from the block into the volatile memory on the tag and decrements it by the delta value. There is no further operation done. To store the value in the same or another block (copy to non-volatile memory on the tag), the user must execute the Transfer Value command.

4.2.14 Restore Value (0x0E)

Command Number	0x0E		
Command Name	Restore Value		
Valid Tag Types	Mifare Classic		
Argument Offset [bytes]	0x00	0x01	0x02
Argument Name	Absolute Block Number	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory from which the value is to be read and restored. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. The block has to be formatted as a value type block before reading.	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If it is zero it will be used as Key A. If bit 6 is set, then the key will be taken from the argument list (next 6 bytes).	Optional parameter used when bit 6 in the preceding Authentication Key Number argument is set. This key will be used as Key B if bit 7 of the Authentication Key Number is set, or as Key A if it is zero. The byte order is the least significant byte first.

Table 4-14

The Restore Value command is an operation on a value-type block, as defined by the MIFARE® standard. It takes as arguments block number of the block where the value is stored (Absolute Block Number), the Authentication Key Number and (optionally) the Authentication Key. If the block is properly formatted, then the value from the block is copied to a volatile memory register on the tag. There is no further operation done. To store the value in the same or another block (copy to non-volatile memory on the tag), the user must execute the Transfer Value command.

4.2.15 Transfer Value (0x0F)

Command Number	0x0F		
Command Name	Transfer Value		
Valid Tag Types	Mifare Classic		
Argument Offset [bytes]	0x00	0x01	0x02
Argument Name	Absolute Block Number	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory from which the value is to be transferred. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. The block is formatted as a value type block after execution of this command.	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If it is zero it will be used as Key A. If bit 6 is set, then the key will be taken from the argument list (next 6 bytes).	Optional parameter used when bit 6 in the preceding Authentication Key Number argument is set. This key will be used as Key B if bit 7 of the Authentication Key Number is set, or as Key A if it is zero. The byte order is the least significant byte first.

Table 4-15

The Transfer Value command is an operation on a value-type block as defined by the MIFARE® standard. It takes as arguments block number of the block where the value is stored (Absolute Block Number), the Authentication Key Number and (optionally) the Authentication Key. The value from the volatile register on the tag is copied to the pointed block. The block is formatted as a value-type block during this operation.

4.2.16 Recover Value (0x10)

Command Number	0x10			
Command Name	Recover Value			
Valid Tag Types	Mifare Classic			
Argument Offset [bytes]	0x00	0x01	0x02	0x03
Argument Name	Absolute Block Number	Data Buffer Offset	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x01	0x06
Argument Description	Absolute order number of the first block in the memory from which the value is to be read and recovered. It is counted from 0. It is not a byte address. For example, the block 0x00 in Mifare memory at sector 0x01 would have an Absolute Block Number of 0x04. The block must be formatted as a value type block before reading.	Data Buffer offset in bytes from where the data from the tag is to be stored. Total read size and the offset cannot exceed the Data Buffer length.	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If it is zero it will be used as Key A. If bit 6 is set, then the key will be taken from the argument list (next 6 bytes).	Optional parameter used when bit 6 in the preceding Authentication Key Number argument is set. This key will be used as Key B if bit 7 of the Authentication Key Number is set, or as Key A if it is zero. The byte order is the least significant byte first.

Table 4-16

The Recover Value command takes as arguments block number of the block where the value is stored (Absolute Block Number), the offset of the Data Buffer where the value and read address will be stored (Data Buffer Offset), the Authentication Key Number and (optionally) the Authentication Key. This command has the same functionality as the Read Value command, except that it can be used on a block which is corrupted – it tries to recover data from a corrupted block. The format of a value-type block allows for some bits to be corrupted and it still be possible to read and recover the proper value. The 4-byte signed value followed by 1-byte address value is stored in the Data Buffer under the Data Buffer Offset index.

4.2.17 Get Version (0x11)

The Get Version command doesn't take any arguments. This command can be used with MIFARE Ultralight® EV1, NTAG213®, NTAG215® and NTAG216® tags. After successful reading, the first 8 bytes in the data buffer are filled with data defined by the NXP standard. Example are shown in Table 4-17 (MIFARE Ultralight® EV1) and Table 4-18 (NTAG®).

Byte No	Description	MF0UL11/MF0ULH11	MF0UL21/MF0ULH21	Interpretation
0	Fixed header	0x00	0x00	
1	Vendor ID	0x04	0x04	NXP Semiconductors
2	Product type	0x03	0x03	MIFARE Ultralight
3	Product subtype	0x01/0x02	0x01/0x02	17 pF / 50 pF
4	Major product version	0x01	0x01	EV1
5	Minor product version	0x00	0x00	V0
6	Storage size	0x0B	0x0E	
7	Protocol type	0x03	0x03	ISO/IEC 14443-3 compliant

Table 4-17

Byte No	Description	NTAG213	NTAG215	NTAG216	Interpretation
0	Fixed header	0x00	0x00	0x00	
1	Vendor ID	0x04	0x04	0x04	NXP Semiconductors
2	Product type	0x04	0x04	0x04	NTAG
3	Product subtype	0x02	0x02	0x02	50 pF
4	Major product version	0x01	0x01	0x01	1
5	Minor product version	0x00	0x00	0x00	V0
6	Storage size	0x0F	0x11	0x13	
7	Protocol type	0x03	0x03	0x03	ISO/IEC 14443-3 compliant

Table 4-18

Please refer to the NXP documentation for more information.

4.2.18 Read Signature (0x12)

The Read Signature command doesn't take any arguments. This command can be used with MIFARE Ultralight® EV1, NTAG213®, NTAG215® and NTAG216® tags. After successful reading, the first 32 bytes in the data buffer are filled with the ECC signature defined by the NXP standard. Please refer to the NXP documentation for more information.

4.2.19 Configure UID (0x13)

Command Number	0x13		
Command Name	Configure UID		
Valid Tag Types	Some Mifare Classic		
Argument Offset [bytes]	0x00	0x01	0x02
Argument Name	UID Type	Authentication Key Number	Authentication Key
Argument Size [bytes]	0x01	0x01	0x06
Argument Description	UID Type configuration byte.	The 6 least significant bits define the Authentication Key Number from 0 to 39 that is to be used to read the data. If the most significant bit (bit 7) is set, then the key will be used as Key B. If it is zero it will be used as Key A. If bit 6 is set, then the key will be taken from the argument list (next 6 bytes).	Optional parameter used when bit 6 in the preceding Authentication Key Number argument is set. This key will be used as Key B if bit 7 of the Authentication Key Number is set, or as Key A if it is zero. The byte order is the least significant byte first.

Table 4-19

The Configure UID command takes as arguments the UID configuration byte (UID Type), the Authentication Key Number and (optionally) the Authentication Key. This command changes the configuration of the UID on some MIFARE Classic® tags. The UID Type parameter is explained in Table 4-20.

UID Configuration Parameter Value	Description
0x00	Anti-collision and selection with the double size UID.
0x01	Anti-collision and selection with the double size UID and optional usage of a selection process shortcut.
0x02	Anti-collision and selection with a single size random ID.
0x03	Anti-collision and selection with a single size NUID where the NUID is calculated from the 7-byte UID.

Table 4-20

4.2.20 Read Counter (0x14)

Command Number	0x14	
Command Name	Read Counter	
Valid Tag Types	Ultralight EV1, NTAG213, NTAG215, NTAG216, NTAG213F, NTAG216F	
Argument Offset [bytes]	0x00	0x01
Argument Name	Counter Number	Data Buffer Offset
Argument Size [bytes]	0x01	0x01
Argument Description	Counter number cannot exceed number of counters in the tag.	Data Buffer offset in bytes from where the counter value is to be stored. Total counter value size and the offset cannot exceed the Data Buffer length.

Table 4-21

The Read Counter command takes as arguments the tag Counter Number and the Data Buffer Offset in bytes. This command reads the counter value of the counter pointed to by the Counter Number and stores it in the Data Buffer at the Data Buffer Offset index as an unsigned 24-bit integer with the least significant byte first.

4.2.21 Increment Counter (0x15)

Command Number	0x15	
Command Name	Increment Counter	
Valid Tag Types	Ultralight EV1	
Argument Offset [bytes]	0x00	0x01
Argument Name	Counter Number	Increment Value
Argument Size [bytes]	0x01	0x03
Argument Description	Counter number cannot exceed number of counters in the tag.	24-bit unsigned integer value which will be added to the counter value. Byte order is least significant byte first.

Table 4-22

The Increment Counter command takes as arguments the tag Counter Number and the Increment Value to be added to the counter value. This command increments the counter value of the counter pointed to by the Counter Number. The Increment Value is a 24-bit unsigned number with the least significant byte first.

4.2.22 Check Tearing Event (0x16)

Command Number	0x16	
Command Name	Check Tearing Event	
Valid Tag Types	Ultralight EV1	
Argument Offset [bytes]	0x00	0x01
Argument Name	Counter Number	Data Buffer Offset
Argument Size [bytes]	0x01	0x01
Argument Description	Counter number cannot exceed number of counters in the tag.	Data Buffer offset in bytes where the check result will be stored. Offset cannot exceed the Data Buffer length.

Table 4-23

The Check Tearing Event command takes as arguments the tag Counter Number and the Data Buffer Offset in bytes. This command checks whether there was a tearing event in the counter and stores the flag value in the Data Buffer at the Data Buffer Offset index. The value '0x00' is stored if there has been no tearing event, and '0x01' is stored if a tearing event occurred.

4.2.23 Password Authentication (0x17)

Command Number	0x17		
Command Name	Password Authentication		
Valid Tag Types	Ultralight EV1, NTAG210, NTAG212, NTAG213, NTAG215, NTAG216, NTAG213F, NTAG216F		
Argument Offset [bytes]	0x00	0x01	0x02
Argument Name	Data Buffer Offset	Password Number	Password
Argument Size [bytes]	0x01	0x01	0x04
Argument Description	Data Buffer offset in bytes where the PACK result will be stored. Offset cannot exceed the Data Buffer length.	The 6 least significant bytes define the Password Number for 0 to 39 that is to be used to authenticate. If the value is equal to 0x80 the password will be taken from next four bytes of the command parameters.	An Optional parameter to be used when the preceding password number is set to 0x80. The byte order is the least significant byte first.

Table 4-24

The Password Authentication command takes as arguments the Data Buffer Offset where the PACK result will be stored, the Password Number and (optionally) the Password. This command tries to authenticate the tag using the chosen (pointed to) password. The 2-byte PACK result is stored in the Data Buffer under Data Buffer Offset index.

4.2.24 Halt (0x18)

The Halt command takes no arguments. It halts the tag and turns off the RF field. It must be executed at the end of each operation on a tag to disable the antenna and reduce the power consumption.

4.2.25 Calculate CRC (0x19)

Command Number	0x19		
Command Name	Calculate CRC		
Valid Tag Types	-		
Argument Offset [bytes]	0x00	0x02	0x04
Argument Name	Memory Address	Data Length	Data Buffer Offset
Argument Size [bytes]	0x02	0x02	0x01
Argument Description	Byte address in the memory from which the CRC calculation is to start. This is an unsigned 16-bit value with least significant byte first.	Data length in bytes upon which the CRC is to be performed. This is an unsigned 16-bit value with least significant byte first.	Buffer offset in bytes where the CRC value is to be stored. The CRC value is 16-bit unsigned with least significant byte first.

Table 4-25

The Calculate CRC command takes as arguments the Memory Address, the Data Length in bytes and the Data Buffer Offset. The CRC calculation starts at the byte pointed at by the memory address and is done on the ‘Data Length’ number of bytes in the volatile memory. The result is stored in the Data Buffer at the Data Buffer Offset index. The result is a 16-bit unsigned value with least significant byte first. The CRC calculation code is shown below.

```
static const uint16_t CCITTCRCTable [256] = {
0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5,
0x60c6, 0x70e7, 0x8108, 0x9129, 0xa14a, 0xb16b,
0xc18c, 0xd1ad, 0xe1ce, 0xf1ef, 0x1231, 0x0210,
0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c,
0xf3ff, 0xe3de, 0x2462, 0x3443, 0x0420, 0x1401,
0x64e6, 0x74c7, 0x44a4, 0x5485, 0xa56a, 0xb54b,
0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6,
0x5695, 0x46b4, 0xb75b, 0xa77a, 0x9719, 0x8738,
0xf7df, 0xe7fe, 0xd79d, 0xc7bc, 0x48c4, 0x58e5,
0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969,
0xa90a, 0xb92b, 0x5af5, 0x4ad4, 0x7ab7, 0x6a96,
0x1a71, 0x0a50, 0x3a33, 0x2a12, 0xdbfd, 0xcbdc,
0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03,
0x0c60, 0x1c41, 0xedae, 0xfd8f, 0xcdec, 0xddcd,
0xad2a, 0xbd0b, 0x8d68, 0x9d49, 0x7e97, 0x6eb6,
0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a,
0x9f59, 0x8f78, 0x9188, 0x81a9, 0xb1ca, 0xa1eb,
0xd10c, 0xc12d, 0xf14e, 0xe16f, 0x1080, 0x00a1,
0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c,
0xe37f, 0xf35e, 0x02b1, 0x1290, 0x22f3, 0x32d2,
0x4235, 0x5214, 0x6277, 0x7256, 0xb5ea, 0xa5cb,
0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447,
0x5424, 0x4405, 0xa7db, 0xb7fa, 0x8799, 0x97b8,
0xe75f, 0xf77e, 0xc71d, 0xd73c, 0x26d3, 0x36f2,
0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9,
0xb98a, 0xa9ab, 0x5844, 0x4865, 0x7806, 0x6827,
0x18c0, 0x08e1, 0x3882, 0x28a3, 0xcb7d, 0xdb5c,
0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0,
0x2ab3, 0x3a92, 0xfd2e, 0xed0f, 0xdd6c, 0xcd4d,
0xbdaa, 0xad8b, 0x9de8, 0x8dc9, 0x7c26, 0x6c07,
0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba,
0x8fd9, 0x9ff8, 0x6e17, 0x7e36, 0x4e55, 0x5e74,
0x2e93, 0x3eb2, 0x0ed1, 0x1ef0 };

static uint16_t GetCCITTCRC(const uint8_t* Data, uint32_t Size) {
uint16_t CRC;
uint16_t Temp;
uint32_t Index;

if (Size == 0) {
return 0;
}

CRC = 0xFFFF;

for (Index = 0; Index < Size; Index++){
Temp = (uint16_t)( (CRC >> 8) ^ Data[Index] ) & 0x00FF;
CRC = CCITTCRCTable[Temp] ^ (CRC << 8);
}

return CRC;
}
```

4.2.26 Copy Data (0x1A)

Command Number	0x1A		
Command Name	Copy Data		
Valid Tag Types	-		
Argument Offset [bytes]	0x00	0x02	0x02
Argument Name	Destination Address	Source Address	Data Length
Argument Size [bytes]	0x02	0x02	0x02
Argument Description	The byte address in the memory to where the data is to be copied.	The byte address in the memory from where the data is to be copied.	Size of the copied data in bytes.

Table 4-26

The Copy Data command copies data around inside the RFID Module memory. This command takes as arguments the Destination Address, the Source Address and the Data Length in bytes.

4.2.27 Unlock (0x1B)

Command Number	0x1B
Command Name	Unlock
Valid Tag Types	-
Argument Offset [bytes]	0x00
Argument Name	Password
Argument Size [bytes]	0x08
Argument Description	8-byte long password with the least significant byte first.

Table 4-27

The Unlock command takes as an argument an 8-bytes long password. If the password matches the actual password in the module the memory which contains:

- AES Initialization Vectors
- AES Encryption Keys

- Authentication Keys and Passwords
- User Memory
- Password

will be unlocked and made accessible for read and write operations.

4.2.28 Lock (0x1C)

This Function takes no arguments. If the device is unlocked and the user executes the Lock command, protected memory which contains:

- AES Initialization Vectors
- AES Encryption Keys
- Authentication Keys and Passwords
- User Memory
- Password

will be saved to non-volatile memory and locked (direct access will be blocked).

4.2.29 Get Module Version (0x1D)

The Get Module Version command takes no arguments. After execution, the Data Buffer at index 0x00 is filled with a NULL-ended ASCII string which describes the hardware and firmware version of the module.

4.2.30 Reset to Default (0x1E)

The Reset to Default command doesn't take any arguments. It resets all memory including protected memory to factory default settings. All locked information is lost.

4.2.31 Enumerate Tags UID (0x1F)

The Enumerate Tags UID command doesn't take any arguments. This command scans for all tags within the range of the antenna and enumerates their UIDs in the Data Buffer. The first byte in the Data Buffer says how many tags were discovered. If there are no tags in the range, then the Result Register is updated with 0x08 (Tag is not present) value. After enumeration, if there is at least one tag in the field, the Result Register is updated with 0x00 (No Error) value. Information about all enumerated tags is stored in the Data Buffer, starting from the second byte of the buffer. There are as many records in the Data Buffer as there were tags found. Each record contains the Tag UID Size as the first byte and the Tag UID is stored in the rest of the bytes in the record. The record length is variable and equal to the UID size plus one byte.

4.2.32 Enumerate Tags UID and Type (0x20)

The Enumerate Tags UID and Type command doesn't take any arguments. This command scans for all tags within the range of the antenna and enumerates their UID and type in the Data Buffer. The first byte in the Data Buffer says how many tags were discovered. If there are no tags in the range, then the Result Register is updated with 0x08 (Tag is not present) value. After enumeration, if there is at least one tag in the field the Result Register is updated with 0x00 (No Error) value. Information about all enumerated tags is stored in the Data Buffer starting from the second byte of the buffer. There are as many records in the Data Buffer as there were tags found. Each record contains the Tag Type as the first byte, the Tag UID Size as the second byte and the Tag UID is stored in the rest of the bytes in the record. The record length is variable and equal to the UID size plus two bytes.

4.2.33 Select Tag (0x21)

Command Number	0x21	
Command Name	Select Tag	
Valid Tag Types	-	
Argument Offset [bytes]	0x00	0x01
Argument Name	UID Size	UID
Argument Size [bytes]	0x01	0x04, 0x07, 0x0A
Argument Description	Size in byte of the UID which follows this parameter.	UID of the tag to select. The size of the UID is determined by previous parameter and cannot be other than 4, 7 or 10 bytes.

Table 4.28

The Select Tag command takes as arguments the UID Size in bytes and the UID bytes themselves. If the command execution is successful, then the addressed tag is selected and the Result Register is updated with 0x00 (No Error) value.

4.2.34 Polling (0x22)

Command Number	0x22					
Command Name	Polling					
Valid Tag Types	All					
Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04	0x05
Argument Name	Period [100ms]	Number of Defined Tags	Defined Tag: TPI Config	Defined Tag: Tag Timeout [100ms]	Undefined Tag: TPI Config	Undefined Tag: Tag Timeout [100ms]
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x01	0x01
Argument Description	Pooling period = value x 100mS	Number of definded Tags in User Memory from Index 7. Syntax: <UIDSize><UID>...	0b0000xxxx - do not use TPI outputs 0b0001xxxS - Set TPI0 to S State if Tag is on list 0b0010xxSx - Set TPI1 to S State if Tag is on list	Delay of next Tag detection	0b0000xxxx - do not use TPI outputs 0b0001xxxS - Set TPI0 to S State if Tag isn't on list 0b0010xxSx - Set TPI1 to S State if Tag isn't on list	Delay of next Tag detection

Table 4.29

Polling Period – Delay between polling attempts for detection of tag presence.
Number Of Defined Tags – States how many UIDs are stored in user memory.
Defined Tag: TPI Config – Bits 4 to 5 defines which TPI is used. Bits 0 to 2 defines state on defined Tag presence. See Table 4.30 for details.
Defined Tag: Tag Timeout – T = Value x 100mS. Delay between Tag detection and polling restart.
Undefined Tag: TPI Config – Bits 4 to 5 defines which TPI is used. Bits 0 to 2 defines the state on defined Tag presence. See Table 4.30 for details.
Undefined Tag: Tag Timeout – T = Value x 100mS. Delay between Tag detection and polling restart.

Parameter name	TPI Config							
Parameter Size	0x01							
Bit Number	7	6	5	4	3	2	1	0
Name	Not used	Not used	TPI1 enable	TPI 0 enable	Not used	Not used	TPI1 Active state	TPI0 Active state
Argument Description	Not used	Not used	If set to 1 then TPI1 if set to output. State is defined by bit 1 (TPI1 Active state). IF set to 0 then TPI1 is not used.	If set to 1 then TPI0 if set to output. State is defined by bit 0 (TPI0 Active state). IF set to 0 then TPI0 is not used.	Not used	Not used	If TPI1 enable is set to 1 then output is set to value of this bit.	If TPI0 enable is set to 1 then output is set to value of this bit.

Table 4.30

The Polling command takes arguments listed in the above tables. If the command execution is successful, then the Result Register is updated with 0x00 (No Error) value. For more information about the Polling command, please see chapter 4.3 Polling Mode.

4.3 Polling Mode

This command enables Tag UID polling. In this mode the module executes the continuous repeated Enumerate Tags UID command, writes the Command Polling value to the Command Register and parameters to the Parameter Registers to start polling. Parameters are described in 4.2.34. Polling mode can be stopped by writing any data to the Module memory.

4.3.1 Polling internal procedure

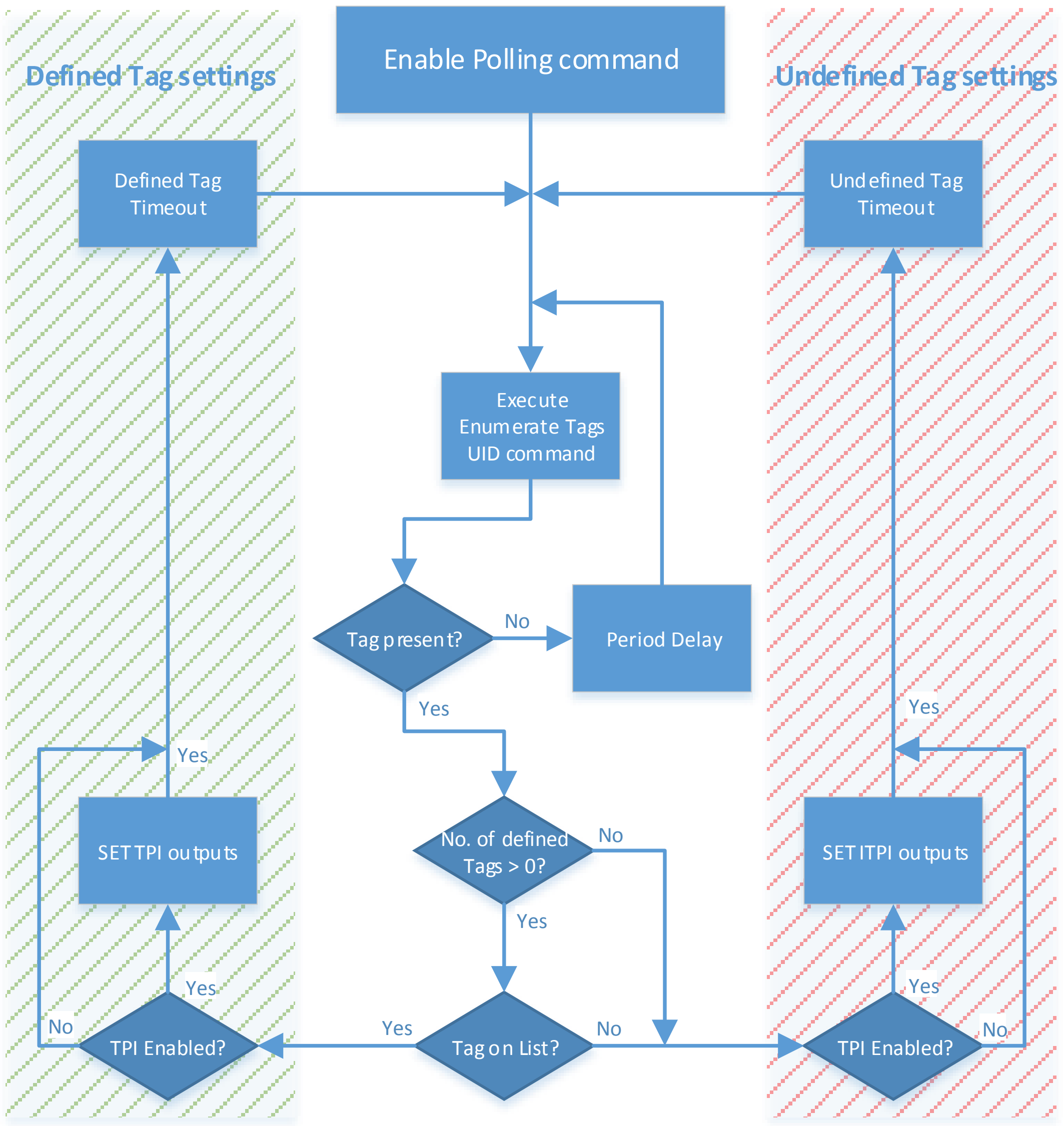


Figure 4.3

4.3.2 Polling auto start

Polling can be automatically started after power-up or reset. This functionality requires that a command and parameters be written to User Memory from index 0. Polling mode can be stopped by writing any data to the Module memory. Access to User Memory must be unlocked by the Unlock command before writing data. Data in the User Memory is saved after a Lock command.

The command followed by parameters is stored in User Memory from index 0x00.

Address hex	0x00	0x01	0x02	0x03	0x04	0x05	0x06
Address dec	0	1	2	3	4	5	6
Value type	**Command (0x22)**	Period [100ms]	Number of Defined Tags	Defined Tag: TPI Config	Defined Tag: Tag Timeout [100ms]	Undefined Tag: TPI Config	Undefined Tag: Tag Timeout [100ms]

Table 4.31

4.3.3 Polling use simple example – Undefined Tags

This example of polling command use demonstrates how to configure simple signalling of any Tag presence. Command Polling followed by parameters:

Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04	0x05
Argument Name	Period [100ms]	Number of Defined Tags	Defined Tag: TPI Config	Defined Tag: Tag Timeout [100ms]	Undefined Tag: TPI Config	Undefined Tag: Tag Timeout [100ms]
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x01	0x01
Argument Value	0x01	0x00	x	x	0x22	0x32
Argument Description	Polling period every 100mS	No Defined Tags in User Memory	Parameter ignored	Parameter ignored	Set TPI1 to High state	Wait for 5 seconds

Table 4.32

The module will check for Tag presence every 100ms. If a tag is detected, then TPI1 is set to the High state. After 5 seconds TPI1 is set to the Low state and the module checks for Tag presence once again.

4.3.4 Polling use simple example – Defined Tags

This example of a polling command use shows how to configure simple signalling of only defined Tag presence. Command Polling followed by parameters:

Argument Offset [bytes]	0x00	0x01	0x02	0x03	0x04	0x05
Argument Name	Period [100ms]	Number of Defined Tags	Defined Tag: TPI Config	Defined Tag: Tag Timeout [100ms]	Undefined Tag: TPI Config	Undefined Tag: Tag Timeout [100ms]
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x01	0x01
Argument Value	0x01	0x05	0x10	0x64	0x00	0x01
Argument Description	Polling period every 100mS	Five Defined Tags in User Memory	Set TPI0 to Low state	Wait for 10 seconds	Do not use TPI	Wait for 100mS

Table 4.33

The module will check for Tag presence every 100ms. If a tag is detected and this tag is on the defined list in user memory then TPI0 is set to the Low state. After 10 seconds TPI0 is set to the Low state and the module checks for Tag presence once again. The module does not react (only waits 100mS) if the detected Tag is not on list. Information of how to prepare a Defined Tag List can be found in chapter 4.3.6.

4.3.5 Polling use simple example – Undefined and Defined Tags

This example of polling command use shows how to configure simple signalling of defined and undefined Tag presence. Command Polling followed by parameters:

Argument Offset [bytes]	0x00	0x01	0x03	0x09	0x0B	0x11
Argument Name	Period [100ms]	Number of Defined Tags	Defined Tag: TPI Config	Defined Tag: Tag Timeout [100ms]	Undefined Tag: TPI Config	Undefined Tag: Tag Timeout [100ms]
Argument Size [bytes]	0x01	0x01	0x01	0x01	0x01	0x01
Argument Value	0x01	0x0A	0x11	0x64	0x20	0x1E
Argument Description	Polling period every 100mS	Ten Defined Tags in User Memory	Set TPI0 to High state.	Wait for 10 seconds	Set TPI1 to Low state.	Wait for 3 seconds

Table 4.34

The module will test for Tag presence every 100ms. If a tag is detected and this tag is on the defined list in the user memory, then TPI0 is set to the high state. After 10 seconds TPI0 is set to the low state and the module checks for tag presence once again. If the tag is not on the defined List, then TPI1 is set to the low state. After 3 seconds TPI1 is set to the high state and the module checks for Tag presence once again. Information of how to prepare a defined Tag List can be found in chapter 4.3.6.

4.3.6 Defined Tag List

The defined tag list is stored in User Memory from index 0x07. UIDSize followed by UID must be written for every listed tag.

Address hex	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0xD	0x0E	0x0F	0x10	0x11	0x12	0x13	…
Address dec	7	8	9	10	11	12	13	14	15	16	17	18	19	
Tag number	0					1								…
Value type	**UIDSize**	UID[0]	UID[1]	UID[2]	UID[3]	**UIDSize**	UID[0]	UID[1]	UID[2]	UID[3]	UID[4]	UID[5]	UID[6]	…
Value	**0x04**	0x00	0x01	0x02	0x03	**0x07**	0xAA	0xBB	0xCC	0xDD	0xEE	0xFF	0x32	

Table 4.35

If any of UIDSize contains an incorrect value (allowed values are 4, 7 and 10) then the module stops comparing the detected UID to the list. The number of Tags on the list is defined by the 'Number of Defined Tags' polling command parameter. Access to User Memory must be unlocked by the Unlock command before writing data. Data in User Memory is saved after a Lock command.

4.3.7 Polling stop and communication with tag example.

To communicate with one of the detected tags user must send Select Tag command followed by UID.

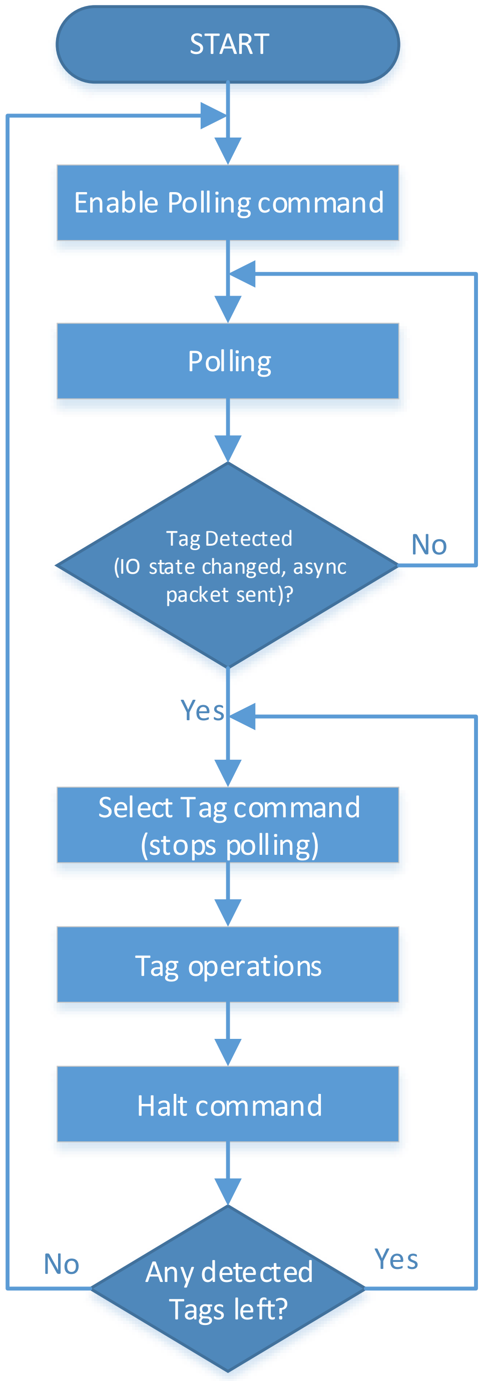


Figure 4.4 Polling stop and Tag r/w example.

4.4 Power Down Mode

The Power Down functionality is provided via the nPWRDN pin. This pin is configured as an input without any pull-up or pull-down resistors, thus it must be driven by the user. For normal operation, this pin should be connected to VCC or driven high. After the user drives this pin low the system prepares to go into Power Down Mode and drives nBUSY line low just before entering this state. To wake up module the nPWRDN pin must be driven high. The module will restart and drive the nBUSY line low to inform the user that it is starting the system again. When start-up is completed, the nBUSY line is driven high. During the restart, none of the volatile memory content is retained.

The reaction time of the system will vary depending upon the active configuration and the following:

- Communication – if the line is pulled low whilst there is ongoing communication, then the Power Manager will wait until communication is finished before going into the Power Down state.
- Command execution – if the line is pulled low whilst the system is executing a command, then the Power Manager will wait until command execution is finished before going into the Power Down state.

When the module is in the Power Down state, all systems are turned-off and all IOs are in a high impedance state. The module won’t respond to any signals or communication coming from outside.

4.5 Memory Locking

The module has an option to lock its memory using an 8-byte long password. The default state of the module after power-up is locked and the default password is all bytes equal to 0x00. The user cannot see the memory content when the module is locked, and any write to the memory will be discarded. During the unlock procedure, the content of the non-volatile memory is copied to the volatile memory and is available for reading and writing. Changes made in volatile memory are updated in non-volatile memory during the lock procedure. Commands using data from this memory range use the volatile memory when the module is unlocked and non-volatile memory when the module is locked

5 Communication Interface

5.1 nBUSY line

The RFID A1 Module provides two communication buses: SPI Slave and I2C Slave. Each of these buses has an equal level of access to the module internal registers. These buses operate independently but the user cannot use both at the same time. The module has no safety system and doing so can make the system work in unpredictable ways.

The RFID A1 Module provides nBUSY signal line which shares the functionality between both communication buses. This signal line is configured as an output. The nBUSY signal is driven low by the module when anything is written to the memory. At the same time when nBUSY line is driven low the result register is set to 0xFF value. This inform the user that module is busy and is parsing data or executing command and nothing else should be written to the module now. The user must wait for nBUSY line going high or Result Register value changing from 0xFF to perform any operation on the module or to read any data from the module which is a result of current processing.

5.2 SPI Bus

The RFID A1 Module provides a SPI slave bus. This is one of two buses provided by the module. It is the fastest communication option in the module. The high clock frequency together with simultaneous communication in both directions gives the fastest data exchange rate, but requires the most number of IO lines.

5.2.1 Bus signals and timings requirements

There are four signals necessary for communication via SPI bus. These are CLK, MOSI, MISO and CS. CLK (Clock), CS (Chip Select) and MOSI (Master Out Slave In) signals are controlled by the master on every transmission and these pins on the RFID A1 module are configured as inputs without any pull-up or pull-down resistors. MISO (Master In Slave Out) signal is driven by the module to Vcc and to ground when there is ongoing transmission. During communication via the SPI bus the user must pay attention to the nBUSY signal or must be sure that the Result Register has other value than 0xFF when writing to the module.

The SPI frame timing together with clock polarisation and phase is shown in Figure 5.2-1.

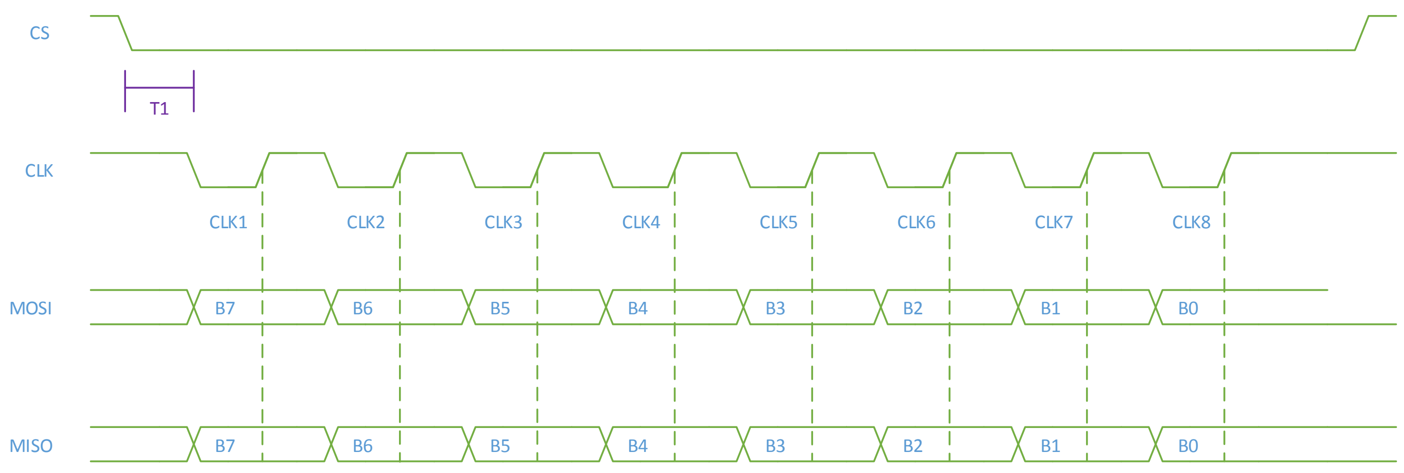


Figure 5.2-1

As Figure 5.2-1 shows the clock idle state is high. On a falling edge of the clock the module switches the state of the data line (MISO) and on a rising edge of the clock the module samples the data line (MOSI). During transmission, the most significant bit is transmitted first. One frame is eight bits long (one byte). The CS signal is active low thus the master must drive the CS signal low before clocking the clock signal. The maximum clock frequency for the system is 500 kHz. An additional restriction is also placed on the minimum allowed time between the CS line going low and the first clock falling edge and the minimum allowed time for CS being high between transmission (please see chapter 2.7). This is the result of low power consumption as the system must jump from sleep state to run state at various points during communication.

5.2.2 Exchanging data (reading or reading and writing)

The SPI bus is a type of synchronous bus where both sides (master and slave) synchronised by the master clock send data at the same time in both directions. Each transmission is initiated when the master drives the CS line low and ends when the CS line is driven high. All bytes transmitted in each direction between falling and rising edge of the CS line are considered as one packet and bytes within those packets are ordered. The First byte is transmitted always after the falling edge of the CS signal. A typical packet exchange is shown in Table 5-1.

Byte number		1	2	3	4		3 + n
MOSI	**Value**	0x00 - 0xFF	0x00 - 0xFF	0x00, 0x01	0x00 - 0xFF		0x00 - 0xFF
	Type	Address LSByte	Address MSByte	0x00 - Read, 0x01 - Read and Write	Data Byte 1	……..	Data Byte n
MISO	**Value**	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF		0x00 - 0xFF
	Type	Trash data	Trash data	Trash data	Data Byte 1	……..	Data Byte n

Table 5-1

When the CS signal goes low the byte index resets. The first three bytes transmitted from the module (MISO line) are trash and should be discarded by the master. The first three bytes transmitted from the master (MOSI line) are used to set up the transmission starting from byte four onwards. The first byte is the least significant byte of the address to where to write to or from where to read the data. The second byte is the most significant byte of the address. The address is represented as an unsigned 16-bit number. The third byte is the write indicator flag. If the master wants the data sent via the MOSI line to be written to the module's registers, then the third byte in the packet should have the value '0x01'. If the master sets this byte to value '0x00', then the RFID A1 module will discard incoming data (will not store them) and just stream the data on the MISO line from the RFID A1 memory.

5.2.3 Command processing

The RFID A1 module after CS goes high (end of communication) starts to process the data. Even if the master didn't write anything to the memory and just read data from the module, it will still drive the nBUSY line low for a while to do internal housekeeping. A typical communication timing is shown below in Figure 5.2-2.

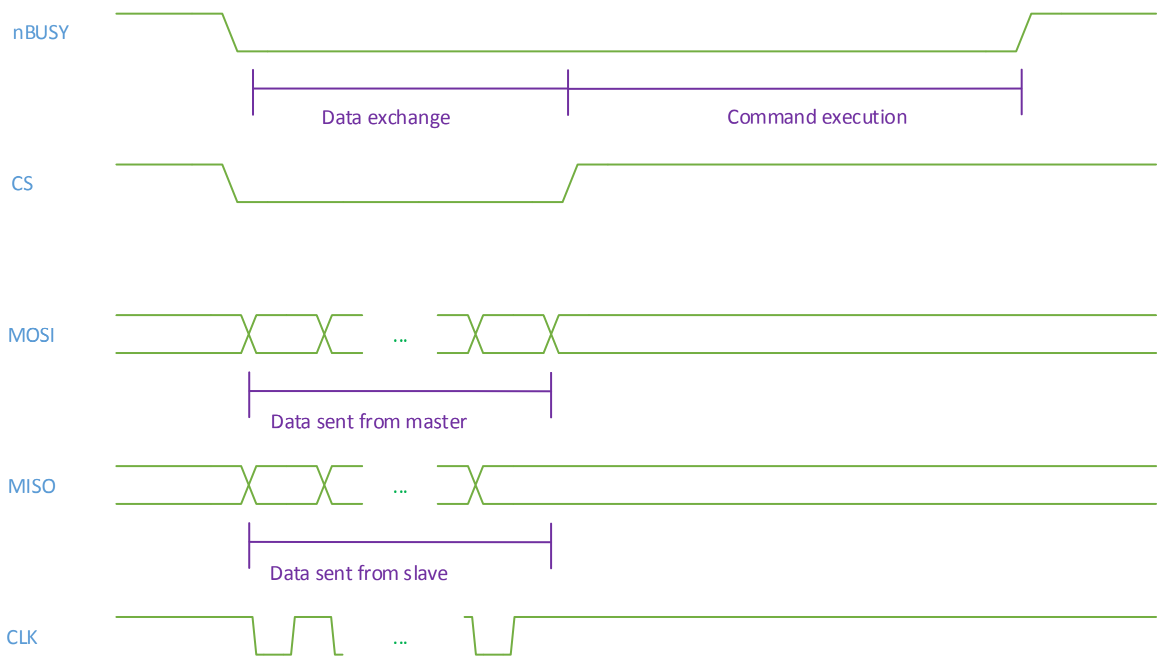


Figure 5.2-2

5.3 I2C Bus

The RFID A1 Module provides a I2C slave bus. The advantages of the I2C are that it requires only two lines to communicate and that multiple devices can be connected to the 2-wire same bus.

The implementation of I2C in the module complies with the 7-bit addressing I2C standard, thus the timing diagrams for bit and frame representation will be omitted in this document. Each transmission on the I2C line begins with an I2C START condition and ends with an I2C STOP condition. In between these events there could be as many repeated STARTs as the user wants. The behaviour of the nBUSY line is the same for all buses, thus as well for I2C. The module starts processing the command after receiving a stop condition on I2C bus.

5.3.1 I2C Address Construction

After reset or power-up when the module starts, it samples three IO lines (I2C AD0, I2C AD1 and I2C AD2) to construct the I2C address to which it will respond. Thanks to this solution there is the possibility to connect to eight such modules on one I2C bus. Byte wise the address sent on the I2C bus after START is constructed as follows:

Bit number	7	6	5	4	3	2	1	0
Value	0	1	0	0	Sampled from AD2	Sampled from AD1	Sampled from AD0	-
Type	Address bit 6	Address bit 5	Address bit 4	Address bit 3	Address bit 2	Address bit 1	Address bit 0	Read / Write bit

Table 5-2

5.3.2 Writing to the memory

Writing to the RFID A1 Module memory via the I2C bus is very similar to other memory type devices. A typical transmission for writing to the registers is shown below in Table 5-3.

Byte number	1	2	3	4		3 + n
Value	I2C Address + 0x00	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF		0x00 - 0xFF
Type	START + I2C address + Write	Address LSByte	Address MSByte	Data Byte 1	……..	Data Byte n + STOP

Table 5-3

The master must initiate the transmission on the I2C bus by sending a START condition followed by the I2C address of the slave device with R/W byte set to 0. The second and third byte sent by the master will set up the address index under which the data will be stored. Starting from the fourth byte the data will be stored under the passed address value and the memory index will be automatically increased by one each time. When the RFID A1 module recognizes its I2C address it drives the nBUSY line low to inform the master that it is receiving the data.

5.3.3 Reading from the memory

The master can read from the module in two possible ways presented below in Table 5-4 and Table 5-5.

Byte number	1	2	3	4	5		5 + n
Value	I2C Address + 0x00	0x00 - 0xFF	0x00 - 0xFF	I2C Address + 0x01	0x00 - 0xFF		0x00 - 0xFF
Type	START + I2C address + Write	Address LSByte	Address MSByte	START + I2C address + Read	Data Byte 1	……..	Data Byte n + STOP

Table 5-4

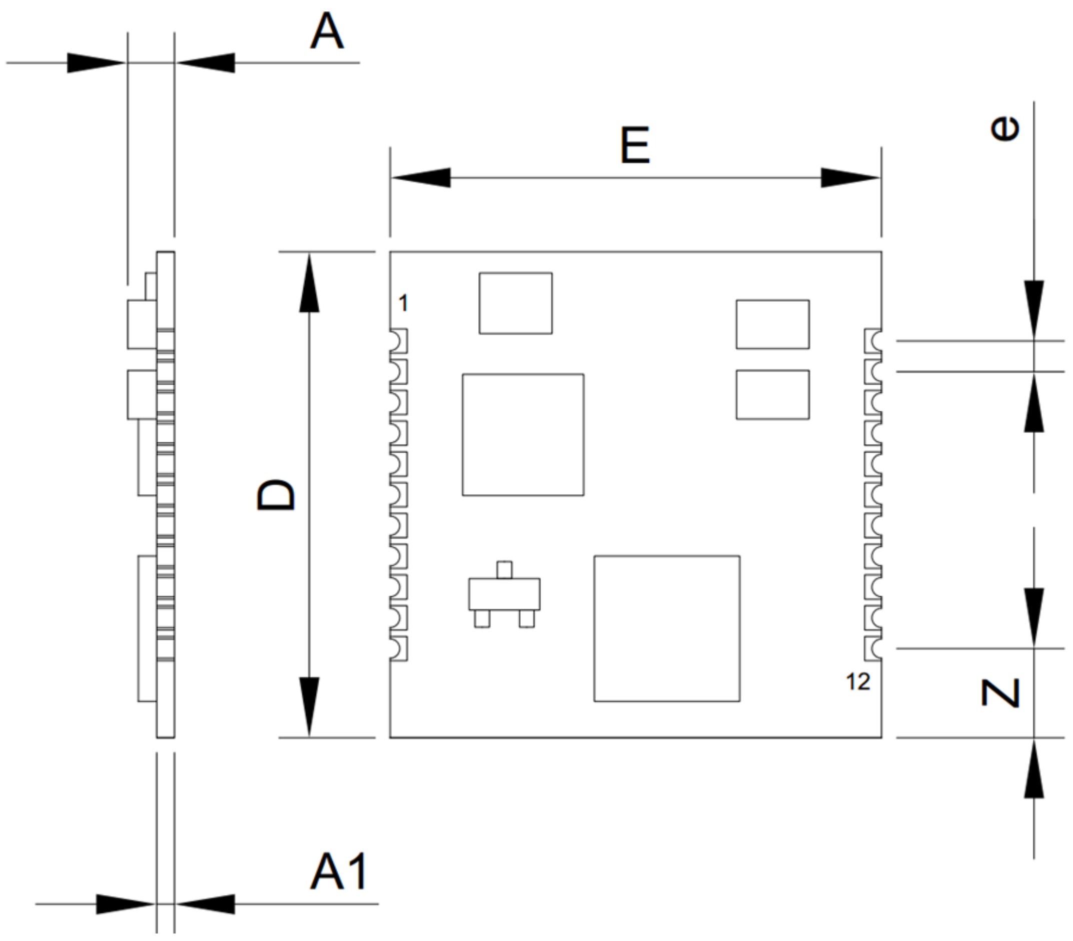
Byte number	1	2		2 + n
Value	I2C Address + 0x01	0x00 - 0xFF		0x00 - 0xFF
Type	START + I2C address + Read	Data Byte 1	……..	Data Byte n + STOP

Table 5-5

As shown in Table 5-4, to read from a certain address the master first must write the address index by sending a START condition followed by the I2C address to be written and two bytes representing the index. Later and without sending a STOP condition the master must send a repeated START signal, this time followed by the I2C address to be read. Bytes after the I2C read address will be transmitted from the module and will contain memory data starting from the address index set up by the master at the beginning of the transmission. The index is automatically increased and stored between transmissions. If the user wants to continue reading data starting from the index value left since last transmission, he can use the simplified version of the transmission shown in Table 5-5.

6 Mechanical

6.1 Dimensions

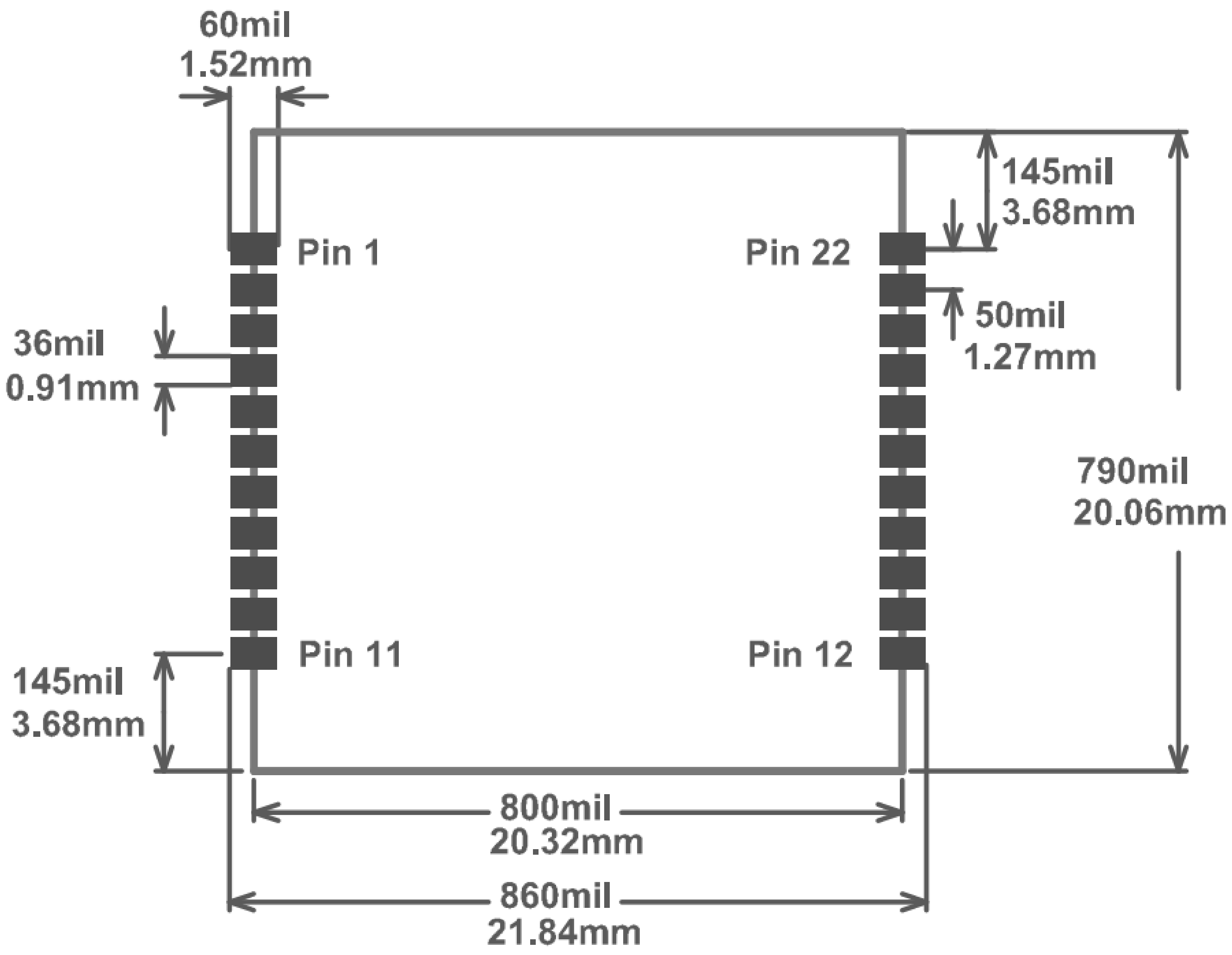


Drawing 1

UNIT	A	A1	D	E	e	Z
mm	2	0.7	20.06	20.32	1.27	3.68
inches	0.079	27.56	0.79	0.8	0.05	0.145

Table 6-1

6.2 Recommended Footprint



Drawing 2

No responsibility is taken for the method of integration or final use of the A1 module

More information about the A1 module and other products can be found at the Internet site:

http://www.eccel.co.uk

or alternatively contact ECCEL Technology (IB Technology) by e-mail at:

sales@eccel.co.uk