BioKey®
Protocol Definition

(03.02.2011, V6.9)

Revision history

Version	Date	Modification
2.16	29.06.2007	Refer to this version as reference
2.20	17.07.2008	Parameter “max. minuz” added in commands 0x08, 0x09, 0x0B Commands for time functions 0x19, 0x1A, 0x1C added Command 0x21 (Identification) extended Command 0x4A (Access Log) added
6.0	21.07.2008	New version system introduced firmware version contains protocol version: e.g. firmware version 6.0.0.0 uses protocol 6.0
6.1	19.08.2008	Command 0x29 added New method using Matching to select the best of three FPT
6.2	17.09.2008	Command 0x60 added: Get Software Composition Command 0x62 added: Get Version Number
6.3	10.12.2008	Cmd. 0x29 summary packet extended with index of best template Cmd. 0x43 one byte added for keeping module address after reset
6.4	02.02.2009	Command 0x1D, 0x1E added Time profile management for identified templates
6.5	29.06.2009	Cmds. 0x00,0x08,0x09 extended with 1 byte for SensorTimeout Cmds. For setting/getting the free time window added (0x1B, 0x1F)
6.6	31.03.2010	Cmds. 0x31 one more choice extended for info byte “buffer”. Templates can also be saved in database.
6.7	22.10.2010	Cmd 0x54 extended with 8 bytes (Info Bytes 1-4, Relay 1&2 Dur). Cmd 0xEA added for Encryptet Authentication. Cmd 0x43, Description updated. Cmd 0x58 can only be used if 0xEA is deactivated.
6.8	20.12.2010	Cmd 0x34 extended with 1 byte (transfer image in fast mode) for BioKey4080
6.9	03.02.2011	Cmd 0x0A extended for DATA1(Newly identified FPT will overlap the matched FPT in the database)

Please notice

IDENCOM does everything possible to keep the protocol working downwardly compatible, i.e. you can use the protocol the same way with a new version of the BioKey® software as you used it before.

In some cases protocol messages are extended e.g. if a new function is added to the software. Please take care that the length of some messages may change.

In the revision history all changes in every new version of the protocol are documented.

Contents

1. Data Transmission

1.1 Hardware

Data is transmitted serially and asynchronously using the TxD and RxD lines. The transmission is realized without handshake (CTS, RTS) and without XON/XOFF protocol. As data is transmitted at CMOS levels (3.3 V and 0 V), a conversion of the signal levels to RS232 levels might be needed.

data rate:	variable (9600, 19200, 38400, 57600 baud)
number of data bits:	8
order of data bits:	LSB (bit 0) first, MSB (bit 7) last
parity:	none
stop bits:	1

1.2 Packets

Data is transmitted in packets whose structure is explained in the following. Packet length may vary but is limited to a maximum of 1024 data bytes.

START	LENGTH	CMD	SENDER	DEST	DATA1	…	DATAx	ERRCHK

START (0x55)

1 byte start symbol, marks the beginning of a packet.

LENGTH

2 bytes packet length (bits 0…7 first), contains the number of data bytes DATA in packet. Not counted are START, LENGTH, CMD, SENDER, DESTINATION and ERRCHK, so a packet with 30 data bytes has a length value of 30. Maximum packet length is 1024.

CMD

1 byte, contains command for receiver.

SENDER

1 byte, contains sender's address.
Address range: 0x00 .. 0xFF (BioKey modules share range 0x01 .. 0xEF; Host PC in bus mode uses 0x00; 0xF0 .. 0xFF reserved)

DESTINATION

1 byte, contains receiver's address.
Address range: 0x00 .. 0xFF (BioKey modules share range 0x01 .. 0xEF; Host PC in bus mode uses 0x00; 0xF0 .. 0xFF reserved)

DATA 1…x

0 … x bytes, contain the data to be transmitted.
A packet can consist of 0 to 1024 data bytes.

ERRCHK

1 byte, contains a CRC check sum for error detection.

The check sum is computed with the functions in the CRC source code. All packet bytes in the given order (START, LENGTH, CMD, SENDER, DESTINATION, DATA1 … DATAx) are used for CRC calculation. The source code (crc.c and crc.h) is provided and documented.

1.3 Link

Module and terminal operate in half-duplex mode. Both point-to-point and bus mode are possible. This protocol only defines a master-slave link with the module being the slave and the terminal being the master.

1.4 Error Control / Flow Control

After reception of an error-free packet the receiver sends an ACK (acknowledge) which is not a packet but a single byte. If the analysis of the check sum shows an erroneous transmission, the receiver ignores the packet. After a timeout the sender has to retransmit the packet. If the module is busy it replies a NAK (negative acknowledge). In this case the sender can retransmit the packet immediately until an ACK is replied.
If the sender receives an undefined or no answer, the packet is retransmitted up to 2 times. In case the sender again does not receive a positive acknowledgement, the process is terminated as the connection or the receiver might by faulty.
If the sender received an ACK and the packet sent requires an answer packet, the sender has to wait up to 5 seconds for the answer.
The same conditions apply in bus mode.

ACK (0x06)

1 byte „Receive OK“, receiver’s answer after error-free reception

NAK (0x15)

1 byte „Busy“, receiver’s reply when busy

Exception:

Whenever secure communication mode is activated using message cmd 0x14, the message cmds, that are locked through this mode will be rejected by the module by responding with the SEC_LOCK Byte (value 0xcc) instead of ACK as long as no successful authentication using message cmd 0x80 has taken place. See message cmd 0x14 for more details.

2. Packet Definitions

2.1 General Information

The packets shown in the following have to be completed with START and ERRCHK. They only show an overview on the relevant data of the particular packet.
The addresses chosen are exemplary - the module's address was set to 0x01 which is the standard in point-to-point mode. Of course in bus mode the correct address has to be used, the modules' address range is 0x01 to 0xEF. It can be set using the configuration structure (see chapter 2.2). The sender's address was exemplarily set to 0x00 which is the standard for the host PC (terminal) in bus mode.
The terms GREY_IMAGE, BIN_IMAGE, FPT1, FPT2 and FPT_BUF used in chapters 2.3, 2.4 and 2.5 refer to module memory that can be read or written from outside the module using the serial interface. The abbrevations mean the following:

GREY_IMAGE:	used for greyscale image acquired from sensor or serial interface (r/w)
BIN_IMAGE:	used for binary image encoded from grayscale image (read-only)
FPT1:	used for template extracted from greyscale image (read-only)
FPT2:	used for template with highest matching score at last match (read-only)
FPT_BUF:	used for template buffering (read/write)

Some packets contain a PID which has a length of 8 bytes. Here the least significant byte (PID0) is transmitted first, the most significant byte last.

2.1.1 Module's answer packets

After the execution of most commands the module sends a standardized packet which contains the executed command as well as an execution status in data byte 1 (DATA1).

Standard answer packet:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	CMD	0x01	0x00	status

Depending on the command the status byte can carry the following values:

0x80: OK	the command has been executed successfully
0x81: ERROR	a non-specified error has occured during execution
0x82: BAD_IMAGE	no or bad sensor image / bad binary image
0x83: BAD_FPT	bad template, insufficient features
0x84: DB_FULL	database full
0x85: DB_EMPTY	database empty
0x86: IDS_USED	given IDs (PID, FID, AID) are used already
0x87: IDS_UNKNOWN	given IDs (PID, FID, AID) do not exist
0x88: RECOGNIZED	person was recognized
0x89: NOT_RECOGNIZED	person was not recognized
0x8A: NO_IMAGE	image from sensor could not be acquired
0x8B: NO_MATCH	templates did not match
0x8C: AUTH_FAILED	an authentication attempt to access secured message cmds in secure communication mode has failed (see command 0x80)

0x8D: FPT_EXISTS	an attempt to enroll with message cmd 0x26 failed because a template that matches the newly acquired one was found in the database
0x8E: FPT_ENROLLED	a newly acquired fingerprint has been enrolled successfully
0x8F: TIMEOUT	a timeout occurred during an enroll/identify process
0x90: FPT_READ	a fingerprint was read during “Enroll best of three” procedure

These status values also apply to other, non-standard answer packets. The structure of those non-standard answer packets will be specified in the chapters of the particular commands.

2.1.2 Module’s automatic information packet after identification (0x54)

This packet is an exception in the communication protocol as the module is not a slave executing the terminal’s commands but sends a packet automatically. After acquiring and matching a finger the module sends an information packet to an address set previously (see 2.2.14), but only if the Continuous Identification Mode (see 2.2.11) and the function Automatic Notification (see 2.2.14) have been activated. If multiple BioKey modules are connected to the terminal via RS485 bus, this message should not be used (disable Automatic Notification) to avoid possible bus collisions. In this case the identification results can be retrieved through continous polling of the modules using message 0x24 (see. 2.3.5).
The packet contains the following information on the matched finger: PID, FID, AID, matching score, classification (recognized, not recognized), the info bytes and the relay time.

Information packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000C	0x54	0x01	0xXX	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	score	status

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18	DATA19	DATA20
INFO 1	INFO 2	INFO 3	INFO 4	Relaytime 1	Relaytime 1	Relaytime 2	Relaytime 2

Packet contents:

DATA1 … DATA8:	PID: PID0 (least significant byte) … PID7 (most significant byte)
DATA9:	FID
DATA10:	AID
DATA11:	matching score
DATA12:	classification (RECOGNIZED / NOT_RECOGNIZED)
DATA13:	Info Byte 1
DATA14:	Info Byte 2 (Bit 1 & 2 contain Relay selection)
DATA15:	Info Byte 3
DATA16:	Info Byte 4
DATA17 – DATA18:	Time for Relay 1 in ms (Data17: least significant 8 bytes, Data18: most significant 8 bytes)
DATA19 – DATA20:	Time for Relay 2 in ms (Data19: least significant 8 bytes, Data20: most significant 8 bytes)

2.1.3 Module’s automatic information packet after enrollment (0x5A)

This packet is an exception in the communication protocol as the module is not a slave executing the terminal’s commands but sends a packet automatically. After enrolling a new finger in Continuous Enroll Mode (see 2.2.11) the module sends an information packet to an

address set previously (see 2.2.14), but only if Continuous Enroll Mode (see 2.2.11) and the function Automatic Notification (see 2.2.14) have been activated.
The enrolled fingerprint template is stored in the FTP_BUF template buffer (see 2.1).

Information packet from module:

LENGTH	CMD	SENDER	DEST
0x0000	0x5A	0x01	0xXX

2.2 Commands for System Configuration

For additional information on the following commands please refer to chapter 3.3.

2.2.1 Setting the sensor type (0x00)

Packet from terminal

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0004	0x00	0x00	0x01	sensor	temp. ctrl.	timeout.L	default

Extension since protocol version 6.5

LENGTH	…	DATA5
0x0005	…	timeout.H

This packet sets the sensor specified in DATA1. DATA2 contains the temperature set point for the Atmel FingerChip. DATA2 will only be evaluated if the sensor type is Atmel:

DATA2:	0	= temperature regulation disabled
	1	= temperature regulation enabled (default set point)
	18…50	= temperature regulation enabled (set point is 18…50 °C)

DATA1 = 0x00: Atmel Fingerchip™ Sensor
DATA1 = 0x01: Infineon FingerTIP™ Sensor
DATA1 = 0x02: Fingerprint Cards FPC1031 line sensor

DATA3, DATA5 specify in 1/10 seconds, how long the sensor will wait for a finger after an according command. DATA5 represents the MSB, DATA3 the LSB:

Timeout:

timeout.H	timeout.L
Bit15 Bit8	Bit7 Bit0

So timeouts up to 109 minutes are possible if the extended command is used.
Due to downward compatibility the former short command can be used too which enables maximal 25.5 seconds timeout.
DATA4 defines whether these values will be stored in flash memory as the default value (DATA4 = 1).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.2 Setting the image size (0x01)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0005	0x01	0x00	0x01	w[0…7]	w[8…15]	h[0…7]	h[8…15]

DATA5
default

This packet sets the image size specified in DATA1 to DATA4. DATA5 defines whether these values will be stored in flash memory as the default values (DATA5 = 1).

DATA1: image width low-byte (bits 0–7)
DATA2: image width high-byte (bits 8–15)
DATA3: image height low-byte (bits 0–7)
DATA4: image height high-byte (bits 8–15)

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.3 Setting the interface and Wiegand usage (0x02)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x02	0x00	0x01	device	default

This packet sets the interface device specified in DATA1. DATA2 defines whether this device will be stored in flash memory as the default device (DATA2 = 1). For Wiegand documentation, see the corresponding document. The following definitions are valid only if the BioKey® Module supports the Wiegand interface.
DATA1:0x00 = all available interfaces are used, Wiegand is enabled
0x01 = all available interfaces are used, but Wiegand is disabled
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.4 Setting the baud rate (0x03)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x03	0x00	0x01	Baudrate	default

This packet sets the baud rate coded in DATA1. DATA2 defines whether this baud rate will be stored in flash memory as the default value (DATA2 = 1).

DATA1:	0x00 = baud rate	9600
	0x01 = baud rate	19200
	0x02 = baud rate	38400
	0x03 = baud rate	57600
	0x04 = baud rate	115200
	0x05 = baud rate	1200
	0x06 = baud rate	2400
	0x07 = baud rate	4800

Baud rates 1200, 2400, 4800 and 115200 are available on BioKey3000 only.

After execution the module sends the standardized answer packet (see 2.1.1) with the previous baud rate. After transmission the baud rate will be set to the new value.
Status possible: OK, ERROR

2.2.5 Setting the LED use (0x04)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x04	0x00	0x01	led	default

This packet configures whether the module’s LED’s are to be used for signaling (DATA1 = 1). DATA2 defines whether this setting will be stored in flash memory as the default setting (DATA2 = 1).
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.6 Setting the debug functions use (0x05)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x05	0x00	0x01	debug	default

This packet configures whether debug functions are to be used. (DATA1 = 1). DATA2 defines whether this setting will be stored in flash memory as the default setting (DATA2 = 1).
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.7 Setting the module address (0x06)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x06	0x00	0x01	address	default

This packet sets the module’s address to the value specified in DATA1. If DATA2 = 1 this address will be stored in flash memory as the default address.
The modules’ address range is 0x01 to 0xEF, other addresses (0x00 or 0xF0 – 0xFF) will not be accepted by the module (status ERROR).
After execution the module sends the standardized answer packet (see 2.1.1) with its new address.
Status possible: OK, ERROR

2.2.8 Resetting module to delivery settings (0x07)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x07	0x00	0x01

This packet causes the module to reset its configuration to the delivery settings and store them in flash memory as default settings.
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.9 Receiving configuration from the module (0x08)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x08	0x00	0x01

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x001A	0x08	0x01	0x00	sensor	version1	version2	w[0…7]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
w[8…15]	h[0…7]	h[8…15]	debug	LED	sec. level	quality	min. minuz

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18	DATA19	DATA20
device	baud rate	address	ReadCont	format	access sig.	acc. dur. 0	acc. dur. 1

DATA21	DATA22	DATA23	DATA24	DATA25	DATA26
notification	notific. addr.	temp. ctrl.	timeout.L	max.minuz	timeout.H

This packet causes the module to send its entire configuration data to the terminal. In the module's answer packet DATA1 to DATA20 contain the configuration information.

DATA1:	0x00 = Atmel Fingerchip™ Sensor
	0x01 = Infineon FingerTIP™ Sensor
DATA2:	Software version 1 - Protocol version (the implemented protocol version)
DATA3:	Software version 2 - Revision number (used to identify small changes between firmware with the same core version and protocol version)
DATA4:	image width bits 0–7
DATA5:	image width bits 8–15
DATA6:	image height bits 0–7
DATA7:	image height bits 8–15
DATA8:	0x00 = not using debug functions
	0x01 = using debug functions
DATA9:	0x00 = not using LEDs
	0x01 = using LEDs
DATA10:	Security level
DATA11:	minimum image quality
DATA12:	minimum number of minutiae in template
DATA13:	0x00 = serial interface
DATA14:	0x00 = baud rate 9600
	0x01 = baud rate 19200
	0x02 = baud rate 38400
	0x03 = baud rate 57600
DATA15:	module address (0x01 … 0xEF)
DATA16:	0x00 = reading from sensor at the push of a button or command reception only
	0x01 = always reading from sensor (continuous read)
DATA17:	Minutiae format:
	0x00 = IDENCOM format
	0x01 = DIN V66400 format
	0x02 = IDENCOM Compact format
DATA18:	access signal use (see 2.2.13):
	0x00 = access signal is not set after successful identification
	0x01 = access signal will be set after successful identification
DATA19:	access signal duration bits 0-7 (see 2.2.13)
DATA20:	access signal duration bits 8-15 (see 2.2.13)

DATA21:	automatic notification (see 2.1.2 / 2.2.14): 0x00 = notification packet will not be sent after identification process 0x01 = notification packet will be sent after identification process
DATA22:	automatic notification address(see 2.1.2 / 2.2.14): contains the address the notification packet will be sent to
DATA23:	Atmel sensor temperature control: 0x00 = temperature regulation disabled otherwise: temperature set point in °C
DATA24:	sensor timeout byte 0 (LSB) in 1/10 seconds (see 2.2.1)
DATA25:	maximum number of minutiae in template
DATA26:	sensor timeout byte 1 (MSB) in 1/10 seconds (see 2.2.1)

Please notice that DATA2 and DATA3 only contain the Protocol and Revision Version of the Firmware. To get the complete Version number including the Core Version please use the command “2.6.14 Get Version Number (0x62)”.

Since protocol version 6.5 the packet is extended by DATA26.

2.2.10 Transmitting configuration to the module (0x09)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0018	0x09	0x00	0x01	sensor	w[0…7]	w[8…15]	h[0…7]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
h[8…15]	debug	LED	sec. level	quality	min. minuz	device	baud rate

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18	DATA19	DATA20
address	ReadCont	format	access sig.	acc. dur. 0	acc. dur. 1	notification	notific. addr.

DATA21	DATA22	DATA23	DATA24
temp. ctrl.	timeout.L	default	max. minuz

Extension since protocol version 6.5

LENGTH	…	DATA25
0x0019	…	timeout.H

This packet contains a new configuration for the module. DATA22 defines whether this configuration will be stored in flash memory as the new default configuration (DATA22 = 1).

DATA1:	0x00 = Atmel Fingerchip™ Sensor 0x01 = Infineon FingerTIP™ Sensor
DATA2:	image width bits 0–7
DATA3:	image width bits 8–15
DATA4:	image height bits 0–7
DATA5:	image height bits 8–15
DATA6:	0x00 = do not use debug functions 0x01 = use debug functions
DATA7:	0x00 = do not use LEDs 0x01 = use LEDs
DATA8:	Security level
DATA9:	minimum image quality
DATA10:	minimum number of minutiae in template

DATA11:	0x00 = serial interface
DATA12:	0x00 = baud rate 9600 0x01 = baud rate 19200 0x02 = baud rate 38400 0x03 = baud rate 57600
DATA13:	module address (0x01 … 0xEF, other ⇒ ERROR)
DATA14:	0x00 = read from sensor at the push of a button or command reception only 0x01 = always read from sensor (continuous read)
DATA15:	Minutiae format: 0x00 = IDENCOM format 0x01 = DIN V66400 format 0x02 = IDENCOM Compact format
DATA16:	access signal use (see 2.2.13): 0x00 = access signal is not set after successful identification 0x01 = access signal will be set after successful identification
DATA17:	access signal duration bits 0-7 (see 2.2.13)
DATA18:	access signal duration bits 8-15 (see 2.2.13)
DATA19:	automatic notification (see 2.1.2 / 2.2.14): 0x00 = notification packet will not be sent after identification process 0x01 = notification packet will be sent after identification process
DATA20:	automatic notification address(see 2.1.2 / 2.2.14): contains the address the notification packet will be sent to
DATA21:	Atmel sensor temperature control: 0x00 = temperature regulation disabled 0x01 = temperature regulation enabled (default set point) 18…50 = temperature regulation enabled (valid range 18…50 °C)
DATA22:	sensor timeout byte 0 (LSB) in 1/10 seconds (see 2.2.1)
DATA23:	0x00 = do not write configuration to flash memory 0x01 = write configuration to flash memory
DATA24:	maximum number of minutiae in template (valid range: min. minuz … 50)
DATA25:	sensor timeout byte 1 (MSB) in 1/10 seconds (see 2.2.1)

Please notice that this command is extended since protocol version 6.5 with DATA25 for sensor timeout setting. Due to downward compatibility the former short command can be used too.
See 2.2.1 for an explanation of the sensor timeout values.

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.11 Activating continuous read from sensor (0x0A)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x0A	0x00	0x01	active	default

This packet controls the sensor reading behaviour of the module. It will be ignored while the continuous mode switch is activated.

If DATA1 = 0, the module will read from the sensor at the push of a button or command reception only.

If DATA1 = 1, it will always read and identify from the sensor (continuous identification mode). In continuous identification mode, if a finger is on the sensor, the finger will be read, its features will be extracted and matched with the entire fingerprint database. If DATA2 = 1, the specified behaviour will be stored in flash memory as the default behaviour.

If DATA1 = 2, it will always read and enroll from the sensor (continuous enroll mode). In continuous enroll mode, if a finger is on the sensor, the finger will be enrolled to the FPT_BUF template buffer. After a successful enrollment the module will send an automatic information packet (see 2.1.3) if this function has been activated and exit the continuous enroll mode. DATA2 is ignored.

If DATA1 = 3, it will be like DATA1 = 1, but the difference is that **the newly sucessfully enrolled and identified FPT will overlap the best matched FPT in the database**. That is to say: **Database will be updated after one sucessful identification.**

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.12 Setting algorithm parameters (0x0B)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4	DATA5
0x0005	0x0B	0x00	0x01	sec. level	quality	min. minuz	max. minuz	default

This packet sets the algorithm parameters security level (DATA1), minimum image quality (DATA2) and minimum and maximum number of minutiae in a template (DATA3 and DATA4). If DATA5 = 1, these parameters will be stored in flash memory as default parameters.
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.13 Setting the access signal use (0x0C)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0004	0x0C	0x00	0x01	active	duration0	duration1	default

This packet sets the use of the access signal (Outport 3). If DATA1 = 1, the access signal will always be set after identification and classification as “recognized“ (matching score greater than security level). If DATA1 = 0, the access signal will never be set.
DATA2/DATA3 sets the duration the access signal will be active after successful recognition. Its unit is milliseconds (DATA2 bits 0-7, DATA3 bits 8-15).
DATA4 defines whether this behaviour will be stored in flash memory as default (DATA4 = 1).
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.14 Activating Automatic Notification (0x0D)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x0D	0x00	0x01	active	receiver	default

This packet controls whether the module sends an automatic notification packet in button mode or after identification or enrollment in continuous mode (see 2.2.11). If DATA1 = 1, this packet will be sent to the address specified in DATA2, otherwise no packet will be sent.

DATA3 defines whether this behaviour will be stored in flash memory as the default behaviour (DATA3 = 1).
For more information on the contents of the notification packet see chapter 2.1.2 and 2.1.3.
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.15 Setting the minutiae format (0x0E)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x0E	0x00	0x01	format	default

This packet sets the minutiae format used to transmit templates. Consequently it affects the size and format of the packets used for template transmission (see commands 0x30, 0x31 and 0x42).

DATA1:	0x00 = IDENCOM format (default)
	0x01 = DIN V66400 format
	0x02 = IDENCOM Compact format
	0x03 … 0xFF reserved

DATA2 defines whether this behaviour will be stored in flash memory as the default behaviour (DATA2 = 1).
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.16 Setting the responsiveness of the Infineon FingerTIP (0x0F) – *preliminary*

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x0F	0x00	0x01	threshold [0…7]	threshold [8…15]	default

This packet sets the responsiveness of the Infineon FingerChip to detect a finger. The default threshold is 32 (decimal).

DATA1:	threshold (bits 0–7)
DATA2:	threshold (bits 8–15)

DATA3 defines whether this behaviour will be stored in flash memory as the default behaviour (DATA3 = 1).
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.17 Activating the use of the external Single / Continuous-Mode switch (0x10)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x10	0x00	0x01	active	default

This packet activates the use of the external inport that controls the single and continuous mode for identification. Default value is active (DATA1 = 0x01). To deactivate the inport, set DATA1 to 0x00.
This function is only available in BioKey® 2103.

DATA1:	external inport activated / deactivated

DATA2 defines whether this behaviour will be stored in flash memory as the default behaviour (DATA2 = 1).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.18 Setting the buzzer durations (0x11)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x11	0x00	0x01	recognized	rejected	default

This packet controls the duration of the buzzer signal output. The durations are transmitted in steps of 10 ms, i.e. a value of 30 will cause the module to put out the signal for 300 ms. DATA1 carries the value for the buzzer signal duration after a successful recognition, DATA2 the signal duration after an unsuccessful recognition (rejection).
This function is only available in BioKey® 3000.

DATA1:	Duration of the buzzer signal after successful recognition
DATA2:	Duration of the buzzer signal after **un**successful recognition

DATA3 defines whether this behaviour will be stored in flash memory as the default behaviour (DATA3 = 1).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.19 Setting the noise mask threshold for Encoding (0x12)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x12	0x00	0x01	threshold	default

This packet sets the value of the noise mask threshold. This threshold is used to mask out noisy areas in the binary image created during the encode process.

Please do not change this value as long as you do not experience great difficulties during enrollment. If so, please check the binary image for large white areas within the fingerprint. When reducing the threshold value, less of the binary image will be masked out. The higher the threshold is, the larger these areas get. Default value is 26 for Atmel FingerChip sensor and 24 for Fingerprint Cards FPC1031 sensor. When using the Set Sensor command (0x00) the noise mask threshold will automatically be reset to default.

Please change this value with great care only.

This function is only available in BioKey® 3000.

DATA1: Noise mask threshold. Range of values: 10 … 40

DATA2 defines whether this behaviour will be stored in flash memory as the default behaviour (DATA2 = 1).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.20 Getting the noise mask threshold for Encoding (0x13)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x13	0x00	0x01

This packet gets the value of the noise mask threshold. This threshold is used to mask out noisy areas in the binary image created during the encode process.

Answer Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x13	0x01	0x00	threshold

This function is only available in BioKey® 3000.

DATA1: Noise mask threshold. Range of values: 10 … 40

2.2.21 Setting the configuration options for secure communication (0x14)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0006	0x14	0x00	0x01	active	autoextend

DATA3	DATA4	DATA5	DATA6
opentime [0..7]	opentime [8..15]	opentime [16..23]	opentime [24..31]

This packet sets the configuration options for the secure communication mode.

DATA1:	Flag indicating whether secure communication mode should be active.
DATA2:	Flag indicating whether the autoextend time feature should be active. If this is activated, every time a secured message cmd is received while the bus is in open state (after successful authentication), the time until the bus is closed again is reset to the opentime.
DATA3-DATA6:	Indicates the timeperiod in milliseconds for which the module accepts secured message cmds after a successful authentication. If the autoextend time feature is activated the timeperiod is renewed with every secured message cmd received. This is sent as a 32 bit unsigned Integer Value, LSB first. Note: To ensure, users do not accidentally lock themselves out by supplying a value of 0 or a very small value here, the module will not accept values less than 500 milliseconds and will reset these to 500 milliseconds instead.

If secure communication mode is activated most of the configuration commands are locked and can only be accessed for a certain time period after an authentication via password has taken place (see command 0x80).
If secure communication mode is active and authentication has not taken place these messages will be rejected by the low level communication layer with SEC_LOCK (0xcc) instead of an ACK (0x06).
The following is a list of message cmds that are locked and password protected when secure communication mode is active:

Message	**CMD**
SET_SENSOR_TYPE	0x00
SET_IMAGE_SIZE	0x01
SET_COMM_DEVICE	0x02
SET_BAUDRATE	0x03
SET_LED_USE	0x04
USE_DEBUG	0x05
SET_ADDRESS	0x06
SET_DELIVERY_SETTINGS	0x07
CONFIG_DOWNLOAD	0x09
CONTINUOUS_READ	0x0A
SET_THRESHOLDS	0x0B
USE_RELAIS	0x0C
SET_NOTIFICATION	0x0D
TEMPLATE_FORMAT	0x0E

INFINEON_VARIANCE	0x0F
CONT_SWITCH_ACTIVATE	0x10
BUZZER_DURATION	0x11
SET_NOISE_THRESHOLD	0x12
SET_SEC_CONFIG	0x14
SET_SEC_PASSWORD	0x16
ENROLL	0x20
CREATE_SUPERTEMPLATE	0x22
IDENTIFY_SIGNAL	0x23
DOWNLOAD_TEMPLATE	0x31
DOWNLOAD_TO_GREY_IMAGE	0x33
DELETE_FPT	0x41
DELETE_DATABASE	0x43
DELETE_ALL_FPTS	0x48
SOFTWARE_UPDATE	0x52
OUTPORT_CONTROL	0x53
WRITE_BYTES	0x57
ERASE_ALL	0x59

Note: When updating the Firmware of your BioKey module, the settings for secure communication mode will be reset to defaults (secure com mode disabled), including the Password (empty).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.2.22 Receiving the configuration options for secure communication (0x15)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x15	0x00	0x01

This packet requests the current configuration options for the secure communication mode to be sent by the module.

Answer Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0006	0x15	0x01	0x00	active	autoextend

DATA3	DATA4	DATA5	DATA6
opentime [0..7]	opentime [8..15]	opentime [16..23]	opentime [24..31]

DATA1:	Flag indicating whether secure communication mode is active.
DATA2:	Flag indicating whether the autoextend time feature is active. If this is activated, every time a secured message cmd is received while the bus is in

	open state (after successful authentication), the time until the bus is closed again is reset to the opentime.
DATA3-DATA6:	Indicates the timeperiod in milliseconds for which the module accepts secured message cmds after a successful authentication. If the autoextend time feature is activated the timeperiod is renewed with every secured message cmd received. This is sent as a 32 bit unsigned Integer Value, LSB first.

2.2.23 Setting the password for secure communication (0x16)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
variable	0x16	0x00	0x01	1. char

DATA2	DATAn
2. char	…

This message sets the Password for the secure communication mode. If secure communication is enabled, reception of secured message cmds is blocked by the module unless an authentication with the correct password has taken place first. (see message cmd 0x80)

DATA1-DATAn:	Password characters (max. 32 allowed), **no** zero termination

Please make sure you remember your password, once this is set and secure communication mode is activated there is no way to reset the password again without supplying the password you chose first.

Note: When updating the Firmware of your BioKey module, the settings for secure communication mode will be reset to defaults (secure com mode disabled), including the Password (empty).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR (password is longer than 32 characters)

2.2.24 Setting the current date and time (0x19)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x000A	0x19	0x00	0x01	Year[0..7]	Year[8..15]	Month

DATA4	DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
Day	DayOfWeek	Hour	Minute	Second	Millisecond [0..7]	Millisecond [8..15]

This message sets the Time and Date for the internal clock of the BioKey Module. The Module will respond with a standard answer packet with Status-Byte indicating OK or ERROR.

2.2.25 Setting the lock time window (0x1A)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x1A	0x00	0x01	Start time	End time

This message sets the lock time window of the BioKey Module. The Module will respond with a standard answer packet with Status-Byte indicating OK or ERROR.

DATA1:	Start time for permanent locking of the module, measured in quarter hours from midnight (e.g. 42 means 10:30 a.m.)
DATA2:	End time for permanent locking of the module, measured in quarter hours from midnight.

2.2.26 Setting the free time window (0x1B)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x1B	0x00	0x01	Start time	End time

This message sets the free time window of the BioKey Module. The Module will respond with a standard answer packet with Status-Byte indicating OK or ERROR.
In the given time interval any finger is accepted and will open the door.
Not every firmware supports this feature. Firmware composition code “0x29” needs to be set in order to enable the free time window.

DATA1:	Start time for permanent opening of the module, measured in quarter hours from midnight (e.g. 42 means 10:30 a.m.)
DATA2:	End time for permanent opening of the module, measured in quarter hours from midnight.

2.2.27 Getting the lock time window (0x1C)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x1C	0x00	0x01

This packet requests the current setting for the lock time window to be sent by the module.

Answer Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x1C	0x01	0x00	Start time	End time

DATA1:	Start time for permanent locking of the module, measured in quarter hours from midnight (e.g. 42 means 10:30 a.m.)
DATA2:	End time for permanent locking of the module, measured in quarter hours from midnight

2.2.28 Getting the free time window (0x1F)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x1F	0x00	0x01

This packet requests the current setting for the free time window to be sent by the module. Not every firmware supports this feature. Firmware composition code “0x29” needs to be set in order to enable the free time window.

Answer Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x1F	0x01	0x00	Start time	End time

DATA1:	Start time for permanent opening of the module, measured in quarter hours from midnight (e.g. 42 means 10:30 a.m.)
DATA2:	End time for permanent opening of the module, measured in quarter hours from midnight

2.2.29 Setting the time profiles (0x1D)

Initial packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0004	0x1D	0x00	0x01	#time profiles [0…7]	#time profiles [8…15]	time profiles per packet	overlap

This packet initiates a procedure during which the module receives time profiles. The time profiles will then be saved in flash. DATA1 and DATA2 contain the number of time profiles to be sent and DATA3 defines the number of time profiles in one packet.

The module responds with a standardized answer packet (see 2.1.1) to state whether the parameters are valid (OK) or not (ERROR), if valid, the time profile packets can be sent subsequently, otherwise the reception will be aborted.

Packet from terminal:

DATA1,2:	the number of time profiles to be sent
DATA3:	packet size of time profiles
DATA4:	if overlap is true, the previous time profiles in module will be overlapped.

Time profile packet(s):

LENGTH	CMD	SENDER	DEST	DATA1
0xXXXX	0x1D	0x00	0x01	time profiles per packet

DATA2	DATA3	DATA4	DATA5	DATA6	DATA7	DATA8	DATA9
ID[0…7]	ID[8…15]	Day1 [0…7]	Day1 [8…15]	Day2 [0…7]	Day2 [8…15]	Day3 [0…7]	Day3 [8…15]

DATA10	DATA11	DATA12	DATA13	DATA14	DATA15	DATA16	DATA17
Day4 [0…7]	Day4 [8…15]	Day5 [0…7]	Day5 [8…15]	Day6 [0…7]	Day6 [8…15]	Day7 [0…7]	Day7 [8…15]

DATA18	DATA19	DATA20	…	DATA (19 + 2*(Length1 -1))	DATA (19 + 2*(Length1 -1))
Length1	Special[0] [0…7]	Special[0] [8…15]	…	Special[Length1-1] [0…7]	Special[Length1-1] [8…15]

DATA (19 + 2*Length1)	DATA (19 + 2*Length1 + 1)	DATA (19 + 2*Length1 + 2)	DATA (19 + 2*Length1 + 3)
Validality [0…7]	Validality [8…15]	Validality [16…23]	Validality [24…31]

…

DATA1 specifies the number of time profiles in the particular packet, the following data bytes contain the time profile data.

The data section from DATA2 to DATA(19 + 2*Length1 + 3) repeats DATA1 times in this packet, but the length of each of this similar section in this packet can be variable because of different length of Special data.

Time profile packet from terminal:

DATA1:	number of time profiles in this packet	
DATA2:	time profil index (bits 7–0)	
DATA3:	time profil index (bits 15–8)	
DATA4:	Monday	(bits 7–0)
DATA5:	Monday	(bits 15–8)
DATA6:	Tuesday	(bits 7–0)
DATA7:	Tuesday	(bits 15–8)
DATA8:	Wednesday	(bits 7–0)
DATA9:	Wednesday	(bits 15–8)
DATA10:	Thursday	(bits 7–0)
DATA11:	Thursday	(bits 15–8)
DATA12:	Friday	(bits 7–0)
DATA13:	Friday	(bits 15–8)
DATA14:	Saturday	(bits 7–0)
DATA15:	Saturday	(bits 15–8)
DATA16:	Sunday	(bits 7–0)
DATA17:	Sunday	(bits 15–0)

Structure of usual days:

Bits (15-13)	bits (12-8)	bits (7-5)	bits (4-0)
start hour	start quarter	end hour	end quarter

DATA18: number of special days(for example, holidays)

DATA19: special day[0] (bits 7–0)
DATA20: special day[1] (bits 15–8)

Structure of special days:

bits (15–11)	bits (10–7)	bits (6–0)
day	month	year

…

DATA (19 + 2*Length(x)):	valid year,month,day(bits 7–0)
DATA (19 + 2*Length(x) + 1):	valid year,month,day(bits 15–8)
DATA (19 + 2*Length(x) + 2):	valid year,month,day(bits 23–16)
DATA (19 + 2*Length(x) + 3):	valid year,month,day(bits 31–24)

Structure of valid year, month, day (4 bytes):

bits (31–27)	bits (26–23)	bits (22-16)	bits (15–11)	bits (10–7)	bits (6–0)
start day	start month	start year	end day	end month	end year

For this bit limit, the period of the year will be reduced to less than $2^7 – 1$, moreover before transmission we set (year = year - 2000) in order to give a valid period.

The module finishes with a standardized answer packet (see 2.1.1) which includes the reception result.

2.2.30 Getting the time profiles (0x1E)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x1E	0x00	0x01	time profiles per packet

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x1E	0x01	0x00	#time profiles in module [0…7]	#time profiles in module [8…15]

Time profile packets:

LENGTH	CMD	SENDER	DEST	DATA1
0xXXXX	0x1E	0x01	0x00	time profiles per packet

DATA2	DATA3	DATA4	DATA5	DATA6	DATA7	DATA8	DATA9
ID[0…7]	ID[8…15]	Day1 [0…7]	Day1 [8…15]	Day2 [0…7]	Day2 [8…15]	Day3 [0…7]	Day3 [8…15]

DATA10	DATA11	DATA12	DATA13	DATA14	DATA15	DATA16	DATA17
Day4 [0…7]	Day4 [8…15]	Day5 [0…7]	Day5 [8…15]	Day6 [0…7]	Day6 [8…15]	Day7 [0…7]	Day7 [8…15]

DATA18	DATA19	DATA20	…	DATA (19 + 2*(Length1 -1))	DATA (19 + 2*(Length1 -1))
Length1	Special[0] [0…7]	Special[0] [8…15]	…	Special[Length1-1] [0…7]	Special[Length1-1] [8…15]

DATA (19 + 2*Length1)	DATA (19 + 2*Length1 + 1)	DATA (19 + 2*Length1 + 2)	DATA (19 + 2*Length1 + 3)
Validality [0…7]	Validality [8…15]	Validality [16…23]	Validality [24…31]

…

DATA bytes of this time profile packet is the same as *Time profile packets from Terminal* (see 2.2.25.). The whole time profile structure can be restored according to the algorithm mentioned in chapter 2.2.29 Setting the time profiles (0x1D).

2.3 Biometric functions

For additional information on the following commands please refer to chapters 3.1 and 3.2.

2.3.1 Enroll (0x20)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000D	0x20	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	source	destination

DATA13
Bus Mode

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x20	0x01	0x00	status	#minutiae	quality

This packet causes the module to add a new template to the flash database depending on DATA12 (0 = database and buffer FPT1, 1 = buffer FPT1 only, 2 = buffer FPT2 only, 3 = FPT_BUF). Depending on DATA11, the template data may have one of the following sources:
1) sensor image (DATA11 = 0x00):
The module will acquire a fingerprint image from the sensor and exctract its features. After processing the original image is located in GREY_IMAGE and the binary image in BIN_IMAGE. The generated template will be located either in FPT_BUF, FPT1, FPT2 or in FPT1 and the flash database (depending on DATA12 (destination)).
2) FPT_BUF (DATA11 = 0x01):
The template located in FPT_BUF will be stored either in FPT1, FPT2 or in FPT1 and the flash database (depending on DATA12 (destination)).
3) image from GREY_IMAGE (DATA11 = 0x02):
The module will exctract the features from the image in GREY_IMAGE and store the generated template in the flash database. After processing the binary image will be located in BIN_IMAGE. The generated template will be located either in FPT_BUF, FPT1, FPT2 or in FPT1 and the flash database (depending on DATA12 (destination)).

For storing the module uses the IDs transmitted in data bytes 1-10 (DATA1 .. DATA8: PID0..7, DATA9: FID, DATA10: AID) where their values are used in the following way:

1) PID = 0, FID = 0, AID = 0:	set PID, FID, AID automatically
2) PID > 0, FID = 0, AID = 0:	use PID, set FID and AID automatically
3) PID > 0, FID > 0, AID = 0:	use PID and FID, set AID automatically
4) PID > 0, FID > 0, AID > 0:	use PID, FID, AID

The Byte DATA13 determines the behaviour of the module concerning a delayed answer when acquiring the image from the sensor. If this Byte is set to 0 (standalone / point to point mode), the module will wait for the finger to be presented and then send an answer packet to the terminal containing the execution status in DATA1, the number of minutiae in the template in DATA2 and the image quality in DATA3. DATA2 and DATA3 are valid only if DATA1 is OK.

Status possible: OK, ERROR, BAD_IMAGE, BAD_FPT, DB_FULL, IDS_USED, NO_IMAGE

If the Byte DATA13 is set to 1 (bus mode) and the acquisition source is from sensor (DATA11), the module will answer immediately with a standard answer packet with Status OK and the result of the Enrollment procedure has to be acquired using the message cmd 0x24 (Poll Identify/Enroll Result). This is used when the module is used in a RS485 bus with other modules, where the communication on the bus would be disturbed by a delayed answer packet after enrolling from sensor.

2.3.2 Identification (0x21)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000D or 0x0019	0x21	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	source1	source2

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18	DATA19	DATA20
Bus Mode	Info2	PIDMask0	PIDMask1	PIDMask2	PIDMask3	PIDMask4	PIDMask5

DATA21	DATA22	DATA23	DATA24	DATA25
PIDMask6	PIDMask7	FIDMask	AIDMask	Info2Mask

Note: Databytes 14 to 25 are optional fields that are only used when DATA12 is 0x04 (Matching against a subset of fingerprints in the database), the LENGTH field takes values of 0x000D or 0x0019 respectively.

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000C	0x21	0x01	0x00	status	score	PID0	PID1

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID2	PID3	PID4	PID5	PID6	PID7	FID	AID

This packet causes the module to match a template with the database or single template. Depending on DATA11 (source1), the template data may have one of the following sources:
1) sensor image (DATA11 = 0x00):
The module will acquire a fingerprint image from the sensor and extract its features. After processing the original image is located in GREY_IMAGE, the binary image in BIN_IMAGE and the generated template in FPT1.
2) FPT_BUF (DATA11 = 0x01):
The template located in FPT_BUF will be used for matching.
3) image from GREY_IMAGE (DATA11 = 0x02):
The module will exctract the features from the image in GREY_IMAGE. After processing the binary image will be located in BIN_IMAGE and the generated template in FPT1.

The DATA12 Byte specifies the matching mode and can take one of the following values:

0	The source template is matched against the whole database.
1	Match only against buffer FPT1
2	Match only against buffer FPT2
3	Match only against database templates with Bit 0 of InfoByte 2 set to 1 (Note: this Mode is now obsolete and only retained for backward compatibility, use Mode 4 instead)
4	Match against selected database templates using the selection mask provided in the fields DATA15 to DATA25. When matching, each database templates PID, FID, AID and InfoByte2 is first AND-masked using the selection mask and then compared to the values provided in DATA1 to 10 and DATA14. Only templates where masked PID, FID, AID and InfoByte2 are equal to the DATA1 – 10 and DATA14 values are then matched. Example1, match only against templates where the lowest 3 Bits of the PID have the value 5: Supply value of 5 in DATA1 (PID0), rest of the PID, FID, AID and InfoByte2 fields get value 0. Then supply a value of 7 for the PIDMask0 Byte (DATA15) and the other mask bytes with 0. Example2, match only against templates where bit 0 of InfoByte 2 is set to 1 (old Mode 3 functionality): Set DATA1 – 10 to 0, DATA14 to 1 and set mask bytes DATA15 – 24 to 0 and DATA 25 to 1.

For matching in mode 0 the module uses the IDs transmitted in data bytes 1-10 (DATA1 .. DATA8: PID0..7, DATA9: FID, DATA10: AID) where their values are used in the following way:

1) PID = 0, FID = 0, AID = 0:	match with entire database
2) PID > 0, FID = 0, AID = 0:	match with all templates with PID
3) PID > 0, FID > 0, AID = 0:	match with all templates with PID and FID
4) PID > 0, FID > 0, AID > 0:	match with template with PID, FID and AID

After processing and matching with the entire database the best matching template will be located in FPT2.
After execution the module sends a packet containing the execution status in DATA1 and the matching score of the best match in DATA2. DATA3-DATA12 contain the IDs of the best matching template, but they will only be valid if the status is either RECOGNIZED or NOT_RECOGNIZED and source2 was 0 (matching with entire database).
Status possible: RECOGNIZED, NOT_RECOGNIZED, BAD_IMAGE, BAD_FPT, DB_EMPTY, IDS_UNKNOWN, NO_IMAGE

The Byte DATA13 determines the behaviour of the module concerning a delayed answer when acquiring the image from the sensor. If this Byte is set to 0 (standalone / point to point mode), the module will wait for the finger to be presented and then send an answer packet to the terminal as described above.
If the Byte DATA13 is set to 1 (bus mode) and the acquisition source is from sensor (DATA11), the module will answer immediately with a standard answer packet with Status OK and the result of the Identification procedure has to be acquired using the message cmd 0x24 (Poll Identify/Enroll Result). This is used when the module is used in a RS485 bus with other modules, where the communication on the bus would be disturbed by a delayed answer packet after identifying from sensor.

2.3.3 Identification with signalling (0x23)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000E	0x23	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12	DATA13
PID4	PID5	PID6	PID7	FID	AID	source1	source2	signals

DATA14
Bus Mode

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000C	0x21	0x01	0x00	status	score	PID0	PID1

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID2	PID3	PID4	PID5	PID6	PID7	FID	AID

This packet causes the module to match a template with the database or single template. Depending on DATA11 (source1), the template data may have one of the following sources:
1) sensor image (DATA11 = 0x00):
The module will acquire a fingerprint image from the sensor and exctract its features. After processing the original image is located in GREY_IMAGE, the binary image in BIN_IMAGE and the generated template in FPT1.
2) FPT_BUF (DATA11 = 0x01):
The template located in FPT_BUF will used for matching.
3) image from GREY_IMAGE (DATA11 = 0x02):
The module will exctract the features from the image in GREY_IMAGE. After processing the binary image will be located in BIN_IMAGE and the generated template in FPT1.

The source template is matched against the database (DATA12 = 0), buffer FPT1 (DATA12 = 1) , buffer FPT2 (DATA12 = 2) or specally marked templates from the database (DATA12 = 3). If DATA12 = 3 the source template is matched against these template from the database where the LSB Bit (Bit 0) from the Infobyte 2 id 1. So it is possible to mark a set of templates from the database for identification.

For matching the module uses the IDs transmitted in data bytes 1-10 (DATA1 .. DATA8: PID0..7, DATA9: FID, DATA10: AID) where their values are used in the following way:

1) PID = 0, FID = 0, AID = 0:	match with entire database
2) PID > 0, FID = 0, AID = 0:	match with all templates with PID
3) PID > 0, FID > 0, AID = 0:	match with all templates with PID and FID
4) PID > 0, FID > 0, AID > 0:	match with template with PID, FID and AID

After processing and matching with the entire database the best matching template will be located in FPT2.
The module may report the execution status using its signalling outports. To activate this, the corresponding bits in DATA13 have to be set. The durations for relay and buzzer have to be set in advance (see chapter 2.2).

DATA13 bit 0 set:	LEDs will report execution status
DATA13 bit 1 set:	Relay will become active when status is RECOGNIZED
DATA13 bit 2 set:	Buzzer (if available) will become active
DATA13 bit 3 set:	Wiegand telegram (if available) will be sent

After execution the module sends a packet containing the execution status in DATA1 and the matching score of the best match in DATA2. DATA3-DATA12 contain the IDs of the best matching template, but they will only be valid if the status is either RECOGNIZED or NOT_RECOGNIZED and source2 was 0 (matching with entire database).
Status possible: RECOGNIZED, NOT_RECOGNIZED, BAD_IMAGE, BAD_FPT, DB_EMPTY, IDS_UNKNOWN, NO_IMAGE

The Byte DATA14 determines the behaviour of the module concerning a delayed answer when acquiring the image from the sensor. If this Byte is set to 0 (standalone / point to point mode), the module will wait for the finger to be presented and then send an answer packet to the terminal as described above.
If the Byte DATA14 is set to 1 (bus mode) and the acquisition source is from sensor (DATA11), the module will answer immediately with a standard answer packet with Status OK and the result of the Identification procedure has to be acquired using the message cmd 0x24 (Poll Identify/Enroll Result). This is used when the module is used in a RS485 bus with other modules, where the communication on the bus would be disturbed by a delayed answer packet after identifying from sensor.

2.3.4 Create Supertemplate (0x22)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x001F	0x22	0x00	0x01	fromSensor	PID1_0	PID1_1	PID1_2

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID1_3	PID1_4	PID1_5	PID1_6	PID1_7	FID1	AID1	PID2_0

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18	DATA19	DATA20
PID2_1	PID2_2	PID2_3	PID2_4	PID2_5	PID2_6	PID2_7	FID2

DATA21	DATA22	DATA23	DATA24	DATA25	DATA26	DATA27	DATA28
AID2	PID3_0	PID3_1	PID3_2	PID3_3	PID3_4	PID3_5	PID3_6

DATA29	DATA30	DATA31
PID3_7	FID3	AID3

This packet causes the module to create a Supertemplate from three templates. As the Supertemplate contains features from all three templates it will only be created if the templates match.
The templates can either be acquired from sensor or be taken from flash database, which is specified in DATA1. If DATA1 = 1, three fingerprints are successively read from sensor and encoded to templates. The intermediate results are transmitted to terminal as standardized answer packets (see 2.1.1). Possible results are OK, NO_IMAGE, BAD_IMAGE, BAD_FPT, NO_MATCH. If the module sends ERROR anytime, the procedure is aborted.
Otherwise (DATA1 = 0), the three templates specified by DATA2 - DATA11 (template 1), DATA12 - DATA21 (template 2) and DATA22 - DATA31 (template 3) are taken from flash database. If no such templates are stored, the module sends an answer packet with status ERROR.
If the three templates do not match, the module also sends an answer packet with status ERROR.

After generation the Supertemplate will be located in FPT_BUF, so it can be stored in the database or transmitted to the terminal.

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.3.5 Poll identify/enroll result (0x24)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x24	0x00	0x01

This packet requests the result of the last identification/enroll process from the module. If the module is in Continuous Identification Mode (see 2.2.11) it will store the results of the last identification process, which can then be retrieved using this message. Also when using the message cmds for enroll/identification with the bus mode flag set, the results can be retrieved with this message. Note that only the result of the last identification/enrollment is stored, overwriting the results of the previous process. To clear the result data after they have been transmitted to the terminal successfully, use message cmd 0x25.

Answer Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000E	0x24	0x01	0x00	status	score	PID0	PID1

DATA5		DATA10	DATA11	DATA12	DATA13	DATA14
PID2	…	PID7	FID	AID	#minutiae	quality

DATA1	status of the identification, this can be RECOGNIZED, NOT_RECOGNIZED, FPT_ENROLLED, TIMEOUT or NO_IMAGE (no identification has taken place since last module reset or clear result message 0x25, see 2.3.6)
DATA2	matching score (only valid if last operation was an identification attempt)
DATA3 – DATA10	PID
DATA11	FID
DATA12	AID
DATA13	number of minutiae of the newly enrolled template (enroll operation) or of the template that matched with the highest matching score (identify operation)
DATA14	image quality of the newly enrolled template (enroll operation) or of the template that matched with the highest matching score (identify operation)

2.3.6 Clear identify/enroll result (0x25)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x25	0x00	0x01

This packet clears the result data of the last identification/enroll process in the module. See 2.3.5.

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.3.7 Enroll with Matching (0x26)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000D	0x26	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	source	destination

DATA13
match thrsld

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000E	0x26	0x01	0x00	status	#minutiae	Quality	PID0

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID1	PID2	PID3	PID4	PID5	PID6	PID7	FID

DATA13	DATA14
AID	Score

This packet is an alternative version of cmd 0x20 (Enroll) and will perform a matching of the newly acquired fingerprint template against the database. This can be used to prevent persons from enrolling multiple times with the same finger under different PIDs.
Also possible similarities between fingers of different persons can be found this way, so the newly enrolling person should then use a different finger.
The parameter match thrsld defines the matching threshold that should be used to match against the database templates. If a database template is found that matches the newly acquired template with a matching score higher or equal to match thrsld, the status in the answer packet will be FPT_EXISTS, the PID/FID/AID of the matched database template will be returned in the answer packet and the newly acquired template will not be stored. In this case DATA14 will hold the matching score. Supplying a 0 for match thrsld will make the module use the default matching threshold, as configured with the parameter security lvl in message cmd 0x09.
If no template in the database matches the newly acquired one, the new template will be stored according to the PID/FID/AID supplied (See Enroll cmd 0x20) and the PID/FID/AID of the new template will be returned in the answer packet.

Depending on the PID/FID/AID supplied some database templates will be excluded from the matching:

PID = 0 / FID = 0 / AID = 0	matching against all templates in the database
PID > 0 / FID = 0 / AID = 0	matching against all templates in the database
PID > 0 / FID > 0 / AID = 0	matching against all templates in the database except those with the same PID/FID (used for creating new alternatives for the same finger of a person)
PID > 0 / FID > 0 / AID > 0	matching against all templates in the database except those with the same PID and FID (used for creating new alternatives for the same finger of a person with supplying the AID)

See cmd 0x20 (Enroll) for further explanation of the parameters src, dst and PID/FID/AID.

Status possible: OK, ERROR, BAD_IMAGE, BAD_FPT, DB_FULL, IDS_USED, NO_IMAGE, FPT_EXISTS

2.3.8 Enroll best of three (0x27)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000B	0x27	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11
PID4	PID5	PID6	PID7	FID	AID	Destination

This command will acquire 3 Fingerprints from the Sensor first and then the one with the highest minutiae * quality product is enrolled according to the setting of the destination Byte in DATA11 (see command Enroll for further explanation of possible destination values).

Module answer packet:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x27	0x01	0x00	status	#minutiae	quality

The module will send the above message after successful acquisition of the first and second fingerprints from the sensor where status will be FPT_READ (0x90).

After successful acquisition of the third fingerprint the same message is sent but with a status, indicating the overall result of the enroll procedure:

FPT_ENROLLED	This indicates, the best of the three acquired templates was successfully enrolled.
BAD_FPT	The best of the three acquired templates did not meet the minimum minutiae or quality requirements, no fingerprint template was enrolled. The person should try enrolling a different finger.

The minutiae and quality bytes will hold the values of the enrolled fingerprint template.

Timeout Condition:

Should a timeout (no finger is presented within the configured sensor timeout time) occur during acquisition of one of the three fingerprints the whole procedure is aborted and a message as above will be sent with the status byte indicating TIMEOUT (0x8F). No fingerprint will be enrolled in this case and the procedure has to be started again.

2.3.9 Enroll best of three using matching (0x29)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000D	0x29	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12	DATA13
PID4	PID5	PID6	PID7	FID	AID	Source	Destination	match thrsld

This command will acquire three templates first and then the one which has the best matching score or the highest minutiae * quality product when matching score is equal is enrolled according to the setting of the Destination Byte in DATA12 (see command Enroll for further explanation of possible destination values).

Depending on DATA11, the template data may have one of the following sources:

1) sensor image (DATA11 = 0x00):
The module will acquire a fingerprint image from the sensor and exctract its features. After processing the original image is located in GREY_IMAGE and the binary image in BIN_IMAGE. The generated template will be located either in FPT_BUF ,FPT1, FPT2 or in FPT1 and the flash database depending on DATA12 (destination).

2) FPT (DATA11 = 0x01):
The templates to be matched located in FPT_BUF, FPT1 and FPT2 can be obtained in the following ways:

i. Enroll(0x20) from sensor for three times and save each FPT to FPT_BUF, FPT1 or FPT2.
ii. Upload three FPTs using the command “Receive templates from terminal (0x31)” to FPT_BUF, FPT1 and FPT2.
iii. Upload Grey Image (***only generated by BioKey Module***) using Receive images from terminal(0x33) and Enroll(0x20). Repeat this procedure three times and save each template to FPT_BUF, FPT1 or FPT2. In Enroll command set Destination = 1,2,3 to save the templates in FPT1, FPT2, FPT_BUF.

The best template will be stored either in FPT1 and the flash database, FPT2, FPT1 or in FPT_BUF depending on DATA12 (see command Enroll for further explanation of possible destination values).

The parameter match thrsld defines the matching threshold that should be used to match against the three enrolled templates. Supplying a 0 for match thrsld will make the module use the default matching threshold, as configured with the parameter security level in the module configuration (see command “Receiving configuration from the module (0x08)”).

Module answer packet:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x29	0x01	0x00	status	#minutiae	quality

If DATA11 is set to 0x00 (source = sensor image) the module will send the above message after each successful acquisition of the three fingerprints ***from the sensor*** where status will be FPT_READ (0x90).

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0006	0x29	0x01	0x00	status	#minutiae	Quality	Score12

DATA5	DATA6	DATA7
Score23	Score13	Best of three index

After successful acquisition of the third fingerprint another similar message is finally sent.
If DATA11 is set to 0x01 (source = FPT buffers) then only this final packet is received.
In this packet three matching scores are included to give a description of the matching results.
The fields DATA2 and DATA3 show the number of minutiae and the quality of the best template. In addition you find the index (1 → FPT1 or 2 → FPT2 or 3 → FPTBuffer) of the best matching template that is actually enrolled.
The status indicates the overall result of the enroll procedure:

FPT_ENROLLED	This indicates, the best of the three acquired templates was successfully enrolled.
BAD_FPT	The best of the three acquired templates did not meet the minimum minutiae or quality requirements, no fingerprint template was enrolled. The person should try enrolling a different finger.
NO_MATCH	This indicates, any of the three matching(FPT1-FPT2, FPT1-FPT3, FPT2-FPT3) does not meet the match thrsld. For example, three different fingers are used for this command in spite of good quality and minutiae.

Timeout Condition:

Should a timeout (no finger is presented within the configured sensor timeout time) occur during acquisition of one of the three fingerprints the whole procedure is aborted and a message as above will be sent with the status byte indicating TIMEOUT (0x8F). No fingerprint will be enrolled in this case and the procedure has to be started again.

2.4 Image and template transmission

For additional information on the following commands please refer to chapters 3.1 and 3.2.

2.4.1 Transmit template to the terminal (0x30)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x30	0x00	0x01	FPT	no. of minutiae per packet

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0011	0x30	0x01	0x00	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	quality	#minutiae

DATA13	DATA14	DATA15	DATA16	DATA17
Info1	Info2	Info3	Info4	no. of packets

Minutiae packets (IDENCOM format):

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0xXXXX	0x30	0x01	0x00	#minutiae	x[0…7]	x[8…15]	y[0…7]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
y[8…15]	w[0…7]	w[8…15]	value	nx[0…7]	nx[8…15]	ny[0…7]	ny[8…15]

…

Minutiae packets (DIN V66400 format):

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0xXXXX	0x30	0x01	0x00	#minutiae	x	y	w

…

Minutiae packets (IDENCOM Compact format)

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0xXXXX	0x30	0x01	0x00	#minutiae	x	y	w[0…7]

DATA5	DATA6	DATA7	…
w[8…15]	nx	ny	

This packet causes the module to transmit the FPT chosen in DATA1 to the terminal. The module's answer packet contains information about the template chosen. After this packet has been acknowledged, the module begins to transmit the minutiae packets with the number of minutiae demanded in DATA2 of the terminal's packet.
If the terminal's command contains invalid parameters, the module will respond with the standardized answer packet (see 2.1.1) with status ERROR and the execution will be aborted.
For further information on the template structure please refer to chapter 3.2.

Packet from terminal:

DATA1:	FPT = 0x00	>> FPT_BUF is sent
	FPT = 0x01	>> FPT1 is sent
	FPT = 0x02	>> FPT2 is sent
DATA2:	demanded number of minutiae per packet (0 = header packet only, no minutiae packet(s))	

Packet from module:

DATA1 … DATA8:	PID (PID0 … PID7)
DATA9:	FID
DATA10:	AID
DATA11:	image quality
DATA12:	number of minutiae in template
DATA13:	Info byte 1 (reserved)
DATA14:	Info byte 2 (reserved)
DATA15:	Info byte 3 (customer specific)
DATA16:	Info byte 4 (customer specific)

DATA17:	number of packets needed for template transmission

Minutiae packet(s) from module (IDENCOM format):

DATA1:	number of minutiae in packet
DATA2:	x position (bits 0–7)
DATA3:	x position (bits 8–15)
DATA4:	y position (bits 0–7)
DATA5:	y position (bits 8–15)
DATA6:	angle (bits 0–7)
DATA7:	angle (bits 8–15)
DATA8:	value
DATA9:	neighbour x position (bits 0–7)
DATA10:	neighbour x position (bits 8–15)
DATA11:	neighbour y position (bits 0–7)
DATA12:	neighbour y position (bits 8–15)

…

Minutiae packet(s) from module (DIN V66400 format):

DATA1:	number of minutiae in packet
DATA2:	x position
DATA3:	y position
DATA4:	angle (the two most significant bits indicate minutiae type)

…

Minutiae packet(s) from module (IDENCOM Compact format):

DATA1:	number of minutiae in packet
DATA2:	x position
DATA3:	y position
DATA4:	angle (bits 0–7)
DATA5:	angle (bits 8–15)
DATA6:	neighbour x position
DATA7:	neighbour y position

…

2.4.2 Receive template from terminal (0x31)

Initial packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0012	0x31	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	quality	#minutiae total

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18
#minutiae per packet	Info1	Info2	Info3	Info4	buffer

This packet initiates a procedure during which the module receives a template. The template will then be located in one of the module's template buffers FPT_BUF, FPT1 or FPT2 or in database depending on DATA18. Data bytes 1 to 17 contain the template header and the number of minutiae that will be sent in the following packets.

The module responds with a standardized answer packet (see 2.1.1) to state whether the header parameters are valid (OK) or not (ERROR), if valid, the minutiae packets can be sent subsequently, otherwise the reception will be aborted.

Packet from terminal:

DATA1 … DATA8:	PID (PID0 … PID7)
DATA9:	FID
DATA10:	AID
DATA11:	image quality
DATA12:	number of minutiae in template
DATA13:	number of minutiae per packet
DATA14:	Info byte 1 (reserved)
DATA15:	Info byte 2 (reserved)
DATA16:	Info byte 3 (customer specific, arbitrary value)
DATA17:	Info byte 4 (customer specific, arbitrary value)
DATA18:	0x00: template will be put into FPT_BUF 0x01: template will be put into FPT1 0x02: template will be put into FPT2 0x03: template will be put into database

Minutiae packet(s) (IDENCOM format):

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0xXXXX	0x31	0x00	0x01	#minutiae	x[0…7]	x[8…15]	y[0…7]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
y[8…15]	w[0…7]	w[8…15]	value	nx[0…7]	nx[8…15]	ny[0…7]	ny[8…15]

…

Minutiae packet(s) (DIN V66400 format):

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0xXXXX	0x31	0x00	0x01	#minutiae	x	y	w

…

Minutiae packet(s) (IDENCOM Compact format):

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0xXXXX	0x31	0x00	0x01	#minutiae	x	y	w

DATA5	DATA6	…
nx	ny	

DATA1 specifies the number of minutiae in the particular packet, the following data bytes contain the minutiae data.

minutiae packet from terminal (IDENCOM format):

DATA1:	number of minutiae in this packet
DATA2:	x position (bits 0–7)

DATA3:	x position (bits 8–15)
DATA4:	y position (bits 0–7)
DATA5:	y position (bits 8–15)
DATA6:	angle (bits 0–7)
DATA7:	angle (bits 8–15)
DATA8:	value
DATA9:	neighbour x position (bits 0–7)
DATA10:	neighbour x position (bits 8–15)
DATA11:	neighbour y position (bits 0–7)
DATA12:	neighbour y position (bits 8–15)

…

Minutiae packet(s) from terminal (DIN V66400 format):

DATA1:	number of minutiae in this packet
DATA2:	x position
DATA3:	y position
DATA4:	angle (the two most significant bits indicate minutiae type)

…

Minutiae packet(s) from terminal (IDENCOM Compact format):

DATA1:	number of minutiae in packet
DATA2:	x position
DATA3:	y position
DATA4:	angle
DATA5:	neighbour x position
DATA6:	neighbour y position

…

For further information on the template structure please refer to chapter 3.2.
The module finishes with a standardized answer packet (see 2.1.1) which includes the reception result.

2.4.3 Transmit image to terminal (0x32)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x32	0x00	0x01	IMAGE

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x32	0x01	0x00	w[0…7]	w[8…15]	h[0…7]	h[8…15]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
#pac[0…7]	#pac[8…15]	xdpi[0…7]	xdpi[8…15]	ydpi[0…7]	ydpi[8…15]

DATA1:	IMAGE = 0x00	>> GREY_IMAGE is sent
	IMAGE = 0x01	>> BIN_IMAGE is sent
	IMAGE = 0x02	>> BIN_IMAGE is sent in fast mode (8 pixels per byte)

This packet causes the module to transmit the image specified in DATA1 to the terminal. The first packet from the module (s.a.) contains information about the image size (width: DATA1 and DATA2, height: DATA3 and DATA4) and the number of packets needed to transmit the image data (DATA5 and DATA6). DATA7/8 contains the x resolution of the image, DATA9/10

the y resolution, both values in dpi. The least significant bits of these parameters are transmitted first.
After the terminal has acknowledged this packet, the module transmits the packets that contain the image data. The image data is transmitted in rows from upper left to lower right.
In fast mode (IMAGE = 0x02), which is supported from software version 3.6 (BioKey® 2103) or 1.5 (BioKey® 3000), each of the bytes in an image data packet contains 8 binary pixels. To ease the reconstruction of an 8-bit-based image, every packet contains the data of one image row (for an image with 200 rows, 200 image packets will be transmitted). The LSB of each byte carries the leftmost of the current 8 pixels, whereas the MSB carries the rightmost one. If a bit's value is "1", the corresponding image value is "0xFF", whereas a bit's value of "0" states the pixel color "0x00".
After having received the acknowledge for the last of those packets, the module sends the standardized answer packet (see 2.1.1).
If the terminal's packet contains an invalid parameter value (IMAGE > 2), the standardized answer packet with status ERROR will be sent instead of the packet containing the image size, and the execution will be aborted
Status possible: OK, ERROR

2.4.4 Receive image from terminal (0x33)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x33	0x01	0x00	w[0…7]	w[8…15]	h[0…7]	h[8…15]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
#pac[0…7]	#pac[8…15]	xdpi[0…7]	xdpi[8…15]	ydpi[0…7]	ydpi[8…15]

This packet causes the module to receive an image from the terminal. DATA1/DATA2 contain the image width, DATA3/DATA4 contain the image height. DATA5/DATA6 tell the module the number of packets needed to transmit the image. DATA7/8 contains the x resolution of the image, DATA9/10 the y resolution, both values in dpi.
The module responds with the standardized answer packet (see 2.1.1) to state whether the terminal's packet contained valid image size parameters (status OK) or not (status ERROR). If so, the execution will be aborted. Otherwise the terminal transmits the packets containing the image data. The image data is transmitted in rows from upper left to lower right and stored in GREY_IMAGE.
After having acknowledged the last of those packets, the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.4.5 Get sensor image and transmit it to the terminal (0x34)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
0x0000	0x34	0x00	0x01	IMAGE

DATA1:	IMAGE = 0x00	>> GREY_IMAGE is sent
	IMAGE = 0x01	>> BIN_IMAGE is sent in fast mode (8 pixels per byte)

The info field DATA1 is supported from protocol version 6.8 (**in BioKey4080**)

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x34	0x01	0x00	w[0…7]	w[8…15]	h[0…7]	h[8…15]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
#pac[0…7]	#pac[8…15]	xdpi[0…7]	xdpi[8…15]	ydpi[0…7]	ydpi[8…15]

This packet causes the module to acquire a fingerprint image from the sensor and transmit it to the terminal subsequently. In fast mode (IMAGE = 0x01), which is supported from protocol version 6.8, each of the bytes in an image data packet contains 8 binary pixels.

After acquiring the image, the module sends the standardized answer packet (see 2.1.1) to state whether acquisition was successful (status = OK or ERROR).
If successful, the module sends another packet (s.a.) containing the image size (width: DATA1 and DATA2, height: DATA3 and DATA4) and the number of packets needed to transmit the image data (DATA5 and DATA6). DATA7/8 contains the x resolution of the image, DATA9/10 the y resolution, both values in dpi. The least significant bits of these parameters are transmitted first.
After the terminal has acknowledged this packet, the module transmits the packets that contain the image data. The image data is transmitted in rows from upper left to lower right.
After having received the acknowledge for the last of those packets, the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.5 Database commands

2.5.1 Copy template from database into template buffer (FPT_BUF) (0x40)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x40	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
PID4	PID5	PID6	PID7	FID	AID

This packet causes the module to copy the template specified with DATA1..DATA10 into the template buffer FPT_BUF.

DATA1 … DATA8:	PID (PID0 … PID7)
DATA9:	FID
DATA10:	AID

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR, DB_EMPTY, IDS_UNKNOWN

2.5.2 Delete templates from database (0x41)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x41	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
PID4	PID5	PID6	PID7	FID	AID

This packet causes the module to tag templates specified with DATA1..DATA10 as “deleted”.

DATA1 … DATA8:	PID (PID0 … PID7)
DATA9:	FID
DATA10:	AID

The data bytes are used in the following way:

1) PID = 0, FID = 0, AID = 0:	delete all templates (compare section 2.5.4, command 0x43)
2) PID > 0, FID = 0, AID = 0:	delete all templates with given PID
3) PID > 0, FID > 0, AID = 0:	delete all templates with given PID and FID
4) PID > 0, FID > 0, AID > 0:	delete template with given PID, FID, AID

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR, DB_EMPTY, IDS_UNKNOWN

2.5.3 Transmit entire database to the terminal (0x42)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x42	0x00	0x01	#minutiae per packet

DATA1: number of minutiae that are to be sent per packet

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x42	0x01	0x00	#FPT[0…7]	#FPT[8…15]

DATA1:	number of templates stored in database (bits 0–7)
DATA2:	number of templates stored in database (bits 8–15)

This packet causes the module to transmit the entire database to the terminal. After having sent the answer packet stating the number of templates stored in database and having received the acknowledge for this packet, the module transmits the packets which contain the template data. Templates may be sent using several packets according to the “number of minutiae per packet“ demanded by the terminal. The template packets’ structure is equal to the one used for the command “2.4.1 Transmit template to the terminal”, except for the command byte carrying the value 0x42.
The module also sends the standardized answer packet (see 2.1.1), either after the last template packet with status OK or after the terminal’s packet with status ERROR, if a parameter error occurred.
Status possible: OK, ERROR

2.5.4 Delete entire database (0x43)

There are 2 versions of this command.

First version: Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x43	0x00	0x01	fast

This packet causes the module to delete the entire database. Depending on DATA1 it is either deleted physically (DATA1 = 0) - which can take up to 30 seconds depending on the database size - or all templates are tagged as “deleted” (DATA1 = 1) which is a lot faster.

Second version: Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x43	0x00	0x01	fast	Keep module address

Additionally to the first version in this case the configuration will be set to delivery state. The module address is set to default address 1 (DATA2 = 0) or the current module address is kept (DATA2 = 1).

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.5.5 Remove deleted templates from database (defrag) (0x44)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x44	0x00	0x01

This packet causes the module to delete all templates that are tagged as deleted from the database. This function does not have to be called as it is invoked automatically when too many templates are tagged as deleted.
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.5.6 Transmit database table of contents to the terminal (0x45)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x45	0x00	0x01	#entries per packet

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x45	0x01	0x00	#packets [0…7]	#packets [8…15]

Table of Contents Packets from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
n	0x45	0x01	0x00	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID4	PID5	PID6	PID7	FID	AID	pos[0…7]	pos[8…15]

…	DATA*n*–11	DATA*n*–10	DATA*n*–9	DATA*n*–8	DATA*n*–7	DATA*n*–6	DATA*n*–5
…	PID0	PID1	PID2	PID3	PID4	PID5	PID6

DATA*n*–4	DATA*n*–3	DATA*n*–2	DATA*n*–1	DATA*n*
PID7	FID	AID	pos[0…7]	pos[8…15]

This packet causes the module to transmit the table of contents of the database. It contains the desired number of entries per packet. Each entry has a size of 12 bytes. The module replies a packet stating the number of packets needed to transmit the data. As the packet length is limited to 1024 data bytes, max. 85 entries fit one packet. Transmitted data is PID, FID, AID and position of all templates stored.

2.5.7 Get free IDs (0x46)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x46	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
PID4	PID5	PID6	PID7	FID	AID

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x46	0x01	0x00	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
PID4	PID5	PID6	PID7	FID	AID

This packet causes the module to compute the smallest free IDs. The information transmitted in data bytes 1-10 (DATA1 … DATA8: PID, DATA9: FID, DATA10: AID) determines how these IDs are found:

1) PID = 0, FID = 0, AID = 0:	get free PID, FID, AID
2) PID > 0, FID = 0, AID = 0:	get free FID und AID for the given PID
3) PID > 0, FID > 0, AID = 0:	get free AID for given PID and FID

The module's answer packet contains the computed IDs in the same order as in the terminal's packet. The results mean the following:

1) PID > 0, FID > 0, AID > 0:	got free PID, FID, AID
2) PID = 0, FID = 0, AID = 0:	all PIDs used
3) PID > 0, FID = 0, AID = 0:	all FIDs used for PID
4) PID > 0, FID > 0, AID = 0:	all AIDs used for PID / FID combination

2.5.8 Is PID in database (0x47)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0008	0x47	0x00	0x01	PID0	PID1	PID2	PID3

DATA5	DATA6	DATA7	DATA8
PID4	PID5	PID6	PID7

This packet causes the module to determine whether the PID transmitted in data bytes 1-8 is used in the database already.
After execution the module sends the standardized answer packet (see 2.1.1) where an OK states that the PID is used already whereas an ERROR states the opposite.
Status possible: OK, ERROR

2.5.9 Get Number of Templates marked for Identification (0x49)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x49	0x00	0x01

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x49	0x01	0x00	num[0...7]	num[8...15]

This packet retrieves the number of Templates in the module that have the Bit0 of Info Byte 2 set to 1, meaning they are marked for Identification when a destination value of 3 is used in the Identification message (0x21).

2.5.10 Get Access Log (0x4A)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x4A	0x00	0x01	numpackage	num[0..7]	num[8..15]

This packet retrieves a log of successful accesses from the module.

DATA1	Number of entries the module should send in each data packet
DATA2..3	Number of entries to retrieve from the module, the module will always sent the latest/newest entries first. The maximum number of entries stored in the module is 3000 after that the oldest entries are overwritten.

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2
0x0002	0x4A	0x01	0x00	num[0..7]	num[8..15]

DATA1..2 Number of entries currently stored in the module.

Subsequently the module will send data packets with the actual entries of the access logs:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0002	0x4A	0x01	0x00	numentries	door[0..7]	door[8..15]	PID0

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10	DATA11	DATA12
PID1	PID2	PID3	PID4	PID5	PID6	PID7	FID

DATA13	DATA14	DATA15	DATA16	DATA17	DATA18	DATA19	DATA20
AID	Year[0..7]	Year[8..15]	Month	Day	DayOfWeek	Hour	Minute

DATA21	DATA22	..
Second	Door[0..7]	..

DATA1	Number of log entries in this packet
DATA2..3	Door Number, this is currently the Address of the Biokey Module.
DATA4..11, 12, 13	PID, FID and AID of the person who gained access
DATA14..21	Date and Time of the access

2.6 Other functions

2.6.1 LifeCheck (0x50)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x50	0x00	0x01

Status packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x000A	0x50	0x01	0x00	version1	version2	#all[0…7]	#all[8…15]

DATA5	DATA6	DATA7	DATA8	DATA9	DATA10
#free[0…7]	#free[8…15]	#saved[0…7]	#saved[8…15]	#deleted[0…7]	#deleted[8…15]

This packet causes the module to send a status packet containing the following 9 data bytes:

DATA1:	software version 1 - Protocol version (the implemented protocol version)
DATA2:	software version 2 - Revision number (used to identify small changes between firmware with the same core version and protocol version)
DATA3:	overall number of slots (database capacity) (bits 0…7)
DATA4:	overall number of slots (database capacity) (bits 8…15)
DATA5:	number of free slots (bits 0…7)
DATA6:	number of free slots (bits 8…15)
DATA7:	number of templates stored (bits 0…7)
DATA8:	number of templates stored (bits 8…15)
DATA9:	number of templates tagged as deleted (bits 0…7)
DATA10:	number of templates tagged as deleted (bits 8…15)

Please notice that DATA1 and DATA2 only contain the Protocol and Revision Version of the Firmware. To get the complete Version number including the Core Version please use the command “2.6.14 Get Version Number (0x62)”.

2.6.2 Module reset (0x51)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x51	0x00	0x01

This packet causes the module to reset and reboot after acknowledging the packet.

2.6.3 Module software update (0x52)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0005	0x52	0x00	0x01	#[0…7]	#[8…15]	size[0…7]	size[8…15]

DATA5
size[16…23]

This packet causes the module to receive new software for a software update. DATA1/DATA2 contain the number of packets needed to transmit, whereas DATA3/DATA4/DATA5 state the size of the program code in bytes.

DATA1:	number of subsequent packets (bits 0–7)
DATA2:	number of subsequent packets (bits 8–15)
DATA3:	size of program code in bytes (bits 0–7)
DATA4:	size of program code in bytes (bits 8–15)
DATA5:	size of program code in bytes (bits 16–23)

Packet from module when ready for receiving the updated software:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x52	0x01	0x00	OK

Packet from module if update process cannot be started:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x52	0x01	0x00	ERROR

If the module sent the OK packet, the terminal transmits the program code. Finally the terminal transmits a packet containing the CRC-32 checksum. After updating flash memory the module sends the standardized answer packet (see 2.1.1, status possible: OK, ERROR), resets and reboots.

2.6.4 Controlling the Outports (0x53)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0009	0x53	0x00	0x01	dura. 0-7	dura 8-15	high 0-7	high 8-15

DATA5	DATA6	DATA7	DATA8	DATA9
low 0-7	low 8-15	outport 0-7	outport 8-15	function

This packet controls the module's outports. Currently the module has 4 outports, but for future use up to 16 outports can be controlled with this command. The outports' functionality can be configured in various ways: An outport can be set to "1" or "0" and it can blink with particular high and low times with a defined startup value. This behaviour can be started for a particular duration or enduring. These functions are controlled with the data bytes 1-9:

DATA1/2:	duration (bits 0-7 / 8-15) in ms controls, how long an outport will be active (0 = unlimited)
DATA3/4:	high phase duration (bits 0-7 7 8-15) in ms controls, how long an outport will remain "1" in blink mode (0 = no blinking)
DATA5/6:	low phase duration (bits 0-7 7 8-15) in ms controls, how long an outport will remain "0" in blink mode (0 = no blinking)
DATA7/8:	the bits set to "1" of the data bytes control the outports the outports whose bits are "0" will remain in their current states DATA7: bit0 controls outport0, .. , bit7 controls outport7 DATA8: bit0 controls outport8, .. , bit7 controls outport15
DATA9:	function ("0" / "1") or startup value ("0" / "1") in blink mode

Example 1: Outport 0 is be set to "1" for 256ms. The following combination of the data bytes is required:

duration:	DATA1 = 0x00, DATA2 = 0x01
high time:	DATA3 = 0x00, DATA4 = 0x00
low time:	DATA5 = 0x00, DATA6 = 0x00
outports:	DATA7 = 0x01, DATA8 = 0x00
function:	DATA9 = 0x01

Example 2: Outports 2 and 3 are to blink for 514 ms, high time 50 ms, low time 60 ms, beginning with "1". The following combination will be required:

duration:	DATA1 = 0x02, DATA2 = 0x02
high time:	DATA3 = 0x32, DATA4 = 0x00
low time:	DATA5 = 0x3C, DATA6 = 0x00
outports:	DATA7 = 0x0C, DATA8 = 0x00
function:	DATA9 = 0x01

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.6.5 Get Atmel Sensor Temperature (0x55)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x55	0x00	0x01

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x55	0x01	0x00	Temp.

This packet causes the module to send the current temperature of the Atmel FingerChip sensor. The returned value is valid only if Atmel sensor is configured (see commands 0x00, 0x08 and 0x09).

DATA1:sensor temperature (18 … 50 °C)

2.6.6 Get State (0x56)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x56	0x00	0x01

Packet from module:

LENGTH	CMD	SENDER	DEST	DATA1
0x0001	0x56	0x01	0x00	state

This packet causes the module to transmit the current state.

DATA1:	0 = module is ready
	1 = identification is running

2.6.7 Write Bytes to Flash Memory (0x57)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	…	DATA*n*
n	0x57	0x00	0x01	offset (low)	offset (high)	byte #0	…	byte #(*n*–3)

This packet causes the module to store data into flash memory (64 KBytes for common use).
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK (all bytes written successfully), ERROR

2.6.8 Read Bytes from Flash Memory (0x58)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0004	0x58	0x00	0x01	offset (low)	offset (high)	number (low)	number (high)

At first the module replies the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR (wrong offset/number)
If the request is OK, the module sends a packet subsequently containing data read from flash memory.
Condiftion: This command is only available if cmd 0x EA (Encrypted Authtication) is deaktivated, otherwise it would be possible to read out the secret marriage key

2.6.9 Erase Flash Memory (0x59)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x59	0x00	0x01

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.6.10 Calibrate Atmel-Sensor Temperature Management (0x5B BioKey®-3000 or newer)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
0x0004	0x5B	0x00	0x01	cool time (low)	cool time (high)	ambient temperature	temperature offset

This packet causes the module to calibrate the Atmel sensor temperature management as there is a big variation in the temperature at delivery state - calibration will result in a more precise management.
Before the calibration can start, the sensor has to cool down to ambient temperature. DATA1 and DATA2 specify the cool down period in seconds. 300 seconds will be a good value to ensure that the sensor cooled down. While cooling down, the module will not respond to any other packet.
DATA3 is the ambient temperature in degrees Celsius. The calibration method works best at an ambient temperature of about 23°C. The more the actual ambient temperature differs from 23°C the less precise the calibration result will be.
The calibration method heats up the sensor to about 43°C (note that this value is an uncalibrated one). DATA4 specifies an offset to use a greater heat up temperature. Normally, you should use 0 as a default value.

After execution the module sends the standardized answer (see 2.1.1).
Status possible: OK, ERROR (bad sensor)

2.6.11 Get Software Information (0x5C)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x5C	0x00	0x01

After this packet the module will send some informations about the running firmware.

LENGTH	CMD	SENDER	DEST	DATA1	…	DATA4	DATA5	…	DATA29
0x1D	0x5C	0x01	0x00	crc[0..7]	…	crc[24..31]	InfoStr1	…	InfoStr25

DATA1…DATA4	32 Bit CRC
DATA5…DATA29	25 ASCII character info string

2.6.12 Control Outports Simultaneously (0x5D)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3	DATA4
variable	0x5D	0x00	0x01	Portnr and ctrlbits	Duration Bits 0..7	Duration Bits 8..15	Blinkon

DATA5	DATA6	…	DATA10	…
Blinkoff	Portnr and ctrlbits	…	Blinkoff	…

This packet is a more versatile version of message cmd 0x53 (Control Outports).

With this packet the terminal can control all of the output ports simultaneously and still set completely different behaviour for each port.
The packet has a variable length, depending on how many ports the user wants to set, the length must be a multiple of 5. The DATA Bytes are divided in blocks of 5 bytes with each block setting the parameters for one output port. The meaning of the 5 bytes is as follows:

BYTE1 (DATA1,6,11 …):	Bits 0..3:	Port Number to control
	Bit 6:	Blink mode (set to 1 to enable blink mode for this port)
	Bit 7:	Port value, if blink mode is enabled this will determine the starting state of the port, otherwise it will determine the ports new state

BYTE2 (DATA2,7,12 …) and
BYTE3 (DATA3,8,13 …) : Overall Activation duration of the port in ms, if set to 0 the port will keep the new state indefinitely or until it is set to a new state with this message or msg 0x53.
Note:
When the port is set to blink mode it will keep the value it is currently at when this time runs out and will not be reset to 0.

BYTE4 (DATA4,9,14 …): High Phase in blink mode in 50ms.
This controls the time the port is set to 1 while blinking.

BYTE5 (DATA5,10,15 …): Low Phase in blink mode in 50ms.
This controls the time the port is set to 0 while blinking.

After execution the module sends the standardized answer (see 2.1.1).
Status possible: OK, ERROR

2.6.13 Get Software Composition (0x60)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x60	0x00	0x01

After this packet the module will send detailed information about the running firmware.

LENGTH	CMD	SENDER	DEST	DATA1	…	DATA N
N (variable)	0x60	0x01	0x00	Code1	…	CodeN

DATA1…DATA N these fields represent an overview about all components of the running firmware

2.6.14 Get Version Number (0x62)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x62	0x00	0x01

Status packet from module:

LENGTH	CMD	SENDER	DEST	DATA1	DATA2	DATA3
0x0003	0x62	0x01	0x00	Version0	Version1	Version2

This packet causes the module to send a status packet containing the following 3 data bytes:

DATA1:	software version 0 - Core version (encoding and matching, i.e. the fingerprint engine)
DATA2:	software version 1 - Protocol version (the implemented protocol version)
DATA3:	software version 2 - Revision number (used to identify small changes between firmware with the same core version and protocol version)

2.6.15 Secure communication authentication (0x80)

Packet from terminal:

LENGTH	CMD	SENDER	DEST	DATA1
Variable	0x80	0x00	0x01	1. char

DATA2	DATAn
2. char	…

This message is sent to the module in order to unlock the secured message cmds when the modules secure communication mode is enabled. If Authentication is successful, these message cmds can then be accessed for the timeperiod configured using the message cmd 0x14 (set secure communication configuration)

DATA2-DATAn: Password characters (max. 32 allowed), **no** zero termination

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible:

OK	Authentication successful, secured message cmds can be now be accessed for the configured timeperiod
AUTH_FAILED	The password was incorrect.

2.6.16 Secure communication close bus (0x81)

Packet from terminal:

LENGTH	CMD	SENDER	DEST
0x0000	0x81	0x00	0x01

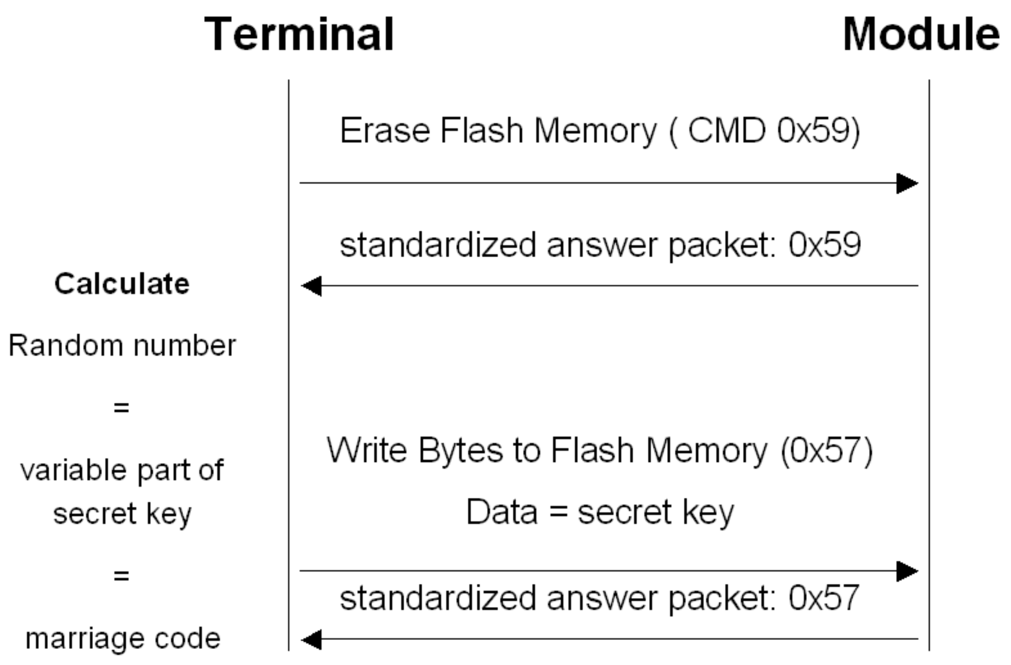
This message is sent to the module in order to explicitly lock the access to secured message cmds again after it was unlocked using message cmd 0x80.
After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

2.7 Protocol encryption

To comply with modern security standards it is possible to encrypt the protocol communication in order to provide confidentiality, integrity and authenticity of the connection.

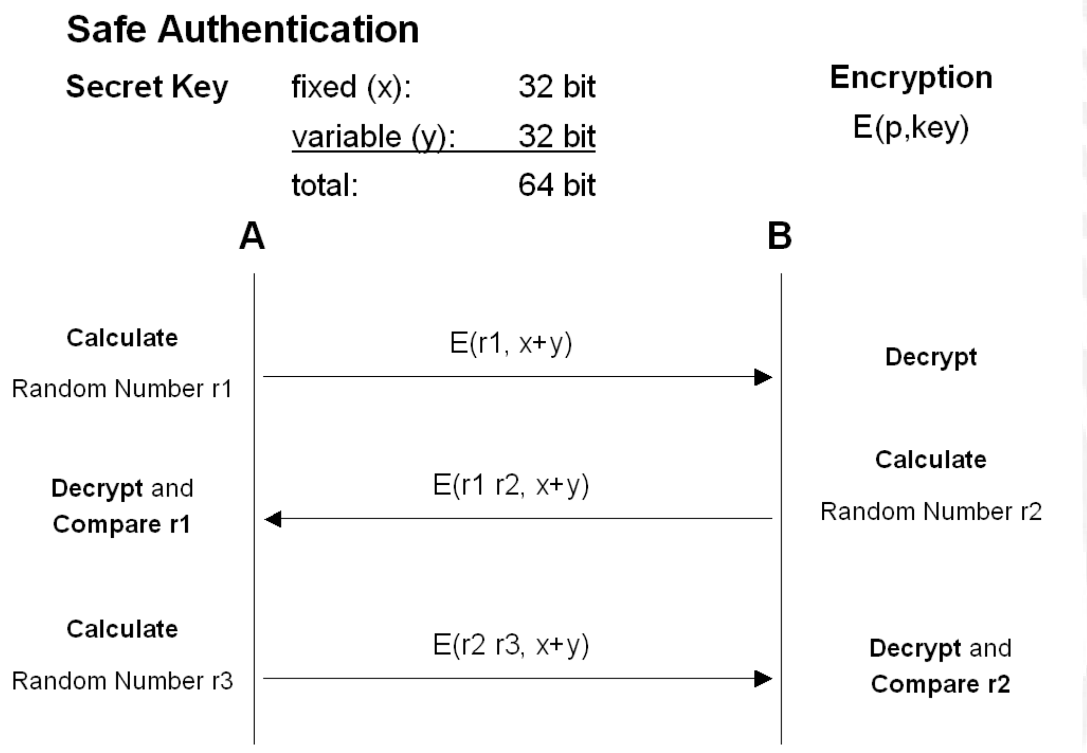
Initialization of the secure channel (secret key)

As a symmetric cipher is used, the secret key for both encryption and decryption is the same. The key has a fixed length of 64 bit (8 byte) and consists of fixed and variable parts. The fixed parts are setup once when delivering the device whereas the variable parts are computed during this initialization process.
If two devices want to start a communication they need to have the same variable part of the key that are exchanged for example in the process of marriage between an internal and outside unit. This initialization process is shown in the following figure. The terminal initiates the connection (here: Erase Flash 0x59) and sends an unencrypted random number to the module. The module sends an answer packet with ID_OK .


Terminal
Module
Erase Flash Memory (CMD 0x59)
standardized answer packet: 0x59
Calculate
Random number
=
variable part of secret key
=
marriage code
Write Bytes to Flash Memory (0x57)
Data = secret key
standardized answer packet: 0x57

Safe Authentication (0xEA)

The following figure shows the authentication process. Node B authenticates to node A. Afterwards node A authenticate to node B.



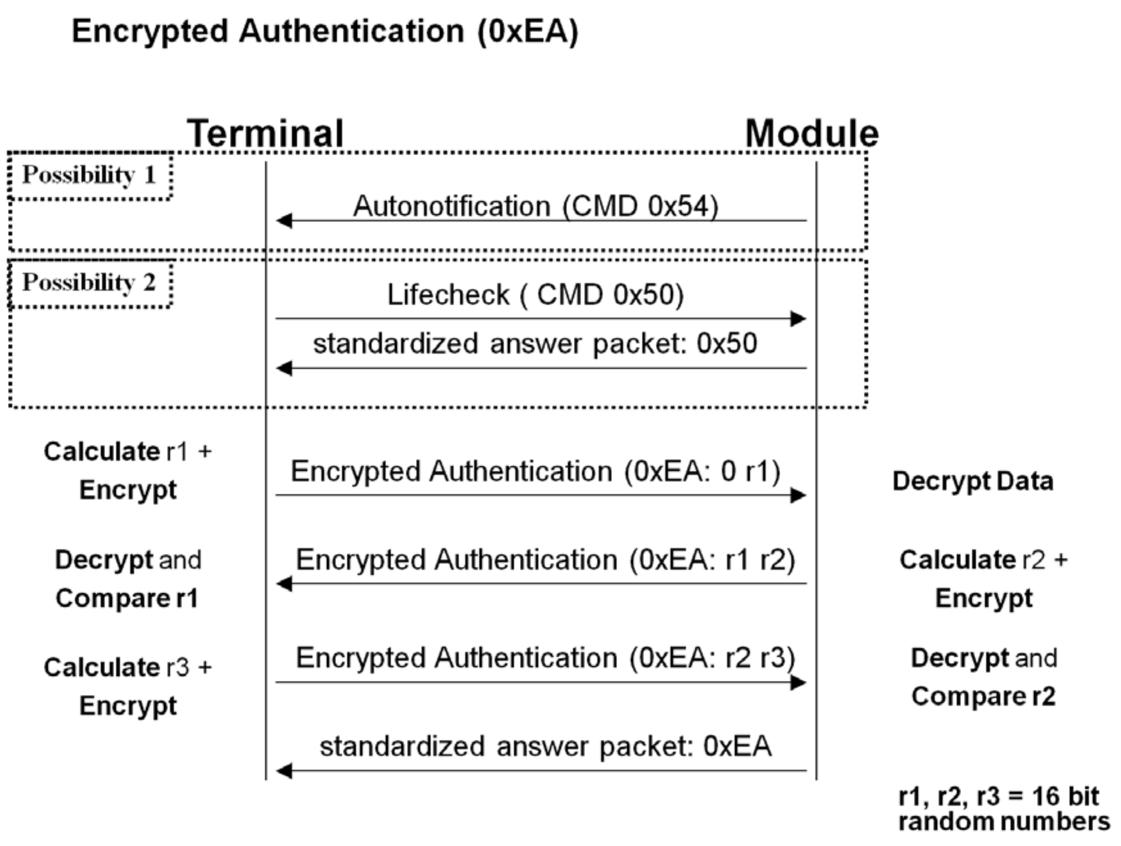
Basically A (internal unit or bus master) starts the process with the first encrypted packet that contains a random number r1. B (outside unit / fingerprint reader or bus slave) decrypts the packet, reads r1 and sends and encrypted answer packet with r1 and an additional number r2. A receives the packet, decrypts it and recognizes that the number r1 is in the packet. So A can assumed that B has the same secret key. In a third step A sends r2 and a third random number r3 back. B decrypts the paket and recognizes that r2 is inside. The authentication is successful.

2.7.1 Encrypted Authentication (0xEA)

To initiate a safe authentication with the Biokey Modules, there are 2 ways:

- after a Lifecheck (0x50) or
- after a Module’s automatic information packet after identification (0x54)

The following figure shows the process:


Encrypted Authentication (0xEA)
Terminal
Module
Possibility 1
Autonotification (CMD 0x54)
Possibility 2
Lifecheck (CMD 0x50)
standardized answer packet: 0x50
Calculate r1 + Encrypt
Encrypted Authentication (0xEA: 0 r1)
Decrypt Data
Decrypt and Compare r1
Encrypted Authentication (0xEA: r1 r2)
Calculate r2 + Encrypt
Calculate r3 + Encrypt
Encrypted Authentication (0xEA: r2 r3)
Decrypt and Compare r2
standardized answer packet: 0xEA
r1, r2, r3 = 16 bit random numbers

Packet from terminal or module:

LENGTH	CMD	SENDER	DEST	DATA1*	DATA2*	DATA3*	DATA4*
0x0004	0xEA	0x00 or 0x01	0x01 or 0x00	Random Number 1 (High)	Random Number 1 (Low)	Random Number 2 (High)	Random Number 2 (Low)

* Data encrypeted with KeeLoq

This packet is designed to accomplish a safe authentication like described above. Therefore the whole data fields are encrypted with the KeeLoq algorithm. The secret key to encrypt the data is 64 bit long and is composed of a 32 bit fixed number (set by Idencom Germany GmbH) and a 32 variable number (set with Cmd 0x57, normally referred as “marriage key”).

In the first transmission Data1 and Data2 of the 0xEA Command contain 0. Data 3 and 4 carry the 16 bit random number r1. At the way back this number r1 is carried in Data 1 and 2. Data 3 and 4 contain the second random number r2 of the module. Now the terminal (e.g. the relay) can decide if the authentication is successful: If the decrypted data still contains r1, it means, that the variable part of the secret key is the same on both modules. So the authentication is successful.
In order that the module knows the terminal is an authenticated partner, the terminal sends r2 back to the module (in Data 1 and 2).

DATA1: encrypted random number 1 (Most significant bits)
DATA2: encrypted random number 1 (Least significant bits)

DATA3: encrypted random number 2 (Most significant bits)
DATA4: encrypted random number 2 (Least significant bits)

After execution the module sends the standardized answer packet (see 2.1.1).
Status possible: OK, ERROR

3. Glossary

3.1 Definition of terms

PID

Personal Identifier, unique 64 bits number, associated to one person.
The range of a PID is 1 to 2^{64} - 1.

FID

Finger Identifier, 8 bits number, associated to one finger of a person.
The range of a FID is 1 to 10.

AID

Affiliation Identifier, 8 bits number, associated to one enrollment of a finger of a person.
The range of a AID is 1 to 255.

FPT

FingerPrintTemplate, template storing extracted fingerprint features and additional information.

Minutiae

Feature in a fingerprint.

3.2 Template structure

A fingerprint template contains administrative information as well as up to 50 minutiae which consist of several data themselves. The structure of the administrative information is equal with all minutiae formats. The maximum template size is 566 bytes in IDENCOM format, 166 bytes in DIN V66400 format and 266 bytes in IDENCOM Compact format.
Note that you can adjust the maximum number of minutiae to a value less than 50 by using the Message Commands 0x09 (Transmit Configuration to Terminal) or 0x0B (Setting Algorithm Parameters). This will lower the maximum template size while sacrificing some template quality.

For further information please refer to the document
Biokey_FingerprintTemplate_Definition.pdf

3.3 Default values for module configuration

This chapter gives information about the module's configuration parameters. The default values stated below are the delivery settings. These can be restored with the command "2.2.8 Resetting module to delivery settings".
Information about how to change these values is given in chapter 2.2.

Sensor	sensor used to acquire fingerprint images *default:* Atmel Fingerchip™ Sensor (0) *range of values:* Atmel Fingerchip™ Sensor (0), Infineon FingerTIP™ Sensor (1)
Sensor temperature control	if activated, the Atmel sensor temperature will be controlled *default:* activated
Sensor timeout	time the sensor waits for a finger after an according command specified in 1/10 seconds *default:* 50 (5 seconds)
Device	device used to communicate with a terminal *default:* serial (RS232, CMOS levels)
Baud rate	data signaling rate *default:* 115200 baud supported are 9600, 19200, 38400, 57600 and 115200 baud
Image size	size of the image read from sensor and processed *default:* width x height: 300 x 400 pixels *maximum values:* 500 x 500 Pixels
Security level	threshold used to classify a matching score as recognized *default:* 40 *range of values:* adjustable between 0 and 99
Minimum image quality	threshold, needed to extract features from a fingerprint image *default:* 40 *range of values*: adjustable between 0 and 99
Minimum minutiae	minimum number of minutiae for a template to be valid *default:* 18 *range of values*: adjustable between 0 and 99
Module address	the module only responds to packets with this receiver address *default:* 1 *range of values*: adjustable between 1 and 239
Continuous Read	if activated, the sensor is read continuously - if a finger is detected, it is encoded and matched *default:* deactivated *range of values:* activated, deactivated
Use LEDs	if activated, the LEDs are used for user guidance *default:* activated *range of values:* activated, deactivated
Use debug functions	if activated, additional debug functions are used - these functions are not described within this protocol, they can be

	used by Idencom for service *default:* deactivated *range of values:* activated, deactivated
Automatic notification	if activated, an automatic notification packet is sent after successful identification (see 2.1.2) *default:* deactivated *range of values:* activated, deactivated
Notification address	address to which the automatic notification packet is sent *default:* 0x00 *range of values:* 0x00 .. 0xEF
Use access signal	if activated, the access signal is set after successful idenfication *default:* activated *range of values:* activated, deactivated
Access signal duration	determines how long the access signal is set (in ms) *default:* 1000 ms *range of values:* 0 .. 65535
Template format	determines which template format is used for communication (see 3.2) *default:* 0 = Idencom *range of values:* 0 (Idencom), 1 (DIN V66400), 2 (Idencom Compact)