TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY

Embedded Systems - IAY0330
Fingerprint Sensor Datasheet
GT-511C1R

Students :

Maxime AUBINEAU

Adeline LEGER

Thibaut MAZERY

Adrien MERLIER

Preface

This datasheet has been written in 2015 by the team which realized the fingerprint sensor project at Tallinn University of Technology. This datasheet describes how to use the fingerprint sensor GT-511C1R and has been written because the original one is not complete and some parts are wrong. We added explanations, chronograms and examples to be more understandable. This document is not protected and can be shared, used and modified as well. Here are links to the original datasheet, given by the constructor :

Manufacturer website : `http://www.adh-tech.com.tw`
Manufacturer e-mail : `sales@adh-tech.com.tw`
Official datasheet : `http://www.adh-tech.com.tw/files/GT-511C1R_datasheet_V1%205_20140312.pdf`

Table of contents

1 Wiring schematic **3**

2 Technical characteristics **4**

3 Packet structure **5**
3.1 General structure . . . 5
3.1.1 Command packet . . . 5
3.1.2 Response packet . . . 5
3.1.3 Data packet . . . 6
3.2 Calculations . . . 6

4 Commands summary **7**

5 Errors summary **8**

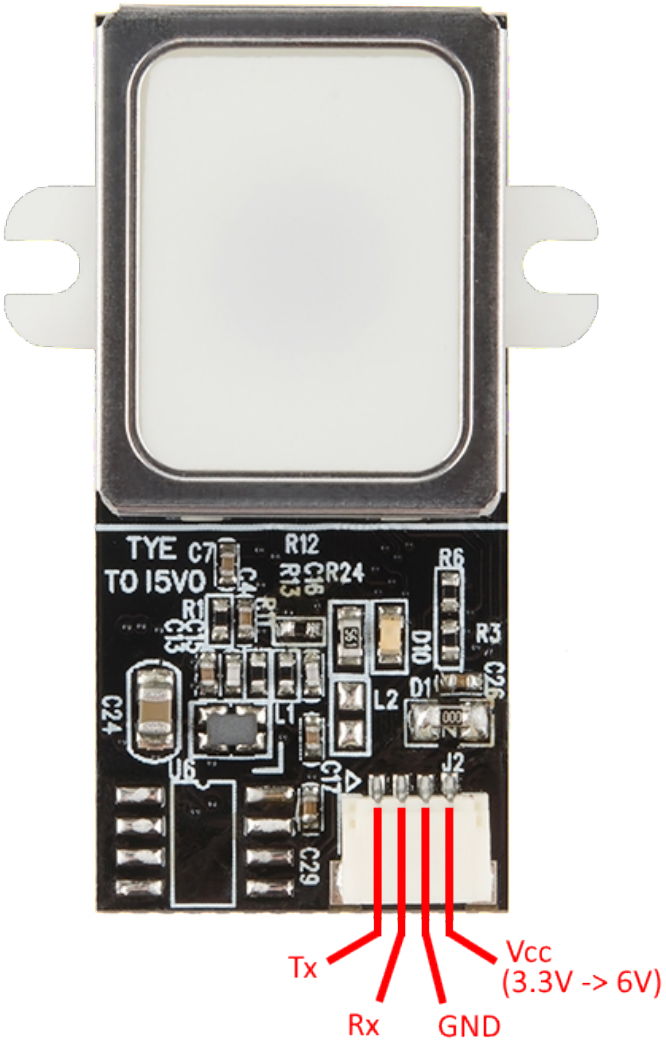
6 Chronogram of reception and emission **9**

7 Protocol : initialization, flowcharts examples **10**
7.1 Description . . . 10
7.2 Enrollment flowchart . . . 10
7.3 Identifying flowchart . . . 10

8 Functions list **11**
8.1 Open . . . 11
8.2 ChangeBaudrate . . . 12
8.3 CmosLed . . . 13
8.4 GetEnrollCount . . . 14
8.5 CheckEnrolled . . . 15
8.6 EnrollStart . . . 16
8.7 Enroll1 . . . 17
8.8 Enroll2 . . . 18
8.9 Enroll3 . . . 19
8.10 IsPressFinger . . . 20
8.11 DeleteID . . . 21
8.12 DeleteAll . . . 22
8.13 Verify . . . 23
8.14 Identify . . . 24
8.15 VerifyTemplate . . . 25
8.16 IdentifyTemplate . . . 27
8.17 CaptureFinger . . . 29
8.18 MakeTemplate . . . 30
8.19 GetImage . . . 31
8.20 GetRawImage . . . 32
8.21 GetTemplate . . . 33
8.22 SetTemplate . . . 34

1 Wiring schematic

The wiring schematic of the fingerprint sensor is the following :



2 Technical characteristics

This device in one chip module with :
– fingerprint algorithm
– optical sensor

The major functions of the sensor are the following :
– ultra-thin optical sensor
– 1 :1 verification, 1 :n identification
– downloading fingerprint image from the device
– reading & writing fingerprint template(s) from/to the device
– simple UART & USB communication protocol

Technical specifications :

CPU	ARM Cortex M3 Core
Sensor	Optical sensor
Effective area of the sensor	14 x 12.5(mm)
Image size	240 x 216 pixels
Resolution	450 dpi
Maximum number of fingerprints	20
Matching mode	1 :1, 1 :N
Template size	504 bytes (template) + 2 bytes (checksum)
Communication interface	UART, 9600 bps (by default)
False acceptance rate (FAR)	$< 0.001\%$
False rejection rate (FRR)	$< 0.1\%$
Enrollment time	< 3 sec (3 fingerprints)
Identification time	1.5 sec (20 fingerprints)
Operating voltage	DC 3.3 6V
Operating current	< 130 mA
Operating environment : temperature	-20°C to +60°C
Operating environment : humidity	20% to 80%
Storage environment : temperature	-20°C to +60°C
Storage environment : humidity	10% to 80%

3 Packet structure

IMPORTANT : Before doing any operation, you should **switch on the CMOS LED of the sensor**, without it, it is not able to work. All the packets need to be sent with the **Little Endian representation** (the least significant bit (LSB) need to be sent first, and you have to finish by the most significant bit(MSB)). In the following packets, you will send the structure of each packets from the buffer 0 to the buffer 11 (except some particular structures, depending on the data sent or received).

3.1 General structure

Each packet is a structure of 12 bytes (characterized by an unsigned char variable for each byte) excepted the data packet which have a particular structure :

BUFFER INDEX	DESCRIPTION
0	Byte for command start code 1
1	Byte for command start code 2
2 to 3	Array of 2 bytes for the device ID
4 to 7	Array of 4 bytes for the parameter
8 to 9	Array of 2 bytes for the command code or the response code
10 to 11	Array of 2 bytes for the checksum

3.1.1 Command packet

BUFFER INDEX	ITEM	DESCRIPTION
0	Command start code 1	Buffer[0] = 0x55 (fixed)
1	Command start code 2	Buffer[1] = 0xAA (fixed)
2 to 3	Device ID 0x01	Buffer[2] = 0x01 (fixed) Buffer[3] = 0x00 (fixed)
4 to 7	Parameter	Given in each function (see chapter 8) If not defined in a function, write 0x00 on each index
8 to 9	Command	Buffer[8] = code given on chapter 4 Buffer[9] = 0x00
10 to 11	Checksum	Calculation explained on chapter 3.2

3.1.2 Response packet

BUFFER INDEX	ITEM	DESCRIPTION
0	Response start code 1	Buffer[0] = 0x55 (fixed)
1	Response start code 2	Buffer[1] = 0xAA (fixed)
2 to 3	Device ID 0x01	Buffer[2] = 0x01 (fixed) Buffer[3] = 0x00 (fixed)
4 to 7	Parameter	If acknowledge (0x30), Parameter = Output Parameter If non-acknowledge (0x31), Parameter = Error code Error code is given in chapter 5
8 to 9	Response	Buffer[8] = 0x30 (acknowledge) Buffer[8] = 0x31 (non-acknowledge) Buffer[9] = 0x00
10 to 11	Checksum	Calculation explained on chapter 3.2

3.1.3 Data packet

A data packet for a template represents an amount of 506 bytes and a data packet for an image represents an amount of 51840 bytes. Then in the data structure below, N represents the number of data we want to send :

BUFFER INDEX	ITEM	DESCRIPTION
0	Data start code 1	Buffer[0] = 0x5A (fixed)
1	Data start code 2	Buffer[1] = 0xA5 (fixed)
2 to 3	Device ID 0x01	Buffer[2] = 0x01 (fixed) Buffer[3] = 0x00 (fixed)
4 to 4+N-1	N bytes of data	Data defined your template/image size Buffer[4] = Data Buffer[5] = Data ... Buffer[4+N-2] = Data Buffer[4+N-1] = Data
4+N to 4+N+1	Checksum	Calculation explained on chapter 3.2

3.2 Calculations

To calculate the checksum, you have to sum every byte you send. Then you write this value in Little Endian as shown in the example of command packet below :

$$\text{Checksum} = \text{BUFFER\_INDEX}[0] + \text{BUFFER\_INDEX}[1] + ... + \text{BUFFER\_INDEX}[9]$$

Example : You want to send a message to switch on the CMOS LED, the message will look like :

BUFFER INDEX	NUMBER
0	0x55
1	+ 0xAA
2	+ 0x01
3	+ 0x00
4	+ 0x01
5	+ 0x00
6	+ 0x00
7	+ 0x00
8	+ 0x12
9	+ 0x00
Checksum	= 0x0113
10	0x13
11	0x01

4 Commands summary

As explained on chapter 3, you need to send a command parameter in the command packet, each parameter you can send is referenced in the table below, please, pay attention, the number is written in **hexadecimal format**.

Original datasheet contains unuseful, obsolete or not supported commands which are not listed here :

NUMBER (HEX)	ALIAS	DESCRIPTION
01	Open	Initialization of the fingerprint sensor
03	UsbInternalCheck	Check if the connected USB device is valid
04	ChangeBaudrate	Change UART baud rate (9600 bps by default)
12	CmosLed	Control Cmos LED
20	GetEnrolledCount	Get the number of enrolled fingerprint
21	CheckEnrolled	Check whether the specified ID is already enrolled
22	EnrollStart	Start an enrollment
23	Enroll1	Make 1st template for an enrollment
24	Enroll2	Make 2nd template for an enrollment
25	Enroll3	Make 3rd template for an enrollment, merge three templates into one template, save merged template to the database
26	IsPressFinger	Check if a finger is placed on the sensor
40	DeleteID	Delete the fingerprint with the specified ID
41	DeleteAll	Delete all fingerprints from the database
50	Verify	1 :1 Verification of the capture fingerprint image with the specified ID
51	Identify	1 :N Identification of the capture fingerprint image with the database
52	VerifyTemplate	1 :1 Verification of a fingerprint template with the specified ID
53	IdentifyTemplate	1 :N Identification of a fingerprint template with the database
60	CaptureFinger	Capture a fingerprint image (256x256) from the sensor
61	MakeTemplate	Make template for transmission
62	GetImage	Download the captured fingerprint image (256x256)
63	GetRawImage	Download raw fingerprint image (320x240)
70	GetTemplate	Download the template of the specified ID
71	SetTemplate	Upload the template of the specified ID

It exists two additional commands used to aknowledge or not the message received :

NUMBER (HEX)	ALIAS	DESCRIPTION
30	Ack	Acknowledge
31	Nack	Non-acknowledge

5 Errors summary

In case of reception of non-acknowledge parameter, the response parameter gives an error code that can give you information for the reason of the problem, all the errors code are listed below, the value of the non-acknowledge parameter is given in **hexadecimal format**.
Original datasheet contains obsolete error codes which are not listed here :

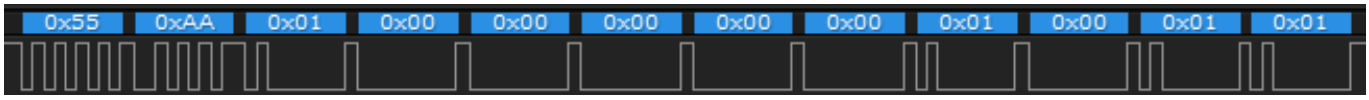
NACK PARAMETER	VALUE	DESCRIPTION
NACK_INVALID_POS	0x1003	The specified ID is not between 0 and 19
NACK_IS_NOT_USED	0x1004	The specified ID is not used
NACK_IS_ALREADY_USED	0x1005	The specified ID is already used
NACK_COMM_ERR	0x1006	Communication error
NACK_VERIFY_FAILED	0x1007	1 :1 Verification failure
NACK_IDENTIFY_FAILED	0x1008	1 :N Identification failure
NACK_DB_IS_FULL	0x1009	The database is full
NACK_DB_IS_EMPTY	0x100A	The database is empty
NACK_BAD_FINGER	0x100C	Too bad fingerprint
NACK_ENROLL_FAILED	0x100D	Enrollment failure
NACK_IS_NOT_SUPPORTED	0x100E	The specified command is not supported
NACK_DEV_ERR	0x100F	Device Error, especially if Crypto-Chip is trouble
NACK_INVALID_PARAM	0x1011	Invalid parameter
NACK_FINGER_IS_NOT_PRESSED	0x1012	Finger is not pressed

6 Chronogram of reception and emission

The UART in not a high speed communication, take care to do not send and receive a message in the same time, this is not possible and you will have conflicts.

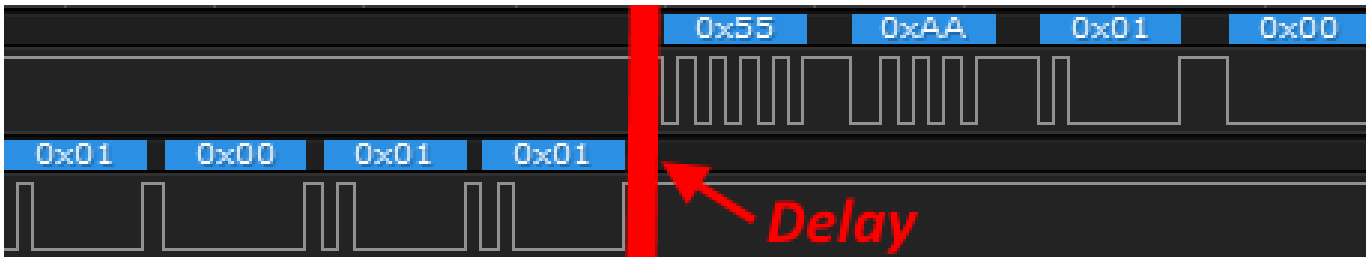
To illustrate this paragraph with an example, here is some chronograms of the initialization function.

This is the sent command :



This is the received command :

This is the representation of the delay between the emission and the reception :



The full representation of the chronogram is given on the wiki :
`http://ati.ttu.ee/embsys/images/b/b9/Fps_init_command.png`

7 Protocol : initialization, flowcharts examples

7.1 Description

An initialization protocol is needed for the fingerprint sensor. Before any flowchart you need to call the Open function that will initialize properly the fingerprint sensor. **Then, switch on the CMOS LED. Otherwise the fingerprint detection will not work.** After this initialization, you can make any flowchart you want, here are some examples to help you.

7.2 Enrollment flowchart

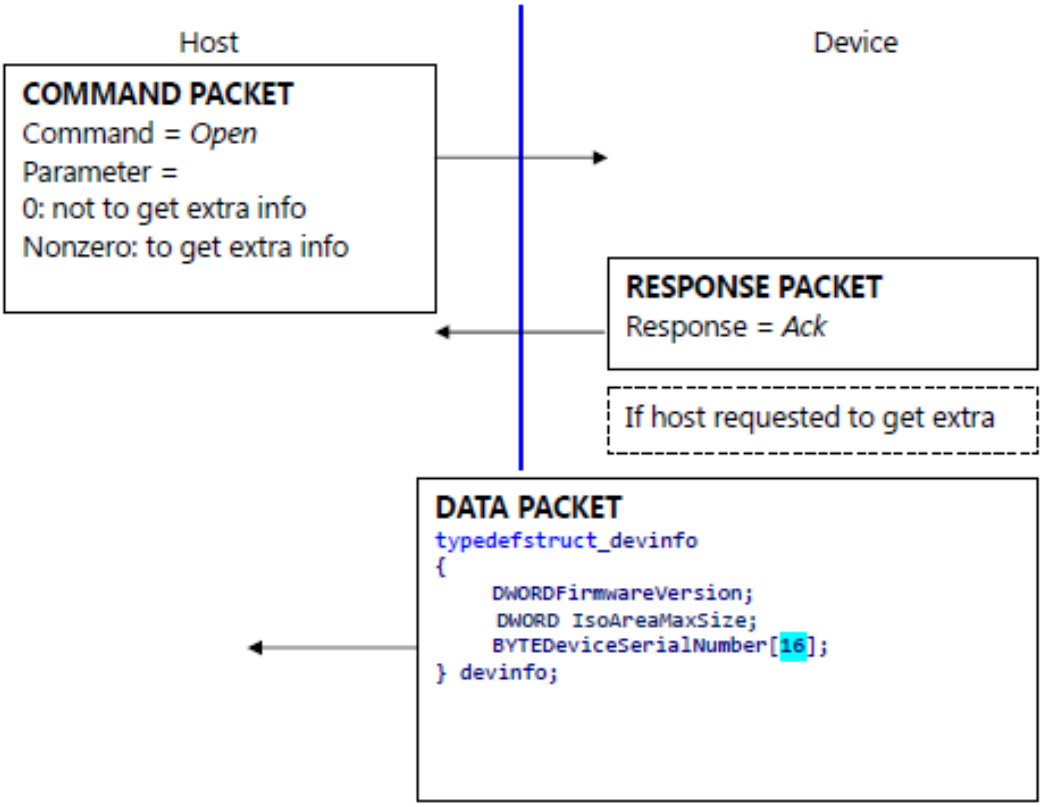
1. EnrollStart with a (not used) ID
2. CaptureFinger
3. Enroll1
4. Wait to take off the finger using IsPressFinger
5. CaptureFinger
6. Enroll2
7. Wait to take off the finger using IsPressFinger
8. CaptureFinger
9. Enroll3

7.3 Identifying flowchart

1. Wait the presence of a finger using IsPressFinger
2. Identify
3. Wait to take off the finger using IsPressFinger

8 Functions list

8.1 Open

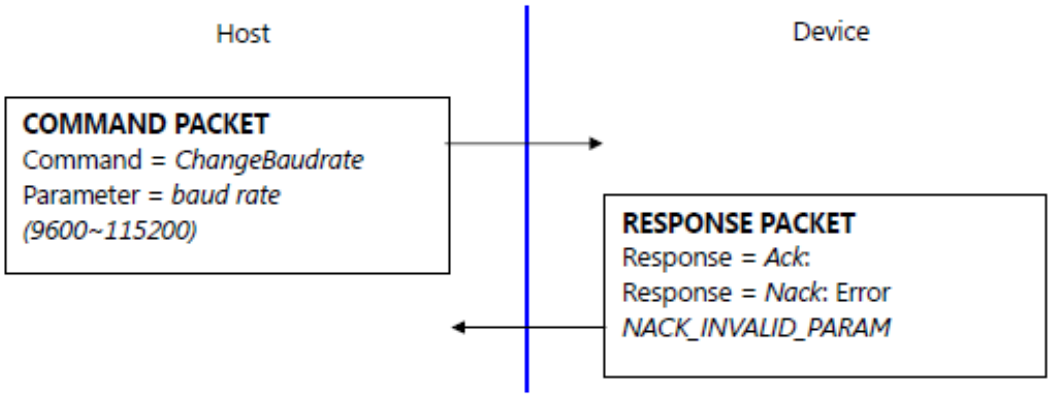


Example : You want to initialize the device without extra information

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x01	0x30
9	0x00	0x00
10	0x01	0x30
11	0x01	0x01

Open command is used to initialize the device ; especially it gets device's static info.

8.2 ChangeBaudrate

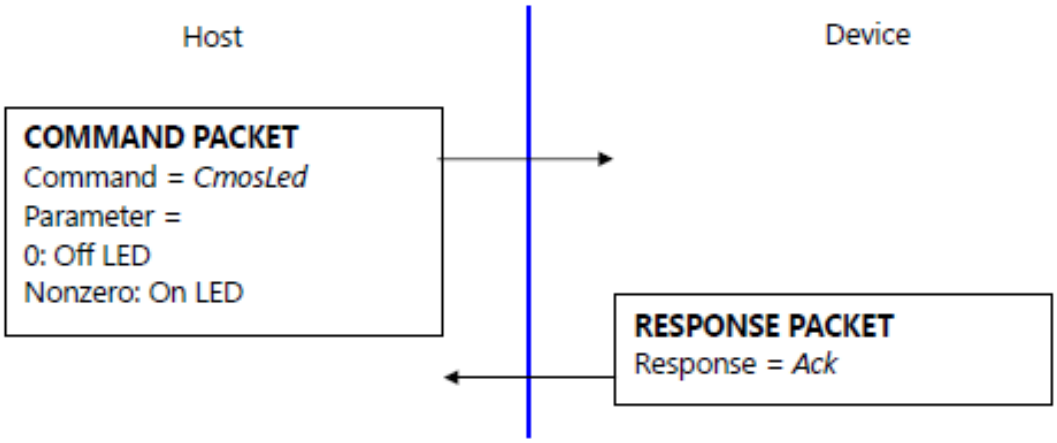


Example : You want to change the baud rate to 115200 bps (in hexadecimal format 115200 = 0x01C200)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0xC2	0x00
6	0x01	0x00
7	0x00	0x00
8	0x04	0x30
9	0x00	0x00
10	0xC7	0x30
11	0x01	0x01

This command changes the UART baud rate at the run-time.
The device initializes its UART baud rate to 9600 bps after power on.

8.3 CmosLed

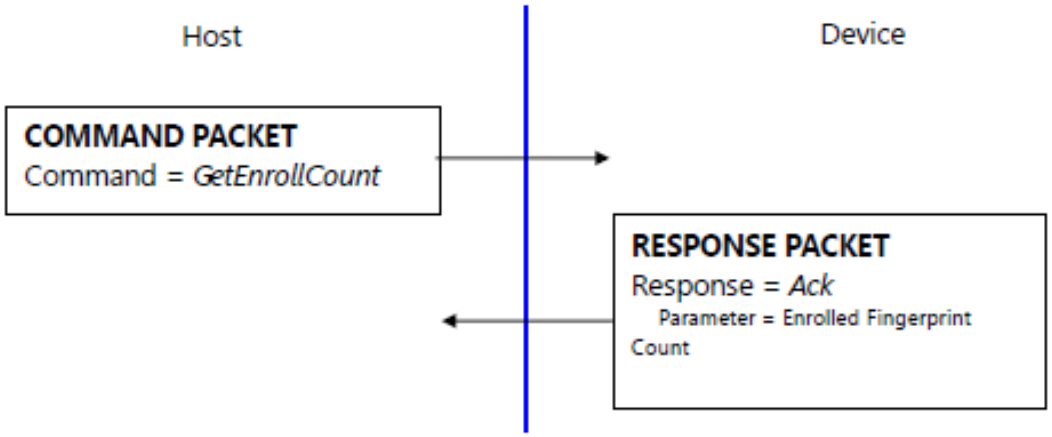


Example : You want to turn on the CMOS LED

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x01	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x12	0x30
9	0x00	0x00
10	0x13	0x30
11	0x01	0x01

Default state of CMOS (Sensor) LED is OFF state.
(But while booting, LED blinks once, this says the LED is OK.)
Therefore, please issue LED ON command prior to any capture.

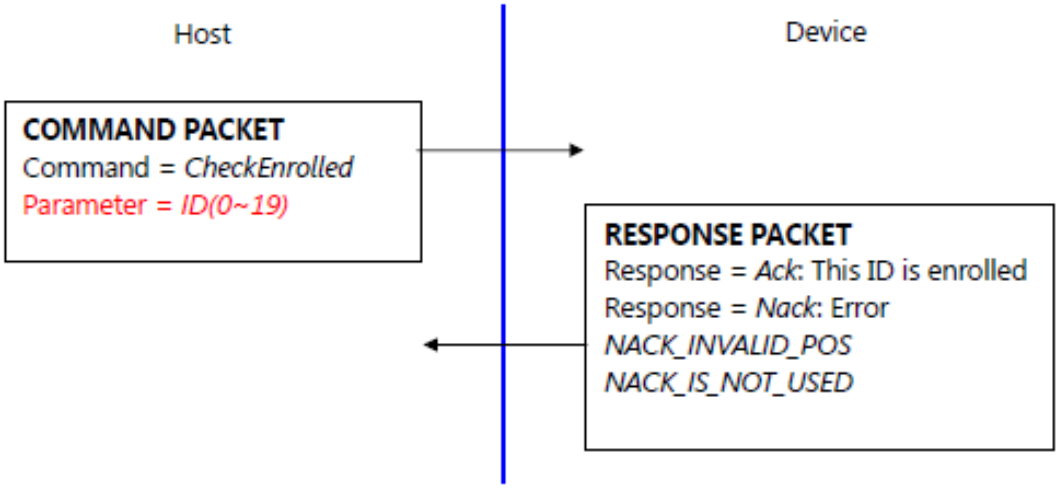
8.4 GetEnrollCount



Example : You want to know your fingerprint ID (for the example, I suppose it is in ID 13, then 0x0D in hexadecimal format)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x0D
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x20	0x30
9	0x00	0x00
10	0x20	0x3D
11	0x01	0x01

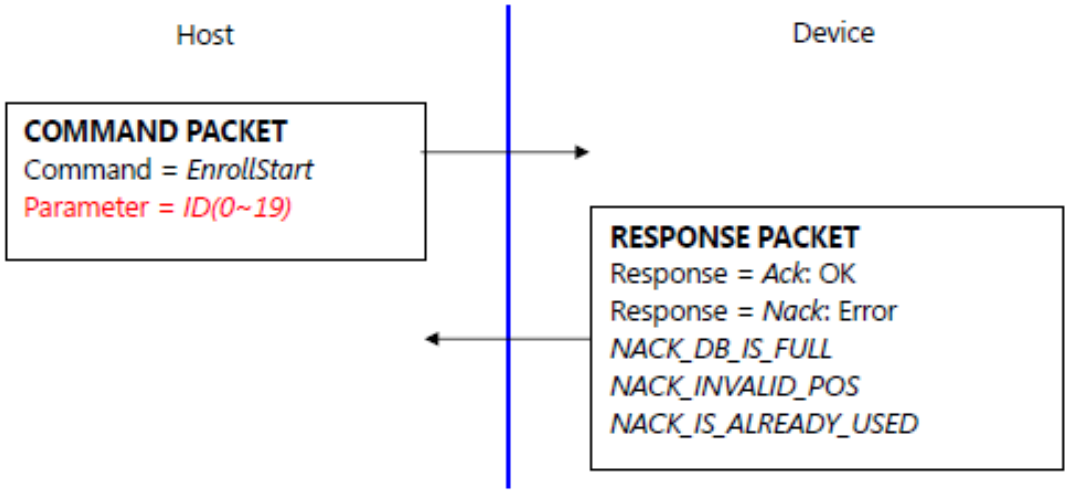
8.5 CheckEnrolled



Example : You want to check if the fingerprint is already enrolled in the fingerprint sensor at the ID 17 (I suppose it is enrolled at this position, 17 = 0x0101 in hexadecimal format)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x01	0x00
5	0x01	0x00
6	0x00	0x00
7	0x00	0x00
8	0x21	0x30
9	0x00	0x00
10	0x23	0x30
11	0x01	0x01

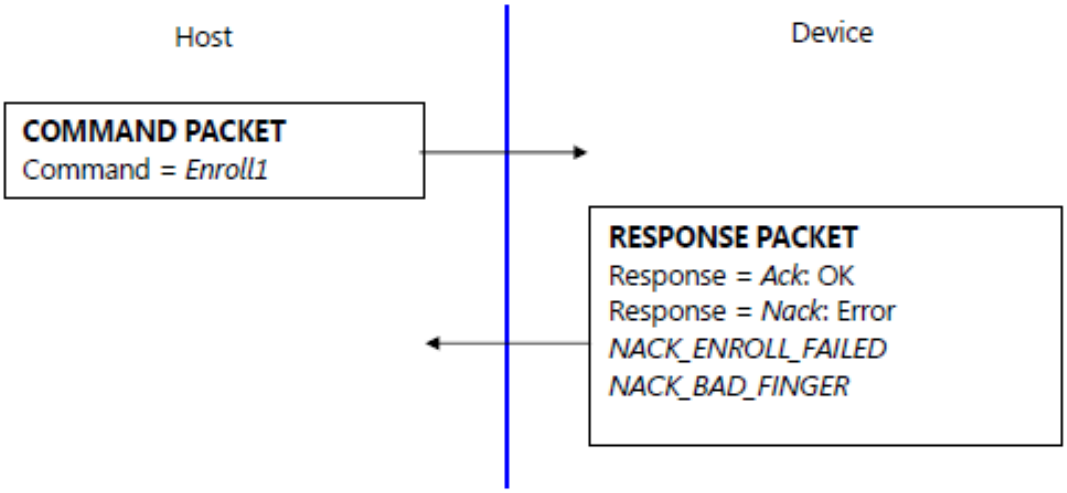
8.6 EnrollStart



Example : You want to enroll your fingerprint at the ID 19 (I suppose that the ID is available, 19 = 0x0103 in hexadecimal format)
Note : This example will be taken also for the functions Enroll1, Enroll2 and Enroll3

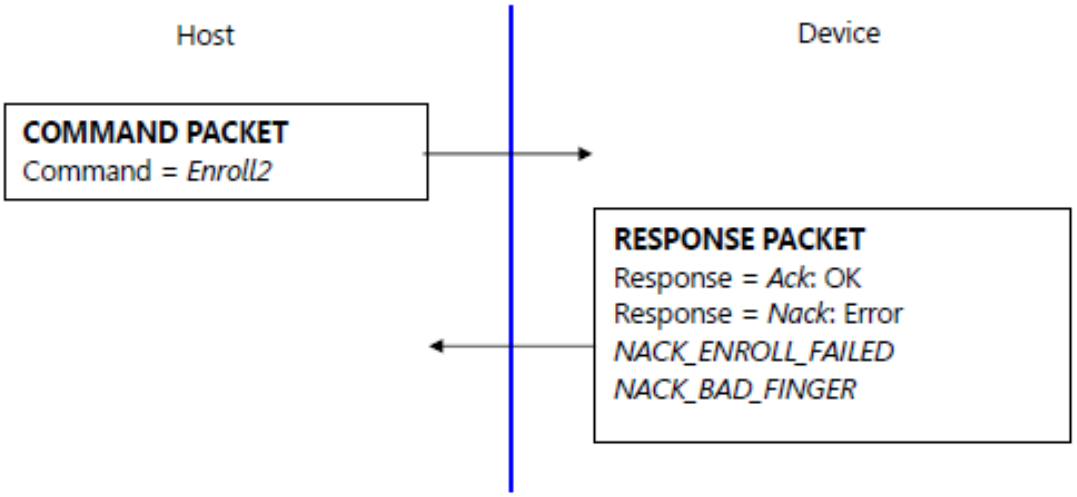
BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x03	0x00
5	0x01	0x00
6	0x00	0x00
7	0x00	0x00
8	0x22	0x30
9	0x00	0x00
10	0x26	0x30
11	0x01	0x01

8.7 Enroll1



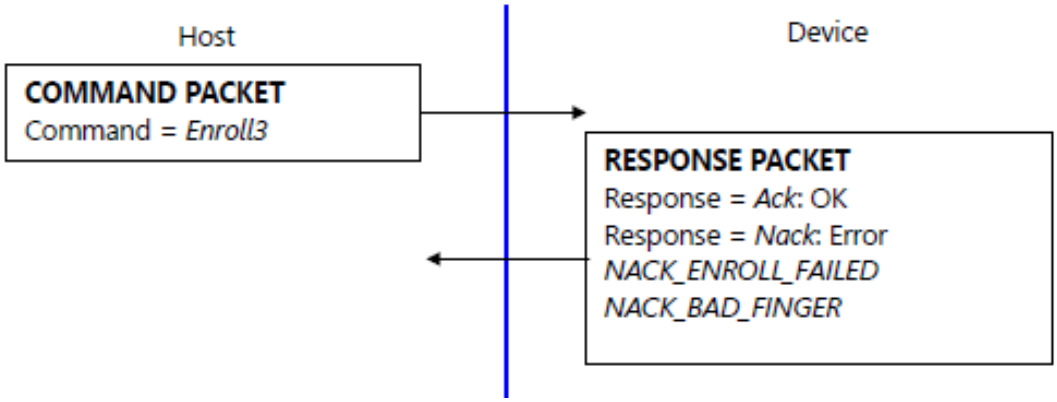
BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x23	0x30
9	0x00	0x00
10	0x23	0x30
11	0x01	0x01

8.8 Enroll2



BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x24	0x30
9	0x00	0x00
10	0x24	0x30
11	0x01	0x01

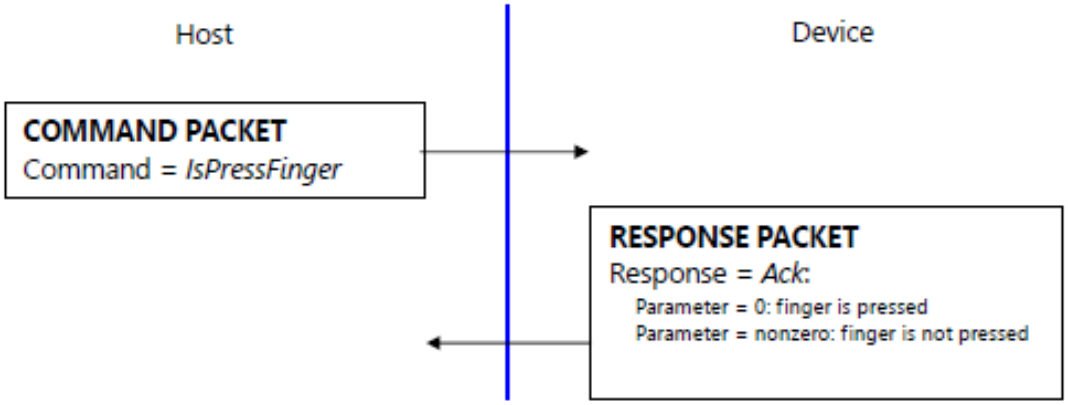
8.9 Enroll3



BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x25	0x30
9	0x00	0x00
10	0x25	0x30
11	0x01	0x01

To enroll a fingerprint, the host must issue above 4 commands, the chapter 7 describes how to organize these commands.

8.10 IsPressFinger

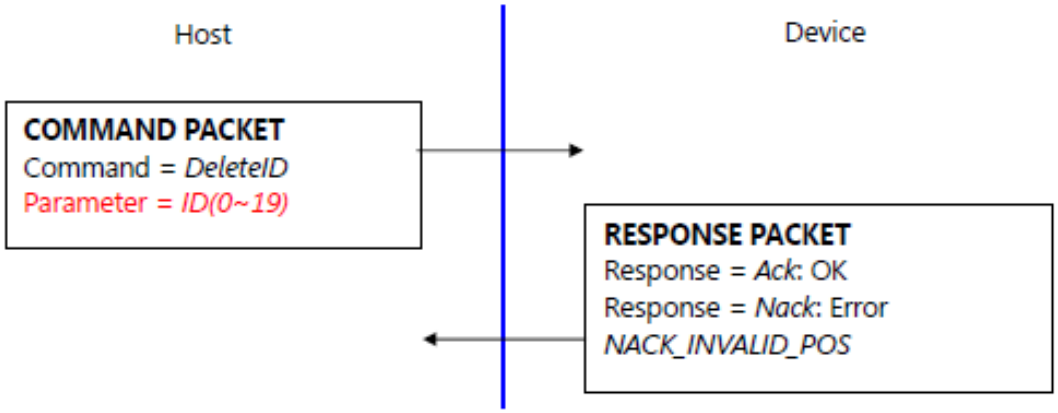


Example : You want to check if a finger is pressed of the sensor

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x26	0x30
9	0x00	0x00
10	0x26	0x30
11	0x01	0x01

This command is used while enrollment, the host waits to take off the finger per enrollment stage.

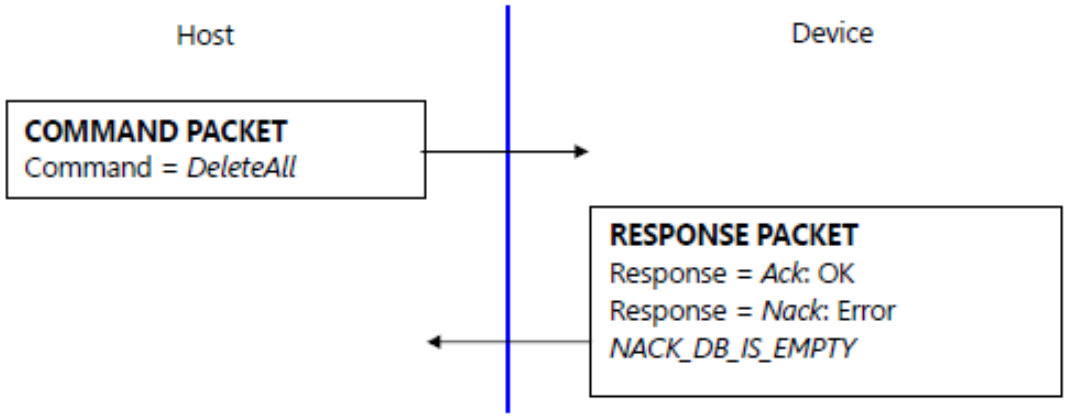
8.11 DeleteID



Example : You want to delete the enrolled fingerprint at the ID 7 (I suppose that a fingerprint is enrolled at this position)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x07	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x40	0x30
9	0x00	0x00
10	0x47	0x30
11	0x01	0x01

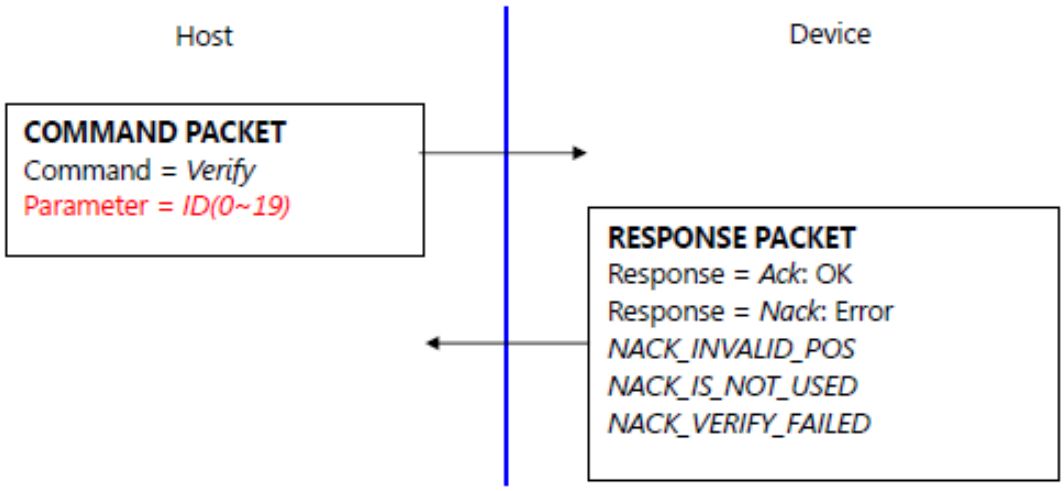
8.12 DeleteAll



Example : You want to delete all the fingerprints in the sensor database

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x41	0x30
9	0x00	0x00
10	0x41	0x30
11	0x01	0x01

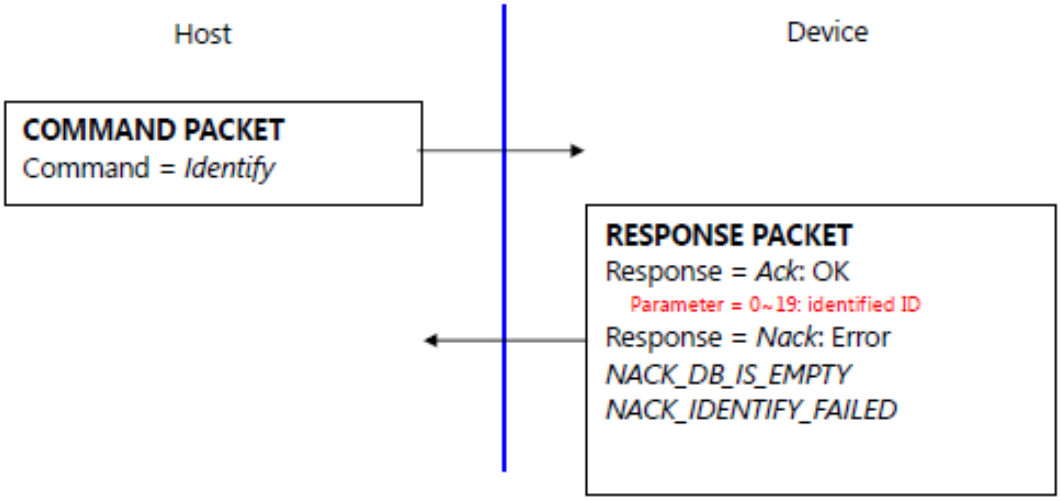
8.13 Verify



Example : You want to know if a friend's fingerprint exists at the ID 10 (for the example, I suppose that the friend is identified and have the ID 10)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x0A	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x50	0x30
9	0x00	0x00
10	0x5A	0x30
11	0x01	0x01

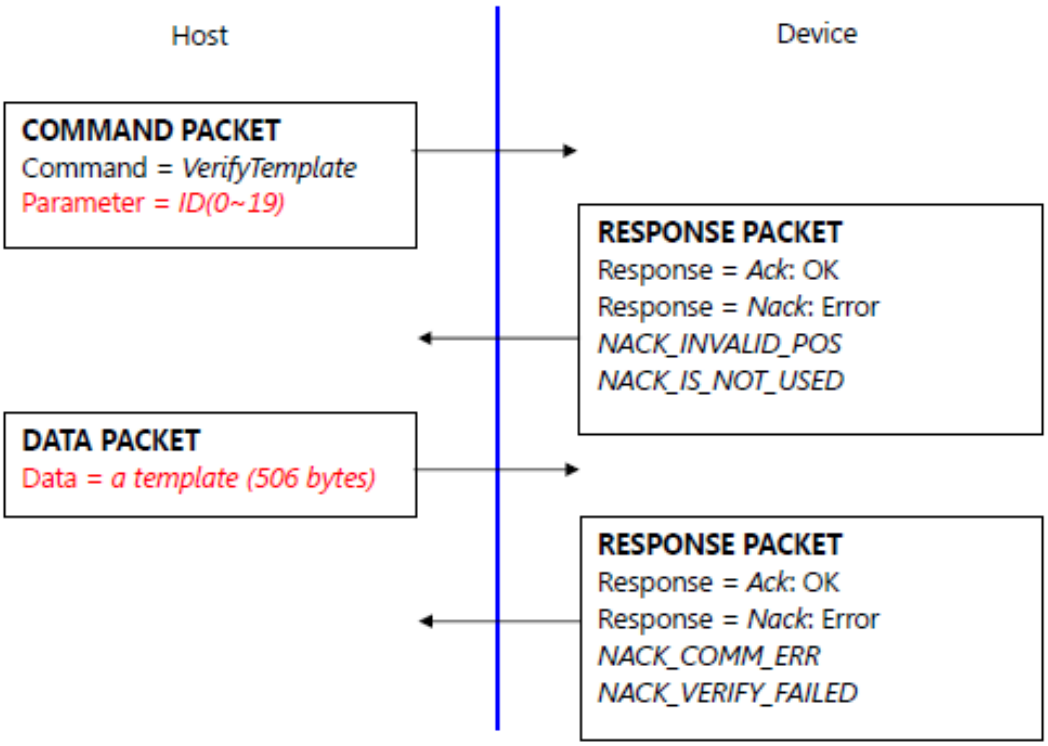
8.14 Identify



Example : You want to know if a friend is identified in the database (for the example, I suppose that the friend is identified and have the ID 14 = 0xE in hexadecimal format)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x0E
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x51	0x30
9	0x00	0x00
10	0x51	0x3E
11	0x01	0x01

8.15 VerifyTemplate



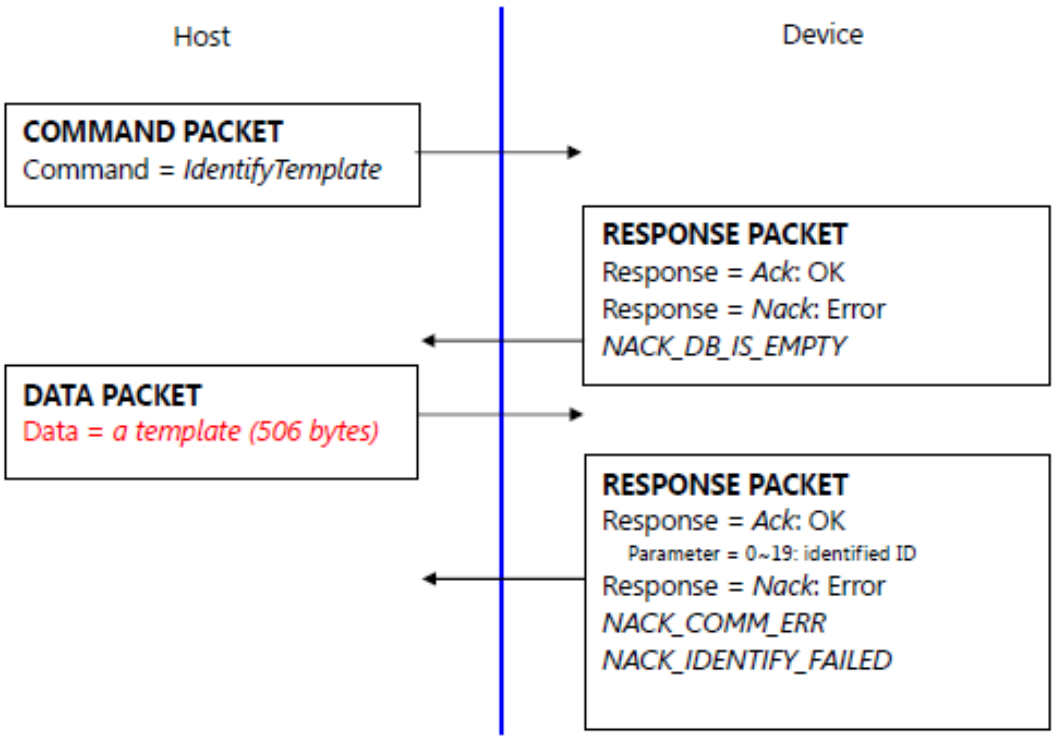
Example : You want to verify that the template is registered at the ID 5 (for the example, I suppose that a fingerprint is registered at the ID 5 but it is not the fingerprint of your template)

BUFFER INDEX	SENT COMMAND	RESPONSE (COMMAND)	REPONSE (DATA)
0	0x55	0x55	0x5A
1	0xAA	0xAA	0xA5
2	0x01	0x01	0x01
3	0x00	0x00	0x00
4	0x05	0x00	0x07
5	0x00	0x00	0x10
6	0x00	0x00	0x00
7	0x00	0x00	0x00
8	0x52	0x30	0x31
9	0x00	0x00	0x00
10	0x57	0x30	0x48
11	0x01	0x01	0x01

The data packet is written like this, the template has a size of 506 bytes :

BUFFER INDEX	SENT DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum

8.16 IdentifyTemplate



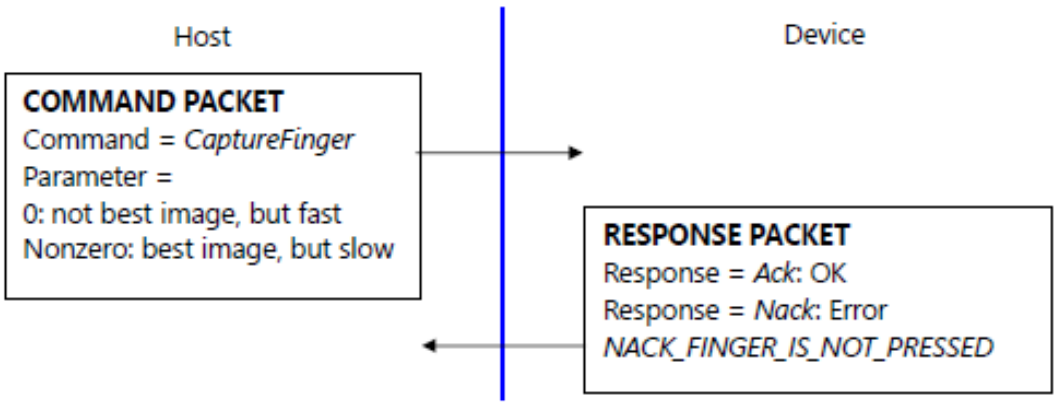
Example : You want to verify that the template is registered on the database (for the example, I suppose that the template is registered at the ID 6)

BUFFER INDEX	SENT COMMAND	RESPONSE (COMMAND)	REPONSE (DATA)
0	0x55	0x55	0x5A
1	0xAA	0xAA	0xA5
2	0x01	0x01	0x01
3	0x00	0x00	0x00
4	0x05	0x00	0x06
5	0x00	0x00	0x00
6	0x00	0x00	0x00
7	0x00	0x00	0x00
8	0x52	0x30	0x30
9	0x00	0x00	0x00
10	0x57	0x30	0x36
11	0x01	0x01	0x01

The data packet is written like this, the template has a size of 506 bytes :

BUFFER INDEX	SENT DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum

8.17 CaptureFinger

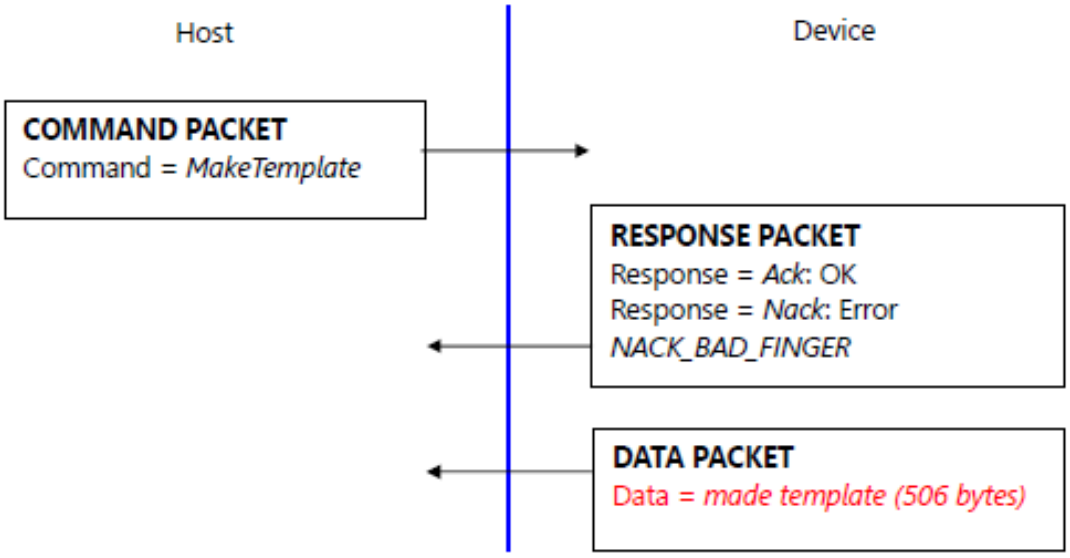


Example : You want to capture a fingerprint in high quality but with low speed

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x01	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x60	0x30
9	0x00	0x00
10	0x61	0x30
11	0x01	0x01

The fingerprint algorithm uses 240x216 image for its input.
This command captures raw image from the sensor and converts it to 240x216 image for the fingerprint algorithm. If the finger is not pressed, this command returns with non-acknowledge.
Please use best image for enrollment to get best enrollment data.
Please use not best image for identification (verification) to get fast user sensibility.

8.18 MakeTemplate



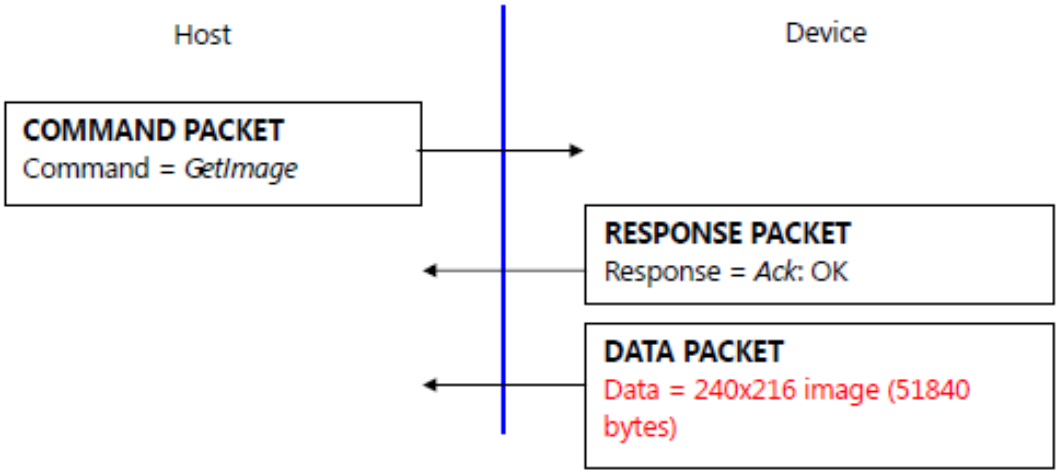
BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x61	0x30
9	0x00	0x00
10	0x61	0x30
11	0x01	0x01

After that, you will receive the template in a data structure of 506 bytes like below :

BUFFER INDEX	DESIRED DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum

This function makes template for transmission. CaptureFinger command should be previously issued. Do not use the template for registration.

8.19 GetImage

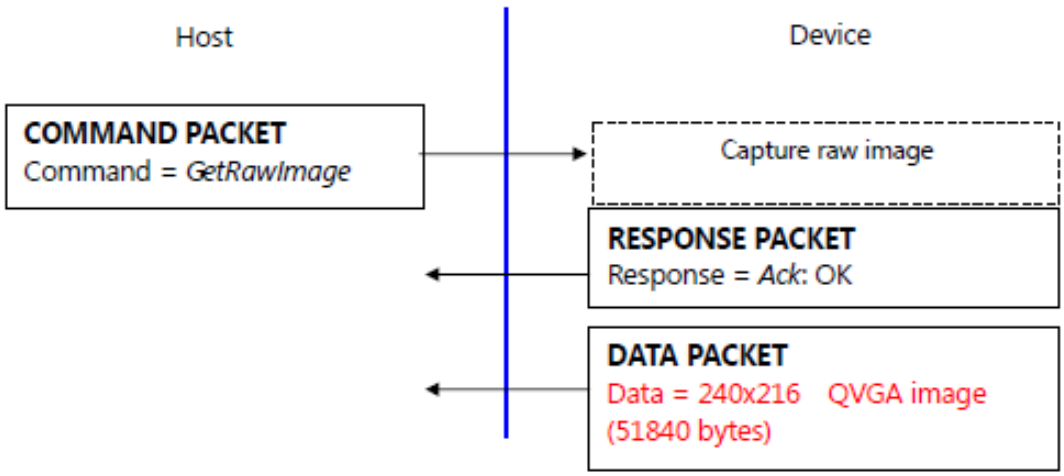


BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x62	0x30
9	0x00	0x00
10	0x62	0x30
11	0x01	0x01

After that, you will receive the fingerprint image in a data structure of 51840 bytes like below :

BUFFER INDEX	DESIRED DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum

8.20 GetRawImage

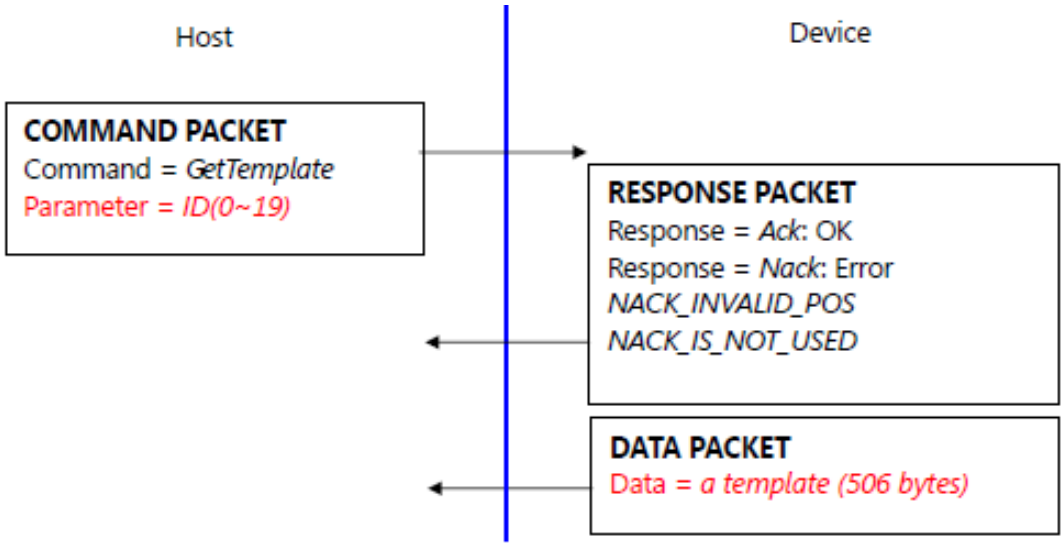


BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x00	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x63	0x30
9	0x00	0x00
10	0x63	0x30
11	0x01	0x01

After that, you will receive the fingerprint QVGA image in a data structure of 51840 bytes like below :

BUFFER INDEX	DESIRED DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum

8.21 GetTemplate



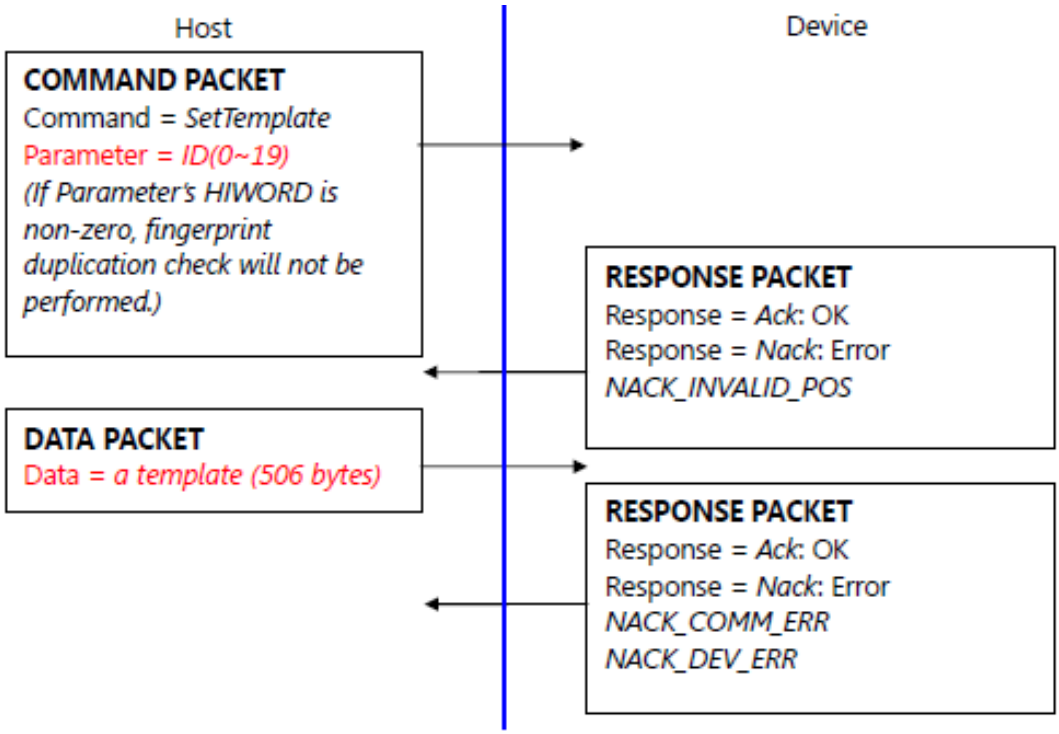
Example : You want to get the template at the ID 19 (19 = 0x13 in hexadecimal format)

BUFFER INDEX	SENT COMMAND	DESIRED RESPONSE
0	0x55	0x55
1	0xAA	0xAA
2	0x01	0x01
3	0x00	0x00
4	0x13	0x00
5	0x00	0x00
6	0x00	0x00
7	0x00	0x00
8	0x70	0x30
9	0x00	0x00
10	0x83	0x30
11	0x01	0x01

After that, you will receive the template in a data structure of 506 bytes like below :

BUFFER INDEX	DESIRED DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum

8.22 SetTemplate



Example : You want to set a template at the ID 7 (for the example, I suppose that the ID is available and that I do not want to perform a duplication check)

BUFFER INDEX	SENT COMMAND	RESPONSE (COMMAND)	REPONSE (DATA)
0	0x55	0x55	0x5A
1	0xAA	0xAA	0xA5
2	0x01	0x01	0x01
3	0x00	0x00	0x00
4	0x07	0x00	0x00
5	0x00	0x00	0x00
6	0x00	0x00	0x00
7	0x01	0x00	0x00
8	0x71	0x30	0x30
9	0x00	0x00	0x00
10	0x79	0x30	0x30
11	0x01	0x01	0x01

The data packet is written like this, the template has a size of 506 bytes :

BUFFER INDEX	SENT DATA
0	0x5A
1	0xA5
2	0x01
3	0x00
4	Something
...	...
4+N-1	Something
4+N	Checksum
4+N+1	Checksum