Programming Guide
GT-521F52
Optical Fingerprint Recognition EMBEDDED Module
Version 1.0
Mar 17, 2017
Table of Contents

Revision History ................................................................................................................................................. 4

1. General Description ........................................................................................................................................... 5
   1.1. About this document ................................................................................................................................. 5
   1.2. Related documents ..................................................................................................................................... 5
   1.3. Comment ...................................................................................................................................................... 5

2. Protocol: Packet Structure ............................................................................................................................... 6
   Command Packet (Command) ............................................................................................................................. 6
   Response Packet (Acknowledge) .......................................................................................................................... 6
   Data Packet (Data) ............................................................................................................................................ 7

3. Protocol: Commands Summary .......................................................................................................................... 8

4. Protocol: Error Codes ....................................................................................................................................... 10

5. Protocol: Command Details ............................................................................................................................. 12
   5.1. Initialization(Open) .................................................................................................................................. 12
   5.2. Termination(Close) ................................................................................................................................... 14
   5.3. Fast searching of the device(UsbInternalCheck) ....................................................................................... 14
   5.4. CMOS LED control(CmosLed) .................................................................................................................... 15
   5.5. Changing UART baud rate (ChangeBaudrate) ............................................................................................ 16
   5.6. Get enrolled fingerprint count(GetEnrollCount) ......................................................................................... 17
   5.7. Check enrollment status(CheckEnrolled) .................................................................................................... 17
   5.8. Start an enrollment(EnrollStart) .................................................................................................................. 18
   5.9. Make 1st template for an enrollment(Enroll1) ............................................................................................ 18
   5.10. Make 2nd template for an enrollment(Enroll2) .......................................................................................... 19
   5.11. Make 3rd template for an enrollment, merge three templates(Enroll3) ...................................................... 19
   5.12. Check finger pressing status(IsPressFinger) ............................................................................................. 21
   5.13. Delete one fingerprint(DeleteID) ............................................................................................................... 22
   5.14. Delete all fingerprints(DeleteAll) ............................................................................................................ 22
   5.15. 1:1 Verification(Verify) ............................................................................................................................ 23
   5.16. 1:N Identification(Identify) ..................................................................................................................... 23
   5.17. 1:1 Verification of Template(VerifyTemplate) .............................................................................................. 24
   5.18. 1:N Identification of Template(IdentifyTemplate) ....................................................................................... 25
   5.19. Capture fingerprint(CaptureFinger) ............................................................................................................ 26
   5.20. Make Template(MakeTemplate) .................................................................................................................. 27
   5.21. Get fingerprint image(GetImage) ................................................................................................................ 28
   5.22. Get raw image(GetRawImage) ..................................................................................................................... 29
   5.23. Get template(GetTemplate) ....................................................................................................................... 29
   5.24. Set template(SetTemplate) ......................................................................................................................... 30
   5.25. Start database download, obsolete(GetDatabaseStart) .............................................................................. 30
5.26.  End database download, obsolete(GetDatabaseEnd) .......... 31
5.27.  Set SecurityLevel ......................................................... 32
5.28.  GetSecurityLevel ........................................................... 32
5.29.  EnterStandbyMode ......................................................... 33
5.30.  Identify_Template2 ....................................................... 34

   6.1.    Capture of the fingerprint image ................................. 35
   6.2.    Identifying and Verifying ......................................... 35
   6.3.    Enrollment ............................................................. 35

7.  PC Demo ........................................................................... 36
## Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1.0</td>
<td>Mar 17, 2017</td>
<td>Created</td>
</tr>
</tbody>
</table>
1. **General Description**

1.1. *About this document*

   The document is created to facilitate the firmware porting for the embedded system.

1.2. *Related documents*

   You can find relative datasheets, include:
   - **GT-521F52** datasheet
   - **GT-521F52** programming guide
   - **GT-521F52** SDK

1.3. *Comment*

   To report errors or make suggestions, please send to email to:

   **sales@adh-tech.com.tw**
# 2. Protocol: Packet Structure

(Multi-byte item is represented as Little Endian.)

**Command Packet (Command)**

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>ITEM</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x55</td>
<td>BYTE</td>
<td>Command start code1</td>
</tr>
<tr>
<td>1</td>
<td>0xAA</td>
<td>BYTE</td>
<td>Command start code2</td>
</tr>
<tr>
<td>2</td>
<td>Device ID</td>
<td>WORD</td>
<td>Device ID: default is 0x0001, always fixed</td>
</tr>
<tr>
<td>4</td>
<td>Parameter</td>
<td>DWORD</td>
<td>Input parameter</td>
</tr>
<tr>
<td>8</td>
<td>Command</td>
<td>WORD</td>
<td>Command code</td>
</tr>
<tr>
<td>10</td>
<td>Check Sum</td>
<td>WORD</td>
<td>Check Sum (byte addition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OFFSET[0]...+OFFSET[9]=Check Sum</td>
</tr>
</tbody>
</table>

**Response Packet (Acknowledge)**

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>ITEM</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x55</td>
<td>BYTE</td>
<td>Response start code1</td>
</tr>
<tr>
<td>1</td>
<td>0xAA</td>
<td>BYTE</td>
<td>Response start code2</td>
</tr>
<tr>
<td>2</td>
<td>Device ID</td>
<td>WORD</td>
<td>Device ID: default is 0x0001, always fixed</td>
</tr>
<tr>
<td>4</td>
<td>Parameter</td>
<td>DWORD</td>
<td><strong>Response == 0x30</strong>: (ACK) Output Parameter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Response == 0x31</strong>: (NACK) Error code</td>
</tr>
<tr>
<td>8</td>
<td>Response</td>
<td>WORD</td>
<td>0x30: Acknowledge (ACK), 0x31: Non-acknowledge (NACK).</td>
</tr>
<tr>
<td>10</td>
<td>Check Sum</td>
<td>WORD</td>
<td>Check Sum (byte addition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OFFSET[0]...+OFFSET[9]=Check Sum</td>
</tr>
</tbody>
</table>
### Data Packet (Data)

<table>
<thead>
<tr>
<th>OFFSET</th>
<th>ITEM</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x5A</td>
<td>BYTE</td>
<td>Data start code 1</td>
</tr>
<tr>
<td>1</td>
<td>0xA5</td>
<td>BYTE</td>
<td>Data start code 2</td>
</tr>
<tr>
<td>2</td>
<td>Device ID</td>
<td>WORD</td>
<td>Device ID: default is 0x0001, always fixed</td>
</tr>
<tr>
<td>4</td>
<td>Data</td>
<td>N BYTES</td>
<td>N bytes Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The size is pre-defined per protocol stage</td>
</tr>
<tr>
<td>4+N</td>
<td>Check Sum</td>
<td>WORD</td>
<td>Check Sum (byte addition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>OFFSET[0]+…+OFFSET[4+N-1]=Check Sum</td>
</tr>
</tbody>
</table>
### 3. Protocol: Commands Summary

In a command packet *Command* can be one of below.

<table>
<thead>
<tr>
<th>Number (HEX)</th>
<th>Alias</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Open</td>
<td>Initialization</td>
</tr>
<tr>
<td>02</td>
<td>Close</td>
<td>Termination</td>
</tr>
<tr>
<td>03</td>
<td>UsbInternalCheck</td>
<td>Check if the connected USB device is valid</td>
</tr>
<tr>
<td>04</td>
<td>ChangeBaudrate</td>
<td>Change UART baud rate</td>
</tr>
<tr>
<td>12</td>
<td>CmosLed</td>
<td>Control CMOS LED</td>
</tr>
<tr>
<td>20</td>
<td>GetEnrollCount</td>
<td>Get enrolled fingerprint count</td>
</tr>
<tr>
<td>21</td>
<td>CheckEnrolled</td>
<td>Check whether the specified ID is already enrolled</td>
</tr>
<tr>
<td>22</td>
<td>EnrollStart</td>
<td>Start an enrollment</td>
</tr>
<tr>
<td>23</td>
<td>Enroll1</td>
<td>Make 1\textsuperscript{st} template for an enrollment</td>
</tr>
<tr>
<td>24</td>
<td>Enroll2</td>
<td>Make 2\textsuperscript{nd} template for an enrollment</td>
</tr>
<tr>
<td>25</td>
<td>Enroll3</td>
<td>Make 3\textsuperscript{rd} template for an enrollment, merge three templates into one template, save merged template to the database</td>
</tr>
<tr>
<td>26</td>
<td>IsPressFinger</td>
<td>Check if a finger is placed on the sensor</td>
</tr>
<tr>
<td>40</td>
<td>DeleteID</td>
<td>Delete the fingerprint with the specified ID</td>
</tr>
<tr>
<td>41</td>
<td>DeleteAll</td>
<td>Delete all fingerprints from the database</td>
</tr>
<tr>
<td>50</td>
<td>Verify</td>
<td>1:1 Verification of the capture fingerprint image with the specified ID</td>
</tr>
<tr>
<td>51</td>
<td>Identify</td>
<td>1:N Identification of the capture fingerprint image with the database</td>
</tr>
<tr>
<td>52</td>
<td>VerifyTemplate</td>
<td>1:1 Verification of a fingerprint template with the specified ID</td>
</tr>
<tr>
<td>53</td>
<td>IdentifyTemplate</td>
<td>1:N Identification of a fingerprint template with the database</td>
</tr>
<tr>
<td>60</td>
<td>CaptureFinger</td>
<td>Capture a fingerprint image(256x256) from the sensor</td>
</tr>
<tr>
<td>61</td>
<td>MakeTemplate</td>
<td>Make template for transmission</td>
</tr>
<tr>
<td>62</td>
<td>GetImage</td>
<td>Download the captured fingerprint image(256x256)</td>
</tr>
<tr>
<td>63</td>
<td>GetRawImage</td>
<td>Capture &amp; Download raw fingerprint image(320x240)</td>
</tr>
<tr>
<td>70</td>
<td>GetTemplate</td>
<td>Download the template of the specified ID</td>
</tr>
<tr>
<td>71</td>
<td>SetTemplate</td>
<td>Upload the template of the specified ID</td>
</tr>
<tr>
<td>72</td>
<td>GetDatabaseStart</td>
<td>Start database download, <strong>obsolete</strong></td>
</tr>
<tr>
<td>Number (HEX)</td>
<td>Alias</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>73</td>
<td>GetDatabaseEnd</td>
<td>End database download, obsolete</td>
</tr>
<tr>
<td>F0</td>
<td>SetSecurityLevel</td>
<td>Set Security Level</td>
</tr>
<tr>
<td>F1</td>
<td>GetSecurityLevel</td>
<td>Get Security Level</td>
</tr>
<tr>
<td>F4</td>
<td>Identity_Template2</td>
<td>Identify of the capture fingerprint image with the specified template</td>
</tr>
<tr>
<td>F9</td>
<td>EnterStandbyMode</td>
<td>Enter Standby Mode (Low power mode)</td>
</tr>
<tr>
<td>30</td>
<td>Ack</td>
<td>Acknowledge</td>
</tr>
<tr>
<td>31</td>
<td>Nack</td>
<td>Non-acknowledge</td>
</tr>
</tbody>
</table>
4. **Protocol: Error Codes**

When a response packet is a non-acknowledge, *Parameter* represents an error code as below.

<table>
<thead>
<tr>
<th>NACK Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NACK_TIMEOUT</td>
<td>0x1001</td>
<td>Obsolete, capture timeout</td>
</tr>
<tr>
<td>NACK_INVALID_BAUDRATE</td>
<td>0x1002</td>
<td>Obsolete, Invalid serial baud rate</td>
</tr>
<tr>
<td>NACK_INVALID_POS</td>
<td>0x1003</td>
<td>The specified ID is not between 0-999</td>
</tr>
<tr>
<td>NACK_IS_NOT_USFD</td>
<td>0x1004</td>
<td>The specified ID is not used</td>
</tr>
<tr>
<td>NACK_IS_ALREADY_USED</td>
<td>0x1005</td>
<td>The specified ID is already used</td>
</tr>
<tr>
<td>NACK_COMM_ERR</td>
<td>0x1006</td>
<td>Communication Error</td>
</tr>
<tr>
<td>NACKVERIFY_FAILED</td>
<td>0x1007</td>
<td>1:1 Verification Failure</td>
</tr>
<tr>
<td>NACK_IDENTIFY_FAILED</td>
<td>0x1008</td>
<td>1:N Identification Failure</td>
</tr>
<tr>
<td>NACK_DB_IS_FULL</td>
<td>0x1009</td>
<td>The database is full</td>
</tr>
<tr>
<td>NACK_DB_IS_EMPTY</td>
<td>0x100A</td>
<td>The database is empty</td>
</tr>
<tr>
<td>NACK_TURN_ERR</td>
<td>0x100B</td>
<td>Obsolete, Invalid order of the enrollment</td>
</tr>
<tr>
<td>(The order was not as:</td>
<td></td>
<td>EnrollStart -&gt; Enroll1 -&gt;</td>
</tr>
<tr>
<td>Enroll2 -&gt; Enroll3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NACK_BAD_FINGER</td>
<td>0x100C</td>
<td>Too bad fingerprint</td>
</tr>
<tr>
<td>NACK_ENROLL_FAILED</td>
<td>0x100D</td>
<td>Enrollment Failure</td>
</tr>
<tr>
<td>NACK_IS_NOT_SUPPORTED</td>
<td>0x100E</td>
<td>The specified command is not supported</td>
</tr>
<tr>
<td>NACK_DEV_ERR</td>
<td>0x100F</td>
<td>Device Error, especially if Crypto-Chip is trouble</td>
</tr>
<tr>
<td>NACK_CAPTURE_CANCELED</td>
<td>0x1010</td>
<td>Obsolete, The capturing is canceled</td>
</tr>
<tr>
<td>NACK_INVALID_PARAM</td>
<td>0x1011</td>
<td>Invalid parameter</td>
</tr>
<tr>
<td>NACK_FINGER_IS_NOT_PRESSED</td>
<td>0x1012</td>
<td>Finger is not pressed</td>
</tr>
<tr>
<td>Duplicated ID</td>
<td>0 – 2999</td>
<td>There is duplicated fingerprint (while enrollment or setting template). This error describes just duplicated ID</td>
</tr>
</tbody>
</table>
5. **Protocol: Command Details**

5.1. **Initialization** (*Open*)

- **COMMAND PACKET**
  - Command = *Open*
  - Parameter =
    - 0: not to get extra info
    - Nonzero: to get extra info

- **RESPONSE PACKET**
  - Response = *Ack*
  - If host requested to get extra

- **DATA PACKET**
  - Data = `typedef struct_devinfo {
    DWORD FirmwareVersion;
    DWORD IsoAreaMaxSize;
    BYTE DeviceSerialNumber[16];
  } devinfo;`

*Open* command is used to initialize the device; especially it gets device's static info.
Description of *devinfo* structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Sample</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>FirmwareVersion</em></td>
<td><em>FirmwareVersion: 20170313.</em></td>
<td>Firmware version</td>
</tr>
<tr>
<td><em>IsoAreaMaxSize</em></td>
<td><em>IsoAreaMaxSize: 0 KB</em></td>
<td>Maximum size of ISO CD image</td>
</tr>
<tr>
<td><em>DeviceSerialNumbe</em> <em>r</em></td>
<td><em>DeviceSN: EF15EF4016C66250-889F1A4139000000</em></td>
<td>Unique serial number of the device</td>
</tr>
</tbody>
</table>

If the Device’s Serial Number is zero, then there is no guarantee for stable operation of the device.
5.2. **Termination** *(Close)*

<table>
<thead>
<tr>
<th>Host</th>
<th>Device</th>
</tr>
</thead>
</table>
| **COMMAND PACKET**
Command = *Close* | **RESPONSE PACKET**
Response = *Ack* |

*Close* command does nothing.

5.3. **Fast searching of the device** *(UsbInternalCheck)*

<table>
<thead>
<tr>
<th>Host</th>
<th>Device</th>
</tr>
</thead>
</table>
| **COMMAND PACKET**
Command = *UsbInternalCheck* | **RESPONSE PACKET**
Response = *Ack*
Parameter = 0x55 |

The device operates as removable CD drive. If another removable CD drive exists in the system, connection time maybe will be long. To prevent this, *UsbInternalCheck* command is used for fast searching of the device.
5.4. CMOS LED control (CmosLed)

**COMMAND PACKET**
- Command = *CmosLed*
- Parameter =
  - 0: Off LED
  - 1: On LED

**RESPONSE PACKET**
- Response = *Ack*

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.
5.5. Changing UART baud rate (*ChangeBaudrate*)

**COMMAND PACKET**
Command = *ChangeBaudrate*
Parameter = *baud rate*
(9600,19200,38400,57600,115200)

**RESPONSE PACKET**
Response = *Ack*:
Response = *Nack*: Error
*NACK_INVALID_PARAM*

This command changes the UART baud rate at the run-time.
The device initializes its UART baud rate to 9600 bps after power on.
5.6. Get enrolled fingerprint count (GetEnrollCount)

**COMMAND PACKET**
Command = GetEnrollCount

**RESPONSE PACKET**
Response = Ack
Parameter = Enrolled Fingerprint Count

5.7. Check enrollment status (CheckEnrolled)

**COMMAND PACKET**
Command = CheckEnrolled
Parameter = ID(0~2999)

**RESPONSE PACKET**
Response = Ack: This ID is enrolled
Response = Nack: Error
NACK_INVALID_POS
NACK_IS_NOT_USED
5.8. Start an enrollment (EnrollStart)

**COMMAND PACKET**
Command = EnrollStart
Parameter = ID(0~2999)
(If Parameter’s HIWORD is non-zero, fingerprint duplication check will not be)
If ID == -1, then “Enrollment without saving” will be stated.

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_DB_IS_FULL
NACK_INVALID_POS
NACK_IS_ALREADY_USED

5.9. Make 1st template for an enrollment (Enroll1)

**COMMAND PACKET**
Command = Enroll1

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_ENROLL_FAILED
NACK_BAD_FINGER
5.10. Make 2\textsuperscript{nd} template for an enrollment(Enroll2)

**COMMAND PACKET**

Command = \textit{Enroll2}

**RESPONSE PACKET**

Response = \textit{Ack}: OK  
Response = \textit{Nack}: Error  
\texttt{NACK\_ENROLL\_FAILED}  
\texttt{NACK\_BAD\_FINGER}

5.11. Make 3\textsuperscript{rd} template for an enrollment, merge three templates(Enroll3)

**COMMAND PACKET**

Command = \textit{Enroll3}

**RESPONSE PACKET**

Response = \textit{Ack}: OK  
Response = \textit{Nack}: Error  
\texttt{NACK\_ENROLL\_FAILED}  
\texttt{NACK\_BAD\_FINGER}  
\texttt{Duplicated\ ID(0~2999)}

If \textit{EnrollStart} was called with \texttt{ID} == -1, then the enrolled template will not be saved, it will be transferred to the host, as below.

**DATA PACKET**

Data = enrolled template (498 bytes)
To enroll a fingerprint, the host must issue above 4 commands, later chapter describes how to organize these commands.
5.12. Check finger pressing status(*IsPressFinger*)

**COMMAND PACKET**
Command = *IsPressFinger*

**RESPONSE PACKET**
Response = *Ack*:
Parameter = 0: finger is pressed
Parameter = nonzero: finger is not pressed

This command is used while enrollment, the host waits to take off the finger per enrollment stage.
5.13. Delete one fingerprint(*DeleteID*)

**COMMAND PACKET**
Command = *DeleteID*
Parameter = *ID(0~2999)*

**RESPONSE PACKET**
Response = *Ack*: OK
Response = *Nack*: Error
*NACK_INVALID_POS*

5.14. Delete all fingerprints(*DeleteAll*)

**COMMAND PACKET**
Command = *DeleteAll*

**RESPONSE PACKET**
Response = *Ack*: OK
Response = *Nack*: Error
*NACK_DB_IS_EMPTY*
5.15. 1:1 Verification (Verify)

**COMMAND PACKET**
Command = Verify
Parameter = ID(0~2999)

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_INVALID_POS
NACK_IS_NOT_USED
NACK_VERIFY_FAILED

5.16. 1:N Identification (Identify)

**COMMAND PACKET**
Command = Identify

**RESPONSE PACKET**
Response = Ack: OK
Parameter = 0~2999: identified ID
Response = Nack: Error
NACK_DB_IS_EMPTY
NACK_IDENTIFY_FAILED
5.17. 1:1 Verification of Template (VerifyTemplate)

**COMMAND PACKET**
- Command = Verify
- Parameter = ID(0~2999)

**DATA PACKET**
- Data = a template (498 bytes)

**RESPONSE PACKET**
- Response = Ack: OK
- Response = Nack: Error
  - NACK_INVALID_POS
  - NACK_IS_NOT_USED

**RESPONSE PACKET**
- Response = Ack: OK
- Response = Nack: Error
  - NACK_COMM_ERR
  - NACK_VERIFY_FAILED
5.18. 1:N Identification of Template (IdentifyTemplate)

**COMMAND PACKET**
Command = IdentifyTemplate

**DATA PACKET**
Data = a template (498 bytes)

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_DB_IS_EMPTY

**RESPONSE PACKET**
Response = Ack: OK
Parameter = 0~2999: identified ID
Response = Nack: Error
NACK_COMM_ERR
NACK_IDENTIFY_FAILED
5.19. Capture fingerprint(*CaptureFinger*)

**COMMAND PACKET**
Command = *CaptureFinger*
Parameter =
0: not best image, but fast
Nonzero: best image, but slow

**RESPONSE PACKET**
Response = *Ack*: OK
Response = *Nack*: Error
*NACK_FINGER_IS_NOT_PRESSED*

The fingerprint algorithm uses **450dpi** 256x256 image for its input.
This command captures raw image from the sensor and converts it to 256x256 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.
5.20. Make Template (*MakeTemplate*)

This function makes template for transmission. *CaptureFinger* command should be previously issued. Do not use the template for registration.
5.21. Get fingerprint image(*GetImage*)

COMMAND PACKET
Command = *GetImage*

RESPONSE PACKET
Response = *Ack: OK*

DATA PACKET
Data = 258x202 image (52116 bytes)
5.22. Get raw image (GetRawImage)

**COMMAND PACKET**
Command = GetRawImage

**Response Packet**
Response = Ack: OK

**Data Packet**
Data = 160x120 QVGA image (19200 bytes)

5.23. Get template (GetTemplate)

**COMMAND PACKET**
Command = GetTemplate
Parameter = ID(0~2999)

**Response Packet**
Response = Ack: OK
Response = NACK: Error
NACK_INVALID_POS
NACK_IS_NOT_USED

**Data Packet**
Data = a template (498 bytes)
5.24. Set template (SetTemplate)

**COMMAND PACKET**
Command = SetTemplate
Parameter = ID(0~2999)
(If Parameter's HIWORD is non-zero, fingerprint duplication check will not be performed.)

**DATA PACKET**
Data = a template (498 bytes)

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_INVALID_POS

5.25. Start database download, obsolete (GetDatabaseStart)

**COMMAND PACKET**
Command = GetDatabaseStart

**RESPONSE PACKET**
Response = Ack

GetDatabaseStart command does nothing. It exists for historical reason; it was used for RS232 communication.
5.26. End database download, obsolete\texttt{(GetDatabaseEnd)}

\begin{center}
\begin{tikzpicture}
\node (host) at (0,0) {Host};
\node (device) at (4,0) {Device};
\node [draw, rounded corners] at (2,1) {
\textbf{COMMAND PACKET}
Command = \texttt{GetDatabaseEnd}
};
\node [draw, rounded corners] at (2,-1) {
\textbf{RESPONSE PACKET}
Response = \texttt{Ack}
};
\draw [->] (host) -- (device);
\end{tikzpicture}
\end{center}

\textit{GetDatabaseEnd} command does nothing. It exists for historical reason; it was used for RS232 communication.
5.27. Set SecurityLevel

COMMAND PACKET
Command = SetSecurityLevel
Parameter = 1~5
Parameter is equal 5. The level is the highest security level
Default value is 3.

RESPONSE PACKET
Response = Ack

5.28. GetSecurityLevel

COMMAND PACKET
Command = GetSecurityLevel

RESPONSE PACKET
Response = Ack
Parameter = Security Level
5.29. EnterStandbyMode

Enter Standby Mode (Low power mode). If user want the fingerprint module to leave standby mode, please send the 0x00 first and wait 20mS to weak up and then send the standard command.
5.30. Identify_Template2

**COMMAND PACKET**
Command = Identify_Template2
Parameter = 500

**DATA PACKET**
Data = Byte[2] + a template (498 bytes)
Byte[0] = 1 and Byte[1] = 1

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_INVALID_PARAM

**RESPONSE PACKET**
Response = Ack: OK
Response = Nack: Error
NACK_IDENTIFY_FAILED

6.1. Capture of the fingerprint image

IsPressFinger checks whether a finger placed on the sensor. This function is used especially while enrollment.

CaptureFinger captures a fingerprint image (256x256), if a finger isn’t placed on the sensor, it returns with error. If this function returns with success, the device’s internal RAM keeps valid fingerprint image for the subsequent commands. If the host issues other command, the fingerprint image will be used and destroyed.

GetRawImage captures a raw live image (320x240), it doesn’t check whether a finger placed on the sensor, this function is used for debug or calibration.

6.2. Identifying and Verifying

Identify and IdentifyTemplate perform 1: N matching operation. Verify and VerifyTemplate perform 1: 1 matching operation.

Just before calling of image-related matching functions (Identify, Verify), the host must call CaptureFinger.

6.3. Enrollment

An enrollment flowchart is as below.

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. **PC Demo**

PC demo program describes how to use the device with its source code.

![Demo program interface](image)

**Demo program is supported with its source code.**

**The project is Microsoft Visual C++ 2005 project.**

**We selected VC2005 to minimize the size of the executable.**