

FM Series

Packet Protocol Manual

Version 2.7

CONTENTS

REVISION NOTES	9
PACKET PROTOCOL.....	14
NETWORK PACKET PROTOCOL	16
BROADCAST PACKET PROTOCOL.....	17
SYSTEM PARAMETER SUMMARY	18
Timeout	23
Template Size	24
Enroll Mode.....	25
Security Level.....	27
Encryption Mode	29
Sensor Type.....	30
Image Format.....	31
Module ID	32
Firmware Version	33
Serial Number	34
Baudrate.....	35
Baudrate2.....	36
Enrolled Finger	37
Available Finger.....	38
Send Scan Success	39
ASCII Packet	40

Rotate Image	41
Rotation	42
Sensitivity	43
Image Quality	44
Auto Response	45
Network Mode	46
Free Scan	48
Provisional Enroll	49
Pass When Empty	50
Response Delay	51
Matching Timeout	52
Build Number	53
Enroll Displacement	54
Lighting Condition	55
Free Scan Delay	56
Fast Mode	57
Watchdog	58
Template Type	59
COMMAND SUMMARY	60
Flag / Error Code	64
SW : System Parameter Write	66
SF : System Parameter Save	68
SR : System Parameter Read	70

CS : Calibrate Sensor	72
SS : System Status Check.....	74
CA : Cancel	76
ID : Get Module ID	78
UG : Upgrade Firmware	81
RS : Reset Module.....	84
LM : Lock Module.....	86
UM : Unlock Module.....	88
MP : Set Master Password	90
ES : Enroll by Scan	92
ESA : ES with Administrator's Verification	101
EI : Enroll by Image.....	106
EIX : EI with Extended Data Transfer Protocol	109
ET : Enroll by Template	112
ETX : ET with Extended Data Transfer Protocol.....	115
EW : Enroll by Wiegand ID	118
EWA : EW with Administrator's Verification	120
VS : Verify by Scan.....	123
VI : Verify by Image	125
VIX: VI with Extended Data Transfer Protocol	127
VT : Verify by Template.....	129
VW : Verify by Wiegand ID.....	131
VH : Verify Host Template by Scan	133

WSL : Write Security Level of a User	135
RSL : Read Security Level of a User	137
IS : Identify by Scan	139
II : Identify by Image.....	141
IIX : II with Extended Data Transfer Protocol.....	143
IT : Identify by Template	145
DA : Delete All Templates	147
DAA : DA with Administrator's Verification	149
DT : Delete Template	151
DS : Delete by Scan	153
DSA : DS with Administrator's Verification.....	155
DW : Delete by Wiegand ID	157
DWA : DW with Administrator's Verification.....	159
LT : List User ID.....	161
LTX : LT with Extended Data Transfer Protocol.....	164
CT : Check User ID.....	167
FP : Fix All Provisional Templates	169
DP : Delete All Provisional Templates.....	171
RI : Read Image	173
RIX : RI with Extended Data Transfer Protocol.....	179
SI : Scan Image.....	181
SIX : SI with Extended Data Transfer Protocol	183
RT : Read Template.....	185

RTX : RT with Extended Data Transfer Protocol	187
ST : Scan Template	189
KS : Scan Template with Challenge Data	191
KW : Write Encryption Key	193
ML : Get Size of User Memory	195
MW : Write to User Memory	197
MR : Read from User Memory	199
TW : Write Current Time	201
TR : Read Current Time	203
LN : Get Number of Log Data	205
LR : Read Log Data	207
LD : Delete Log Data	209
LC : Set/Get the Custom Log Field	211
RCL : Read Current Log in Cache	213
CCL : Clear Log Cache	215
WW : Write Wiegand Configuration (Deprecated)	217
WR : Read Wiegand Configuration (Deprecated)	219
WG : Get Wiegand Input (Deprecated)	221
WS : Set Wiegand Output (Deprecated)	223
WM : Map Wiegand ID to Input Function	225
WL : List Wiegand ID Mapping	227
WC : Clear Wiegand ID Mapping	229
WWX : Write Wiegand Configuration	231

WRX : Read Wiegand Configuration	233
WGX : Get Wiegand Input.....	235
WSX : Set Wiegand Output	237
WFW : Set Wiegand Field	239
WFR : Get Wiegand Field	241
WPW : Write Wiegand I/O Setting	243
WPR : Read Wiegand I/O Setting	245
IW : Write Input Configuration	247
IR : Read Input Configuration	250
IG : Get Input State.....	252
OW : Write Output Configuration.....	254
OR : Read Output Configuration	258
OL : Read Output Configuration List.....	260
OS : Set Output State.....	262
GW : Write GPIO Configuration	264
GR : Read GPIO Configuration	268
GC : Clear GPIO Configuration	270
GD : Set Default GPIO Configuration	272
AW : Write Administration Level of a User	274
AR : Read Administration Level of a User	276
AC : Clear All the Administration Levels.....	278
UW: Write Authentication Mode of a User	280
UR : Read Authentication Mode of a User	282

UC : Reset Authentication Modes of All Users.....	284
UL : List User ID by Authentication Mode.....	286
ABL : Add a User to the Blacklist	288
DBL : Delete a User ID from the Blacklist	290
RBL : Read the Blacklist.....	292
CBL : Clear the Blacklist	294
WME : Write Entrance Limit	296
RME : Read Entrance Limit	298
CME : Clear Entrance Limit	300
APPENDIX A. GPIO CONFIGURATION.....	302
APPENDIX B. EXTENDED DATA TRANSFER PROTOCOL.....	309
APPENDIX C. EXTENDED WIEGAND PROTOCOL.....	312
APPENDIX D. PACKET PROTOCOL FOR BIOENTRY™.....	316
Overview of BioEntry™.....	316
FM Module vs. BioEntry™	317
Commands for BioEntry™ Smart	318
CR : Read Smartcard	319
CW : Write Smartcard	322
CF : Format Smartcard	324
CC : Configure Card Input Function.....	326
CG : Get Card Input Function	328
VC : Verify by Smartcard	330
ECX : Enroll Templates to Smartcard.....	332

CKW : Write Site Key	334
CKR : Read Site Key Option.....	336
CLW : Write Card Layout	337
CLR : Read Card Layout.....	339

Revision Notes

- V1.0 2002-07-08 Created.
- V1.1 2002-11-09 VH command added.
- V1.2 2003-01-14 Minor typo corrected.
- V1.3 2003-02-25 CT, RS command added.
ES, ET, EP command support auto ID.
- V1.4 2003-03-24 SendScanSuccess system parameter added.
- V1.5 2004-01-15 Network protocol supported.
GPIO configuration (GR, GW, GC, GD command) supported.
SI, FP, DP, KW, KS command added.
ASCIIPacket, RotateImage, Sensitivity, ImageQuality, AutoResponse, NetworkMode, FreeScan, ProvisionalEnroll, PassWhenEmpty, ResponseDelay system parameter added.
IS, II, IT command support group identification.
EI(EP), VI(VP), II(IP), RI(RP), ST(RS) command name changed.
- V1.6 2004-04-08 IS, II, IT command support timeout for matching.
MatchingTimeout system parameter added.
TIMEOUT_MATCH error code added.
VH command supports multiple templates.
- V1.7 2004-05-06 BuildNumber system parameter added.
ImageQuality system parameter has strongest qualification value.
- V1.8 2004-07-02 CHECK_ID flag code for ES, EI, ET command is added.
EnrollDisplacement system parameter is added.

ImageFormat system parameter has 4-bit gray image value.

Number of fingerprints which can be enrolled in the same ID is changed from 2 to 10.

V1.9 2004-09-25 CS command added.

EnrollMode system parameter has 2 templates & 2 templates II.

SecurityLevel system parameter has additional levels.

Baudrate system parameter has additional baudrates.

(*The changes in V1.9 are only applicable to the FM series.)

V2.0 2004-11-11 DS command added.

EW, VW, DW command added.

ML, MW, MR command added.

TW, TR, LN, LR, LD command added.

WW, WR, WG, WS command added.

IW, IR, IG command added.

OW, OR, OL, OS command added.

Baudrate2 system parameter added.

NetworkMode system parameter has full duplex mode.

AutoResponse system parameter has auxiliary port.

(*The changes in V2.0 are only applicable to the FM series.)

V2.1 2004-12-6 BUSY, CANCELED, DATA_ERROR, DATA_OK error code added.

BUSY status code added.

CA command added.

WM, WL, WC command added.

AW, AR, AC command added.

ESA, EWA, DSA, DWA, DAA command added.

Extended Data Transfer Protocol added.

EIX, IIX, VIX, SIX, RIX command added.

(*The changes in V2.1 are only applicable to the FM series.)

- V2.2 2005-01-18 Template Size system parameter added.
Rotation system parameter added.
Enroll commands(ES, EW, ESA, EWA, EI) return image quality score
- V2.2.1 2005-02-14 Description of Template Size system parameter added.
- V2.3 2005-04-29 Broadcast Packet Protocol added.
ID command added.
CHECK_FINGER and CHECK_FINGER_AUTO_ID flag added.
EXIST_FINGER error code added.
Block index and block size parameters are added to LT command.
Minimum security level is changed to 1/10,000.
Each of system parameter is described in its own sub section.
- V2.3.1 2005-06-01 Free Scan Delay system parameter added.
- V2.4 2005-07-11 Broadcast Packet Protocol is changed from 13 bytes to 15 bytes.
In network mode, the modules only respond to 15 byte network packets.
Lighting Condition system parameter added.
Fast Mode system parameter added
ADD_DURESS flag added.
DURESS_FINGER error code added.

Tamper Switch In input function added.

Verify Duress Finger, Identify Duress Finger output event added.

Tamper Switch On, Tamper Switch Off output event added.

UG command added.

LTX command added.

WWX, WRX, WGX, WSX, WFW, WFR, WPW, WPR command added.

V2.5 2005-11-15 ETX, RTX command added.

ADD_CHECKSUM option is added to ST command.

DELETE_ONLY_ONE and DELETE_MULTIPLE_ID options are added to DT command.

RS command added.

System Started output event added.

Watchdog system parameter added.

The default value of Lighting Condition parameter is changed to Outdoor.

The default value of Free Scan Delay parameter is changed to 1 second.

LM, UM, MP command added.

LOCKED error code added.

Appendix D. Packet Protocol for BioEntry™ added.

V2.6 2006-03-23 LC command added.

CCL, RCL command added.

WSL, RSL command added. The minimum security level is lowered to 1/1,000 for 1:1 matching.

AUTOMATIC NORMAL, AUTOMATIC SECURE,

AUTOMATIC MORE SECURE security level added.

V2.7 2006-11-13 Template Type system parameter is added to support the standard template format defined in ISO 19794-2.

UW, UR, UC, UL command added for FM series.

WME, RME, CME command added for FM series.

ABL, DBL, RBL, CBL command added for FM series.

ENTRANCE_LIMIT, REJECTED_ID error code added.

AUTOMATIC fast mode added.

Packet Protocol

In the packet protocol of FM Modul, 1 packet is 13 bytes long and its structure is as follows

Start code	Command	Param	Size	Flag/Error	Checksum	End code
1byte	1byte	4bytes	4bytes	1byte	1byte	1byte

1. Start code: 1 byte. Indicates the beginning of a packet. It always should be 0x40.
 2. Command: 1 byte. Refer to the Command Table in a later chapter of this document.
 3. Param: 4 bytes. Indicates user ID or system parameters.
 4. Size: 4 bytes. Indicates the size of binary data following the command packet such as fingerprint templates or images.
 5. Flag/Error: 1 byte. Indicates flag data in the request command sent to the module, and error code in the response command received from the module, respectively.
 6. Checksum: 1 byte. Checks the validity of a packet. Checksum is a remainder of the sum of each field, from the Start code to Flag/Error, divided by 256 (0x100).
 7. End code: 1 byte. Indicates the end of a packet. It always should be 0x0A. It is also used as a code indicating the end of a binary data such as fingerprint templates.
- The packet transmitted between the host and the module has the same structures. The commands transmitting from the host to the module are called “request command,” and the response transmitting from the module to the hosts, “response command.”
 - In general, the host receives one response command for one request command. However, it may receive two response commands for some commands such as Enroll by Scan. The first command comes in the intermediate stage, notifying image scanning completed and the other

command notifies process result. If you do not want to receive the intermediate command, you can disable it by changing a system parameter.

- The fingerprint templates vary in size by sensor and firmware version.
- In the following document, # indicates numbers and the number of # specifies the number of digits. N/A indicates that any value can be applied to the field and Null indicates 0x00.
- In the transmission of actual data, the byte order is little endian, the lowest byte is transmitted first. For instance, when transmitting data of 400 (0x190) in the Size field, which is 4 bytes long, the data transmission order is as follows: 0x90, 0x01, 0x00, 0x00. The rule also holds for the data received from the module.
- Take ES command (0x05) as an example, the following is the data transmission order and computation of the checksum of the command:

When enrolling a new fingerprint with an ID, '0x9929':

Start	Command	Param	Size	Flag/Error	Checksum	End
0x40	0x05	0x9929	0x00	0x00	0x07	0x0A

Actual values input in the Param:

1 st place	2 nd place	3 rd place	4 th place
0x29	0x99	0x00	0x00

Byte transmission order:

0x40, 0x05, 0x29, 0x99, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0A

To compute the checksum, first compute the sum of each byte of the packet (from start code to flag/error) as demonstrated below:

$$0x40+0x05+0x29+0x99+0x00+0x00+0x00+0x00+0x00+0x00 = 0x107$$

Divide the above value by 0x100(256). The remainder of this division is 0x07 and this value is the checksum.

Network Packet Protocol

In order to support RS422 or RS485 network interfaces, FM module modules support network packet protocols. Network packet is composed of 15 bytes, whose start code is different from the standard packet, and includes 2 bytes for terminal ID. The terminal ID is correspondent to the lower 2 bytes of Module ID of system parameter.

Field	Start code	Terminal ID	Command	Param	Size	Flag / Error	Checksum	End code
Bytes	1	2	1	4	4	1	1	1
Value	0x41	1 ~ 0xFFFF	Same as standard protocol				Checksum of 13 bytes	0x0A

The contents of the network packet, including command, param, size, and flag are same as those of standard packet. Checksum field is the checksum value of preceding 13 bytes.

Until firmware V1.3, FM modules respond both standard and network packets regardless of Network Mode system parameter. However, since firmware V1.4, the modules only respond to 15 byte network packets if Network Mode system parameter is on. See Network Mode system parameter for details.

Broadcast Packet Protocol

In RS422 or RS485 network environments, a host can send broadcast packets to all the modules in the network. The only difference between network packets and broadcast packets is that the Terminal ID field of the latter should be 0x00.

Field	Start code	Terminal ID	Command	Param	Size	Flag / Error	Checksum	End code
Bytes	1	2	1	4	4	1	1	1
Value	0x41	0x00	Same as network protocol					

All the modules in the network process the broadcast packet, but do not send response packet to prevent packet collisions. Therefore, if the host wants to confirm that the broadcast request is handled correctly, it should send another request packet to each module using Network Packet Protocol. There is only one exception to this rule. See ID command for details.

Broadcast Packet Protocol is provided for FM only. The 13 byte broadcast packet protocol introduced in V1.3 firmware is not supported any longer.

Three types of packet protocol can be summarized as follows:

	Start Code	Packet Length	Communication Type	Operation Mode	Response Packet
Standard	0x40	13	Peer-to-peer	Single	O
Network	0x41	15	Peer-to-peer	Network	O
Broadcast	0x41(Terminal ID: 0x00)	15	1 to N	Network	X

System Parameter Summary

Name	Code	Description	Value (* denotes default value)
Timeout	0x62	Timeout period	0x30 : infinite 0x31 : 1 second ... *0x3A : 10 seconds ... 0x44 : 20 seconds
Template Size ⁽¹⁾	0x64	Template size. When the Encryption Mode is on, it should be a multiple of 32.	Integer between 256 and *384 ⁽⁴⁾
Enroll Mode	0x65	Enroll mode	0x30 : 1 time 0x31 : 2 times (1 request command & 2 response commands) 0x32 : 2 times II (2 request commands & 2 response commands) *0x41 : 2 templates (1 request command & 2 response commands) ⁽¹⁾ 0x42 : 2 templates II (2 request commands & 2 response commands) ⁽¹⁾ (*
Security Level	0x66	Security level	0x30 : 1/10 FAR(False Acceptance Rate) ⁽¹⁾⁽⁵⁾ 0x31 : 1/100 ⁽⁵⁾ 0x32 : 1/1,000 ⁽⁵⁾ 0x33 : 1/10,000 0x34 : 1/100,000 0x35 : 1/1,000,000 0x36 : 1/10,000,000 ⁽¹⁾ 0x37 : 1/100,000,000 ⁽¹⁾ 0x40 : 3/100 ⁽¹⁾⁽⁵⁾

			<p>0x41 : 3/1,000 ⁽¹⁾⁽⁵⁾</p> <p>0x42 : 3/10,000 ⁽¹⁾⁽⁵⁾</p> <p>0x43 : 3/100,000 ⁽¹⁾</p> <p>0x44 : 3/1,000,000 ⁽¹⁾</p> <p>0x45 : 3/10,000,000 ⁽¹⁾</p> <p>0x46 : 3/100,000,000 ⁽¹⁾</p> <p>*0x50 : Automatic Normal⁽¹⁾</p> <p>0x51 : Automatic Secure⁽¹⁾</p> <p>0x52 : Automatic More Secure⁽¹⁾</p>
Encryption Mode	0x67	Encryption mode	<p>*0x30 : Encryption off</p> <p>0x31 : Encryption on</p>
Sensor Type	0x68	Sensor type	<p>0x30 : Infineon FingerTip</p> <p>0x31 : Atmel FingerChip</p> <p>0x32 : AuthenTec FingerLoc AF-S2</p> <p>0x33 : Optical fingerprint sensor</p> <p>0x34 : STMicro TouchChip</p> <p>0x35 : BMF BLP-100</p> <p>0x36 : Second optical fingerprint sensor</p> <p>0x37 : Fujitsu MBF310</p> <p>0x38 : Third optical fingerprint sensor</p> <p>0x39 : Fourth optical fingerprint sensor</p> <p>0x3A : AuthenTec FingerLoc AFS8600</p>
Image Format	0x6C	Image format	<p>0x30 : Gray image</p> <p>*0x31 : Binary image</p> <p>0x32 : 4 bit gray image</p>
Module ID	0x6D	Module ID	Integer 0 ~ 65535
Firmware Version	0x6E	Firmware version	4bytes character
Serial Number	0x6F	Module serial number	Integer (4 bytes)
Baudrate	0x71	Host baudrate setting	<p>0x31 : 9600 bps</p> <p>0x32 : 19200 bps</p>

			0x33 : 38400 bps 0x34 : 57600 bps *0x35 : 115200 bps
Baudrate2 ⁽²⁾	0x72	Auxiliary baudrate setting	0x31 : 9600 bps 0x32 : 19200 bps 0x33 : 38400 bps 0x34 : 57600 bps *0x35 : 115200 bps
Enrolled Finger	0x73	Current number of fingerprints enrolled	Integer (4 bytes)
Available Finger	0x74	The available number of fingerprints that can be enrolled	Integer (4 bytes)
Send Scan Success	0x75	Enable sending SCAN_SUCCESS response	0x30 : No SCAN_SUCCESS message *0x31 : Send SCAN_SUCCESS message
ASCII Packet	0x76	Flag for packet exchange though HEX-ASCII format	*0x30 : Hexadecimal packet 0x31 : ASCII packet
Rotate Image	0x77	Flag for rotating sensor image in capture	*0x30 : Upright image 0x31 : Upside down image
Rotation ⁽¹⁾	0x78	Maximum allowable rotation for matching	0x31: 15 degree 0x32: 30 degree 0x33: 45 degree 0x34: 60 degree 0x35: 75 degree *0x36: 90 degree (*For Atmel's FingerChip sensor, 0x33 is default)
Sensitivity	0x80	Parameter for sensor sensitivity	0x30 : Least sensitive ... *0x37 : Most sensitive
Image Quality	0x81	Parameter for qualifying scanned image	0x30 : Weak qualification *0x31 : Moderate qualification

			0x32 : Strong qualification 0x33 : Strongest qualification
Auto Response	0x82	Flag for sending automatic response as the result of GPIO input or FreeScan	*0x30 : No response command 0x31 : Send response command (host) 0x32 : Send response command (aux) ⁽²⁾ 0x33 : Send response command (both) ⁽²⁾
Network Mode	0x83	Flag for default operation mode	*0x30 : Single mode 0x31 : Network mode (half duplex) 0x32 : Network mode (full duplex) ⁽²⁾
Free Scan	0x84	Scan always fingerprint images for identification on idle state	*0x30 : Normal mode 0x31 : Free scan mode
Provisional Enroll	0x85	Save enrolled templates at flash memory permanently or not	*0x30 : Permanent enrollment 0x31 : Provisional enrollment
Pass When Empty	0x86	Pass or fail when fingerprint DB is empty	*0x30 : Fail when DB is empty 0x31 : Pass when DB is empty
Response Delay	0x87	Delay for response command	*0x30 : No delay 0x31 : 20 msec ... 0x35 : 100 msec ... 0x3A : 200 msec
Matching Timeout	0x88	Timeout period for matching in identification	*0x30 : infinite 0x31 : 1 second ... 0x3A : 10 seconds ... 0x44 : 20 seconds
Build Number	0x89	Build number	4bytes character
Enroll Displacement ⁽³⁾	0x8A	Displacement between two fingerprints for enrollment in case EnrollMode parameter is 2	*0x30 : No check 0x31 : Above 1 mm away

		times or 2 templates	... 0x35 : Above 5 mm away ... 0x3A : Above 10 mm away
Lighting Condition ⁽¹⁾	0x90	Tune optical sensors based on lighting conditions	*0x30: Outdoor 0x31: Indoor
Free Scan Delay ⁽¹⁾	0x91	Delay between consecutive identification processes in Free Scan Mode.	0x30: No delay *0x31: 1 second ... 0x40: 10 seconds
Fast Mode ⁽¹⁾	0x93	Fast mode for 1:N matching	0x30: Normal 0x31: Fast mode 1 ... 0x35: Fast mode 5(Fastest) *0x36: Automatic
Watchdog ⁽¹⁾	0x94	Watchdog timer	0x30: Don't use Watchdog timer *0x31: Use Watchdog timer
Template Type ⁽¹⁾	0x95	Template type	*0x30:proprietary 0x31: ISO 19794-2

Table 1 System parameter

(1) FM only

(2) FM only

(3) xx

(4) Reducing template size might affect authentication performance.

(5) Since firmware V1.3, the minimum security level s is changed to 1/10,000.

Timeout

Timeout period for user input. If users do not scan fingerprints or Wiegand input is not received during this period, TIMEOUT error will be returned.

Configuration

Code	Type	Valid Values	Default Value
0x62	Read/Write	0x30 : infinite 0x31 : 1 second ... 0x3A : 10 seconds ... 0x44 : 20 seconds	0x3A : 10 seconds

Compatibility

FM

Template Size

Template size can be specified between 256 and 384 bytes. Template size does not change the template capacity of a module and reducing it might affect the authentication performance.

Configuration

Code	Type	Valid Values	Default Value
0x64	Read/Write	Integer between 256 and 384. When the Encryption Mode(0x67) system parameter is on, it should be a multiple of 32.	384

Compatibility

FM

Enroll Mode

- One Time(0x30): Enrolls a fingerprint template with one scanned image.
- Two Times(0x31 and 0x32): Enrolls a fingerprint template with two scanned images. In this mode, the two fingerprint images are compared to each other. If two images do not match, they are rejected. If they match, the one with better quality will be enrolled. By enhancing the quality of enrolled templates, the authentication performance will be improved compared to the One Time mode. This mode is also classified into two sub modes according to the way in which the second scanning is initiated. In 0x31, the module will start the second scanning automatically. In 0x32, the second scanning will be started after the host sends another request.
- Two Templates(0x41 and 0x42): Enrolls two fingerprint templates. The enrollment process is identical to the Two Times mode. However, instead of selecting one of the two templates, the module will enroll both of them. By enrolling two templates for each user ID, the authentication performance will be improved further than the Two Times mode. This mode is also classified into two sub modes according to the way in which the second scanning is initiated.

See the ES section for examples of each mode.

Users are strongly advised to choose Two Templates mode. Not only does it show the best authentication performance, it is also adaptable to temporal changes of fingerprints. With Two Templates mode, the module will update one of the enrolled templates automatically when the scanned image has better quality. In this way, it can follow up the changes of fingerprints to a degree.

Configuration

Code	Type	Valid Values	Default Value
0x65	Read/Write	0x30 : 1 time 0x31 : 2 times (1 request command & 2 response commands)	. For FM templates(0x41) is

		<p>0x32 : 2 times II(2 request commands & 2 response commands)</p> <p>0x41 : 2 templates (1 request command & 2 response commands)</p> <p>0x42 : 2 templates II (2 request commands & 2 response commands)</p> <p>* 2 template modes(0x41 and 0x42) are only available for FM .</p>	default.
--	--	---	----------

Compatibility

FM

Security Level

Security level specifies FAR(False Acceptance Ratio). If it is set to 0x34(1/100,000), it means that the probability of accepting false fingerprints is 1/100,000. Since FAR and FRR(False Rejection Ration) is in inverse proportion to each other, FRR will increase with higher security levels.

These FAR values are for 1:1 matching. When identification commands are used for 1:N matching, the FAR would become higher. Therefore, we recommend that users set higher security level – lower FAR value – for 1:N matching, especially when more than hundreds of templates are stored on a module.

Since V1.6 firmware, automatic security levels have been added. With these settings, the module adjusts security level for identification automatically as the number of enrolled templates changes. The following table shows the relationships between the automatic security levels and the number of enrolled templates. For example, when the security level is Automatic Secure(0x51) and the number of enrolled templates is 500, the actual FAR for identification will be 1/10,000,000. The security level for verification is not changed.

Automatic Level	Verification (1:1)	Identification (1:N)			
		1 ~ 9	10 ~ 99	100 ~ 999	1000 ~
Normal(0x50)	1/10,000	1/10,000	1/100,000	1/1,000,000	1/10,000,000
Secure(0x51)	1/100,000	1/100,000	1/1,000,000	1/10,000,000	1/100,000,000
More Secure(0x52)	1/1,000,000	1/1,000,000	1/10,000,000	1/100,000,000	1/100,000,000

Configuration

Code	Type	Valid Values	Default Value
0x66	Read/Write	0x30 : 1/10 FAR(False Acceptance Rate) ⁽¹⁾ ⁽²⁾ 0x31 : 1/100 ⁽²⁾ 0x32 : 1/1,000 ⁽²⁾	0x50: Automatic Normal

		<p>0x33 : 1/10,000</p> <p>0x34 : 1/100,000</p> <p>0x35 : 1/1,000,000</p> <p>0x36 : 1/10,000,000⁽¹⁾</p> <p>0x37 : 1/100,000,000⁽¹⁾</p> <p>0x40 : 3/100⁽¹⁾⁽²⁾</p> <p>0x41 : 3/1,000⁽¹⁾⁽²⁾</p> <p>0x42 : 3/10,000⁽¹⁾⁽²⁾</p> <p>0x43 : 3/100,000⁽¹⁾</p> <p>0x44 : 3/1,000,000⁽¹⁾</p> <p>0x45 : 3/10,000,000⁽¹⁾</p> <p>0x46 : 3/100,000,000⁽¹⁾</p> <p>0x50 : Automatic Normal⁽¹⁾</p> <p>0x51 : Automatic Secure⁽¹⁾</p> <p>0x52 : Automatic More Secure⁽¹⁾</p> <p>(1) 0x30, 0x36 ~ 0x52 are only available for FM</p> <p>(2) Since firmware V1.3, the minimum security level of FM series is changed to 1/10,000(0x33). If users set a lower security level such as 1/100, it will be changed to 1/10,000 automatically.</p>	
--	--	--	--

Compatibility

FM

Encryption Mode

FM modules support 256 bit AES encryption algorithm for higher security.

When encryption mode is on, all the templates are transferred and saved in encrypted form. See the KW section for setting 256 bit encryption key.

Configuration

Code	Type	Valid Values	Default Value
0x67	Read/Write	0x30 : Encryption off 0x31 : Encryption on	0x30 : Encryption off

Compatibility

FM

Sensor Type

Specifies the type of sensor connected to the module. Refer to the Datasheet of each model for compatible sensors.

Configuration

Code	Type	Valid Values	Default Value
0x68	Read/Write	0x30 : Infineon FingerTip 0x31 : Atmel FingerChip 0x32 : AuthenTec FingerLoc AF-S2 0x33 : Optical fingerprint sensor 0x34 : STMicro TouchChip 0x35 : BMF BLP-100 0x36 : Second optical fingerprint sensor 0x37 : Fujitsu MBF310 0x38 : Third optical fingerprint sensor 0x39 : Fourth optical fingerprint sensor 0x3A : AuthenTec FingerLoc AFS8600	

Compatibility

FM

Image Format

Users can read fingerprint images using SI and RI commands. Image format parameter determines in which format a host receives the scanned image. If it is set to binary or 4 bit gray image, the module will down-convert 8 bit gray image to specified format before sending it. See the RI section for detailed descriptions of image format.

Regardless of this parameter, modules process only 8 bit gray images. Therefore, to enroll, identify, or verify fingerprint images, they should be always in 8 bit gray format.

Configuration

Code	Type	Valid Values	Default Value
0x6C	Read/Write	0x30 : Gray image 0x31 : Binary image 0x32 : 4 bit gray image	0x31 : Binary image

Compatibility

FM

Module ID

Specifies the ID of a module which is used in Network Packet Protocol. In Network Packet Protocol, modules respond to requests only if the terminal ID of a request packet is identical to its module ID.

Configuration

Code	Type	Valid Values	Default Value
0x6D	Read/Write	1 ~ 65535	

Compatibility

FM

Firmware Version

4 byte character which denotes the firmware version of a module. For example, if the version is V1.3A, 0x56313341 – ASCII codes for ‘V’, ‘1’, ‘3’, and ‘A’ – will be returned.

Since firmware V1.4, the first character represents the model number of the module. FM modules start with ‘B’.

Configuration

Code	Type	Valid Values	Default Value
0x6E	Read Only		

Compatibility

FM

Serial Number

4 byte integer which uniquely identifies each module.

Configuration

Code	Type	Valid Values	Default Value
0x6F	Read Only	4 byte integer	

Compatibility

FM

Baudrate

Baudrate setting for host interface. The module should be reset after saving system parameters.

Configuration

Code	Type	Valid Values	Default Value
0x71	Read/Write	0x31 : 9600 bps 0x32 : 19200 bps 0x33 : 38400 bps 0x34 : 57600 bps 0x35 : 115200 bps	0x35 : 115200 bps

Compatibility

FM

Baudrate2

FM series modules have two serial interfaces, host and auxiliary. This parameter specifies the baudrate setting of auxiliary interface.

Configuration

Code	Type	Valid Values	Default Value
0x72	Read/Write	0x31 : 9600 bps 0x32 : 19200 bps 0x33 : 38400 bps 0x34 : 57600 bps 0x35 : 115200 bps	0x35 : 115200 bps

Compatibility

FM

Enrolled Finger

The number of fingerprint templates enrolled in a module.

Configuration

Code	Type	Valid Values	Default Value
0x73	Read Only		

Compatibility

FM

Available Finger

The available number of fingerprint templates that can be used for future enrollment. The maximum number of available templates varies according to models. Refer to the Datasheet for template capacity of each model.

Configuration

Code	Type	Valid Values	Default Value
0x74	Read Only		

Compatibility

FM

Send Scan Success

With this parameter on, the module sends SCAN_SUCCESS message when it scans a fingerprint image or reads a Wiegand input successfully. It also sends SCAN_SUCCESS message when it receives a fingerprint template or a fingerprint image successfully in such commands as ET and EI. See the ES section for usage of SCAN_SUCCESS message.

Configuration

Code	Type	Valid Values	Default Value
0x75	Read/Write	0x30 : No SCAN_SUCCESS message 0x31 : Send SCAN_SUCCESS message	0x31 : Send SCAN_SUCCESS

Compatibility

FM

ASCII Packet

Determines the packet translation mode. If it is set to ASCII mode, the binary packet should be converted to ASCII format first before being sent to the module. Response packets are in ASCII format, too.

For example, if changing the baud rate of a module to 19200,

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x01	0x00	0x32	0x71	0xE4	0x0A

Byte transmission order in hexademical mode:

0x40, 0x01, 0x00, 0x00, 0x00, 0x00, 0x32, 0x00, 0x00, 0x00, 0x71, 0xE4, 0x0A

Byte transmission order in ASCII mode:

0x34, 0x30, 0x30, 0x31, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x30, 0x33, 0x32,

0x30 0x30, 0x30, 0x30, 0x30, 0x30, 0x37, 0x31, 0x45, 0x34, 0x30, 0x41

Configuration

Code	Type	Valid Values	Default Value
0x76	Read/Write	0x30 : Hexademical packet 0x31 : ASCII packet	0x30 : Hexademical packet

Compatibility

FM

Rotate Image

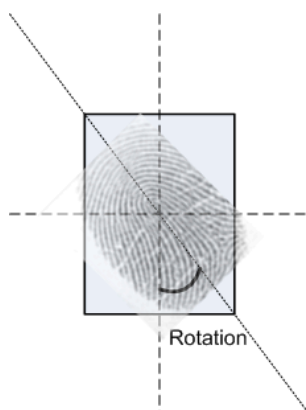
With this parameter on, the module rotates a fingerprint image upside down before processing it. This parameter would be useful when the sensor should be installed upside down.

Configuration

Code	Type	Valid Values	Default Value
0x77	Read/Write	0x30 : Upright image 0x31 : Upside down image	0x30 : Upright image

Rotation

Specifies the maximum rotation for fingerprint input. If rotation is beyond this limit, matching will fail even if two templates are identical. This parameter also affects matching speed. The wider the maximum rotation is, the slower matching speed is.



Configuration

Code	Type	Valid Values	Default Value
0x78	Read/Write	0x31: 15 degree 0x32: 30 degree 0x33: 45 degree 0x34: 60 degree 0x35: 75 degree 0x36: 90 degree	0x36: 90 degree (*For Atmel's FingerChip sensor, 0x33 is default)

Sensitivity

Specifies sensor sensitivity to detect a finger. On high sensitivity, the module will accept the finger input more easily. While, by decreasing the sensitivity, the input fingerprint image will be more stabilized. In case of optical models, sensitivity to sunlight is also alleviated by decreasing sensitivity parameter.

In FM modules, this parameter can be used to turn on the sensor heating mode. When the parameter is set to 0x30, the FC sensor will be heated before sensing fingerprint inputs. In general, the image quality would be better with sensor heating mode. However, this mode consumes more power and users should wait for a few seconds before the sensor gets heated.

Configuration

Code	Type	Valid Values	Default Value
0x80	Read/Write	0x30 : Least sensitive ... 0x37 : Most sensitive	0x37 : Most sensitive

Image Quality

When a fingerprint is scanned, the module will check if the quality of the image is adequate for further processing. If it is poor, the module will send TRY_AGAIN error message. Image quality parameter specifies the strictness of this quality check.

Configuration

Code	Type	Valid Values	Default Value
0x81	Read/Write	0x30 : Weak qualification 0x31 : Moderate qualification 0x32 : Strong qualification 0x33 : Strongest qualification	0x31 : Moderate qualification

Auto Response

If the module processes GPIO inputs or Wiegand inputs, or works in Free Scan mode, response packets are not sent to serial interface by default. By setting this parameter, a host can receive response packets in those cases, too.

Configuration

Code	Type	Valid Values	Default Value
0x82	Read/Write	0x30 : No response command 0x31 : Send response command (host) 0x32 : Send response command (aux) 0x33 : Send response command (both)	0x30 : No response command (*0x32 and 0x33 are only available in FM)

Network Mode

FM modules provide functionality to support RS422/485 network interface.

- Not for FM Module !!: To integrate modules into RS422/485 network, the following requirements should be met.
 - External level conversion circuitry between serial port of the module and RS422/485 line. Refer to Application Notes: RS422/RS485 Interface for reference circuitry.
 - Network mode parameter should be set to 0x31. In this case, GPIO7 is automatically allocated as flow control output for external circuitry.
- FM Module: External level conversion circuitry is not needed and flow control of output signal is handled by firmware itself. Operation mode – full or half duplex – is determined by network mode parameter.

Network mode system parameter also determines which packet protocols are used.

Network Mode	Supported Packet Protocol		
	13 byte Packet Protocol	15 byte Network Packet Protocol	15 byte Broadcast Packet Protocol
Single(0x30)	O	O	O
Network(0x30/0x31)	X	O	O

Configuration

Code	Type	Valid Values		Default Value
0x8	Read/Write F		FM	0x30 : Single mode
			0x30 : Single mode 0x31 : Half duplex 0x32 : Full duplex	

Free Scan

In normal mode, users have to send a command through serial interface or GPIO to initiate identification process. With Free Scan mode on, however, the module starts identification process automatically on sensing fingerprint input. To receive response packets through serial interface in Free Scan mode, Auto Response parameter should be on. Other commands are handled normally even if Free Scan mode is on.

Configuration

Code	Type	Valid Values	Default Value
0x84	Read/Write	0x30 : Normal mode 0x31 : Free scan mode	0x30 : Normal mode

Provisional Enroll

Determines if enrolled templates are saved permanently into flash memory or temporarily into DRAM. With provisional enroll, enrolled templates on DRAM will be erased if the module is turned off. To move templates into flash memory, FP(0x23) command should be executed. DP(0x24) command will erase the provisional templates on DRAM.

Configuration

Code	Type	Valid Values	Default Value
0x85	Read/Write	0x30 : Permanent enrollment 0x31 : Provisional enrollment	0x30 : Permanent enrollment

Pass When Empty

Determines if identification succeeds or fails when there is no enrolled template in a module.

Configuration

Code	Type	Valid Values	Default Value
0x86	Read/Write	0x30 : Fail when DB is empty 0x31 : Pass when DB is empty	0x30 : Fail when DB is empty

Response Delay

Specifies the delay time for which the module should be waiting before sending a response packet. Response delay can be useful when packets are lost due to slow communication channel.

Configuration

Code	Type	Valid Values	Default Value
0x87	Read/Write	0x30 : No delay 0x31 : 20 msec ... 0x35 : 100 msec ... 0x3A : 200 msec	0x30 : No delay

Matching Timeout

Timeout period for 1:N matching. If identification process is not finished until this period, MATCH_TIMEOUT error will be returned.

Configuration

Code	Type	Valid Values	Default Value
0x88	Read/Write	0x30 : infinite 0x31 : 1 second ... 0x3A : 10 seconds ... 0x44 : 20 seconds	0x30 : infinite

Build Number

4 byte character which denotes the build date of the firmware. For example, if the build date is 2005 April 28, 0x05042800 will be returned.

Configuration

Code	Type	Valid Values	Default Value
0x89	Read Only		

Enroll Displacement

Two times and two templates enroll modes are provided to enhance authentication performance. However, if users scan two identical fingerprint images, the purpose of these modes becomes pointless. To prevent these cases, enroll displacement constraint can be imposed. If displacement of two fingerprint images is smaller than this constraint, enroll will fail.

Configuration

Code	Type	Valid Values	Default Value
0x8A	Read/Write	0x30 : No check 0x31 : Above 1 mm away ... 0x35 : Above 5 mm away ... 0x3A: Above 10 mm away	0x30 : No check

Lighting Condition

Optical sensors are sensitive to lighting condition. With this parameter, users can tune optical sensors to be adapted for their lighting environment. For example, if a module is installed in outdoor, this parameter should be set to 0x30.

Configuration

Code	Type	Valid Values	Default Value
0x90	Read/Write	0x30 : Outdoor 0x31 : Indoor	0x30 : Outdoor (* Until V1.4 firmware, the default had been Indoor)

Free Scan Delay

With Free Scan mode on, the module starts identification process automatically on sensing fingerprint input. Using Free Scan Delay parameter, users can specify delay between consecutive identification processes in Free Scan mode.

Configuration

Code	Type	Valid Values	Default Value
0x91	Read/Write	0x30 : No delay 0x31 : 1 second ... 0x40 : 10 seconds	0x31 : 1 second (* Until V1.4 firmware, the default had been No delay.)

Fast Mode

When more than hundreds of templates are stored in a module, the matching time for identification can be very long. Fast Mode parameter can be used to shorten the 1:N matching time with little degradation of authentication performance. The security level - FAR - is not affected by this parameter, but the FRR can be a bit higher than normal mode. In typical cases, Fast Mode 1 is 2 ~ 3 times faster than Normal mode. And Fast Mode 5 is 6 ~ 7 times faster than Normal mode.

If it is set to Automatic(0x36), the fast mode will be adjusted automatically as follows.

Enrolled Templates	Fast Mode
1 ~ 99	Normal
100 ~ 499	Fast Mode 1
500 ~ 999	Fast Mode 2
1000 ~ 1999	Fast Mode 3
2000 ~ 3999	Fast Mode 4
4000 ~	Fast Mode 5

Configuration

Code	Type	Valid Values	Default Value
0x93	Read/Write	0x30 : Normal 0x31 : Fast Mode 1 ... 0x35 : Fast Mode 5(Fastest) 0x36 : Automatic	0x36 : Automatic

Watchdog

A watchdog timer is a hardware timing device that triggers a system reset if the main program, due to some fault condition, such as a hang, neglects to regularly service the watchdog. The purpose of watchdog timer is to bring the system back from the hung state into normal operation. It is most useful for mission critical systems that must remain in continuous operation without human intervention.

Configuration

Code	Type	Valid Values	Default Value
0x94	Read/Write	0x30 : Don't use Watchdog timer 0x31 : Use Watchdog timer	0x31 : Use Watchdog timer

Template Type

Since firmware V1.7, FM modules support the standard template format defined in ISO 19794-2. Two template formats – proprietary one and ISO 19794-2 – are not compatible. Therefore, you can change this parameter only if there is no enrolled template.

Configuration

Code	Type	Valid Values	Default Value
0x96	Read/Write	0x30 : proprietary 0x31 : ISO 19794-2	*0x30 : proprietary

Command Summary

Category	Name	Code	Description	1000/ 2000 series	3000 series	F M series
System Configuration	SW	0x01	Write system parameter	O	O	O
	SF	0x02	Save system parameter	O	O	O
	SR	0x03	Read system parameter	O	O	O
	CS	0x1A	Calibrate sensor	X	O	O
	SS	0x04	Check system status	O	O	O
	CA	0x60	Cancel	X	O	O
	ID	0x85	Get the module ID	X	O	O
	UG	0x62	Upgrade firmware	X	O	O
	RS	0xD0	Reset the module	X	O	O
	LM	0xB1	Lock the module	X	X	O
	UM	0xB0	Unlock the module	X	X	O
	MP	0xB2	Change the master password	X	X	O
Enroll	ES	0x05	Enroll by scan	O	O	O
	ESA	0x70	ES with administrator's verification	X	O	O
	EI	0x06	Enroll by image	O	O	O
	EIX	0x80	EI with extended data transfer protocol	X	O	O
	ET	0x07	Enroll by template	O	O	O
	ETX	0x87	ET with extended data transfer protocol	X	O	O
	EW	0x1C	Enroll by Wiegand ID	X	X	O
	EWA	0x71	EW with administrator's verification	X	X	O
Verify	VS	0x08	Verify by scan	O	O	O
	VI	0x09	Verify by image	O	O	O
	VIX	0x82	VI with extended data transfer protocol	X	O	O
	VT	0x10	Verify by template	O	O	O
	VW	0x1D	Verify by Wiegand ID	X	X	O
	VH	0x22	Verify host template by scan	O	O	O

	WSL	0x6B	Write security level of a user	X	O	O
	RSL	0x6C	Read security level of a user	X	O	O
Identify	IS	0x11	Identify by scan	O	O	O
	II	0x12	Identify by image	O	O	O
	IIX	0x81	II with extended data transfer protocol	X	O	O
	IT	0x13	Identify by template	O	O	O
Delete	DA	0x17	Delete all templates	O	O	O
	DAA	0x74	DA with administrator's verification	X	O	O
	DT	0x16	Delete template	O	O	O
	DS	0x1E	Delete by scan	X	O	O
	DSA	0x72	DS with administrator's verification	X	O	O
	DW	0x1F	Delete by Wiegand ID	X	X	O
	DWA	0x73	DW with administrator's verification	X	X	O
Template management	LT	0x18	List user ID	O	O	O
	LTX	0x86	List user ID with extended data transfer protocol	X	O	O
	CT	0x19	Check user ID	O	O	O
	FP	0x23	Fix all provisional templates	O	O	O
	DP	0x24	Delete all provisional templates	O	O	O
Get image and template	RI	0x20	Read image	O	O	O
	RIX	0x84	RI with extended data transfer protocol	X	O	O
	SI	0x15	Scan image	O	O	O
	SIX	0x83	SI with extended data transfer protocol	X	O	O
	RT	0x14	Read template	O	O	O
	RTX	0x89	RT with extended data transfer protocol	X	O	O
	ST	0x21	Scan template	O	O	O
	KS	0x35	Scan template with challenge data	O	O	O
	KW	0x34	Write encryption key	O	O	O
User memory management	ML	0x31	Get size of user memory	X	O	O
	MW	0x32	Write to user memory	X	O	O

	MR	0x33	Read from user memory	X	O	O
Time and log management	TW	0x3A	Write current time	X	X	O
	TR	0x3B	Read current time	X	X	O
	LN	0x3C	Get number of log data	X	X	O
	LR	0x3D	Read log data	X	X	O
	LD	0x3E	Delete log data	X	X	O
	LC	0x3F	Set/get the custom log field	X	X	O
	RCL	0xEC	Read log cache	X	X	O
	CCL	0xEB	Clear log cache	X	X	O
Wiegand configuration	WW	0x41	Write Wiegand configuration	X	X	O
	WR	0x42	Read Wiegand configuration	X	X	O
	WG	0x43	Get Wiegand input	X	X	O
	WS	0x44	Set Wiegand output	X	X	O
	WM	0x68	Map Wiegand ID to input function	X	X	O
	WL	0x69	List Wiegand ID mapping	X	X	O
	WC	0x6A	Clear Wiegand ID mapping	X	X	O
Extended Wiegand configuration	WWX	0xC0	Write Wiegand configuration	X	X	O
	WRX	0xC1	Read Wiegand configuration	X	X	O
	WGX	0xC2	Get Wiegand input	X	X	O
	WSX	0xC3	Set Wiegand output	X	X	O
	WFW	0xC4	Set alternative value of a field	X	X	O
	WFR	0xC5	Get alternative value of a field	X	X	O
	WPW	0xC6	Write Wiegand I/O settings	X	X	O
	WPR	0xC7	Read Wiegand I/O settings	X	X	O
Input configuration	IW	0x47	Write input configuration	X	X	O
	IR	0x48	Read input configuration	X	X	O
	IG	0x49	Get input state	X	X	O
Output configuration	OW	0x4A	Write output configuration	X	X	O
	OR	0x4B	Read output configuration	X	X	O
	OL	0x4C	Read output configuration list	X	X	O

	OS	0x4D	Set output state	X	X	O
GPIO configuration	GW	0x37	Write GPIO configuration	O	O	X
	GR	0x36	Read GPIO configuration	O	O	X
	GC	0x38	Clear GPIO configuration	O	O	X
	GD	0x39	Set default GPIO configuration	O	O	X
Admin level configuration	AW	0x65	Write administration level	X	O	O
	AR	0x66	Read administration level	X	O	O
	AC	0x67	Clear administration level	X	O	O
Auth. mode configuration	UW	0xA3	Write authentication mode	X	X	O
	UR	0xA4	Read authentication mode	X	X	O
	UC	0xA5	Reset authentication mode to default	X	X	O
	UL	0xA6	List user IDs grouped by authentication mode	X	X	O
Blacklist configuration	ABL	0xF3	Add an ID to blacklist	X	X	O
	DBL	0xF4	Delete an ID from blacklist	X	X	O
	RBL	0xF5	Read blacklist	X	X	O
	CBL	0xF6	Clear blacklist	X	X	O
Entrance limit configuration	WME	0xF0	Write entrance limit	X	X	O
	RME	0xF1	Read entrance limit	X	X	O
	CME	0xF2	Clear entrance limit	X	X	O

Table 2 Command

Flag / Error Code

Name	Code	Description
CHECK_ID	0x70	Check if the requested user ID exists.
ADD_NEW	0x71	Adding more fingerprints to a current existing user ID.
CONTINUE	0x74	There is more data to be sent.
AUTO_ID	0x79	Automatically assign user ID in enrollment.
CHECK_FINGER ⁽¹⁾	0x84	Check if the finger is already enrolled.
CHECK_FINGER_AUTO_ID ⁽¹⁾	0x85	Check if the finger is already enrolled. If not, assign user ID automatically.
ADD_DURESS ⁽²⁾	0x92	Add duress fingerprints to an existing user ID.

Table 3 Flag code

(1) These flags are available for FM only.

(2) These flags are available for FM only.

Name	Code	Description
SUCCESS	0x61	Successfully completed.
SCAN_SUCCESS	0x62	Fingerprint input has succeeded.
SCAN_FAIL	0x63	Sensor or fingerprint input has failed.
NOT_FOUND	0x69	There is no requested data found.
NOT_MATCH	0x6A	Fingerprint does not match.
TRY_AGAIN	0x6B	Fingerprint image is not good.
TIME_OUT	0x6C	Timeout for fingerprint input.
MEM_FULL	0x6D	Maximum template capacity exceeded.
EXIST_ID	0x6E	The requested user ID exists.
FINGER_LIMIT	0x72	The number of fingerprints enrolled in same ID exceeds its limit (10).
CONTINUE	0x74	There is more data to be sent.
UNSUPPORTED	0x75	The command is not supported.
INVALID_ID	0x76	The requested user ID is invalid or missing.

TIMEOUT_MATCH	0x7A	Timeout for matching in identification.
BUSY ⁽¹⁾	0x80	Module is processing another command.
CANCELED ⁽¹⁾	0x81	The command is canceled.
DATA_ERROR ⁽¹⁾	0x82	The checksum of a data packet is incorrect.
DATA_OK ⁽¹⁾	0x83	The checksum of a data packet is correct.
EXIST_FINGER ⁽¹⁾	0x86	The finger is already enrolled.
DURESS_FINGER ⁽²⁾	0x91	A duress finger is detected.
LOCKED ⁽²⁾	0xA1	Module is locked.
REJECTED_ID ⁽²⁾	0x90	Authentication mode of the user is AUTH_REJECT or the ID is in the blacklist.
ENTRANCE_LIMIT ⁽²⁾	0x94	Authentication fails since the entrance limit is exceeded.

Table 4 Error code

(1) These error codes are available for FM only.

(2) These error codes are available for FM only.

SW : System Parameter Write

Changes the value of system parameters. Since the changed value will be deleted if the module is reset, save the changed value in the flash memory of the module when it is necessary. This can be done by SF command (System parameter save) to be presented next. To validate the modification of SENSOR_INDEX and BAUDRATE, the module should be restarted.

Request Command

Field	Data	Description
Start code	0x40	
Command	0x01	
Param	NULL	
Size	Parameter value	See System Parameter Table
Flag	Parameter ID	See System Parameter Table
Checksum	#	
End code	0x0A	

Response Command

Field	Data	Description
Start code	0x40	
Command	0x01	
Param	Parameter ID	See System Parameter Table
Size	NULL	
Error	0x61 0x69 0x80	SUCCESS NOT_FOUND BUSY
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Parameter setup is successfully completed.
NOT_FOUND	There is no requested parameter ID found.
BUSY	Module is processing another command.

Example

If the communication baud rate of the module is changed to 19200,

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x01	0x00	0x32	0x71	0xE4	0x0A

the actual values input to the size are as follows:

1 st place	2 nd place	3 rd place	4 th place
0x32	0x00	0x00	0x00

Byte transmission order:

0x40, 0x01, 0x00, 0x00, 0x00, 0x00, 0x32, 0x00, 0x00, 0x00, 0x71, 0xE4, 0x0A

SF : System Parameter Save

Stores the system parameter data of the module in the flash memory. SF command is used to store the parameter data in the flash memory so that it can be sustained even when the module is reset. This command should be strictly used only for factory-default setting. That is, it should not be used on a regular basis.

Request command

Field	Data	Description
Start code	0x40	
Command	0x02	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x02	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
	0x80	BUSY
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Parameter save is successfully completed.
BUSY	Module is processing another command.

SR : System Parameter Read

Reads the system parameter values that correspond to the ID in the flag field of the request command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x03	
Param	NULL	
Size	NULL	
Flag	Parameter ID	See System Parameter Table
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x03	
Param	Parameter ID	See System Parameter Table
Size	Parameter value	See System Parameter Table
Error	0x61	SUCCESS
	0x69	NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	The value of the requested parameter ID has been successfully read.
NOT_FOUND	There is no requested parameter ID found.

CS : Calibrate Sensor

Calibrate fingerprint sensor. All type of sensors should not be necessarily calibrated. This command is supported for AuthenTec's FingerLoc AF-S2 and STMicro's TouchChip. After using the CS command, the SF command should be used to save calibration result in flash memory.

Request command

Field	Data	Description
Start code	0x40	
Command	0x1A	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x1A	
Param	NULL	
Size	NULL	
Error	0x61 0x63 0x75 0x80	SUCCESS SCAN_FAIL UNSUPPORTED BUSY
Checksum	#	

End code	0x0A	
----------	------	--

Error code

Error code	Description
SUCCESS	Calibration of sensor is successfully completed.
SCAN_FAIL	Calibration of sensor failed.
UNSUPPORTED	Calibration of sensor is not needed.
BUSY	Module is processing another command.

SS : System Status Check

Checks the current status of the module.

Request command

Field	Data	Description
Start code	0x40	
Command	0x04	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x04	
Param	Status code	Current status of the module
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Status code

Status	Data	Description
ALIVE	0x30	Module is in normal condition

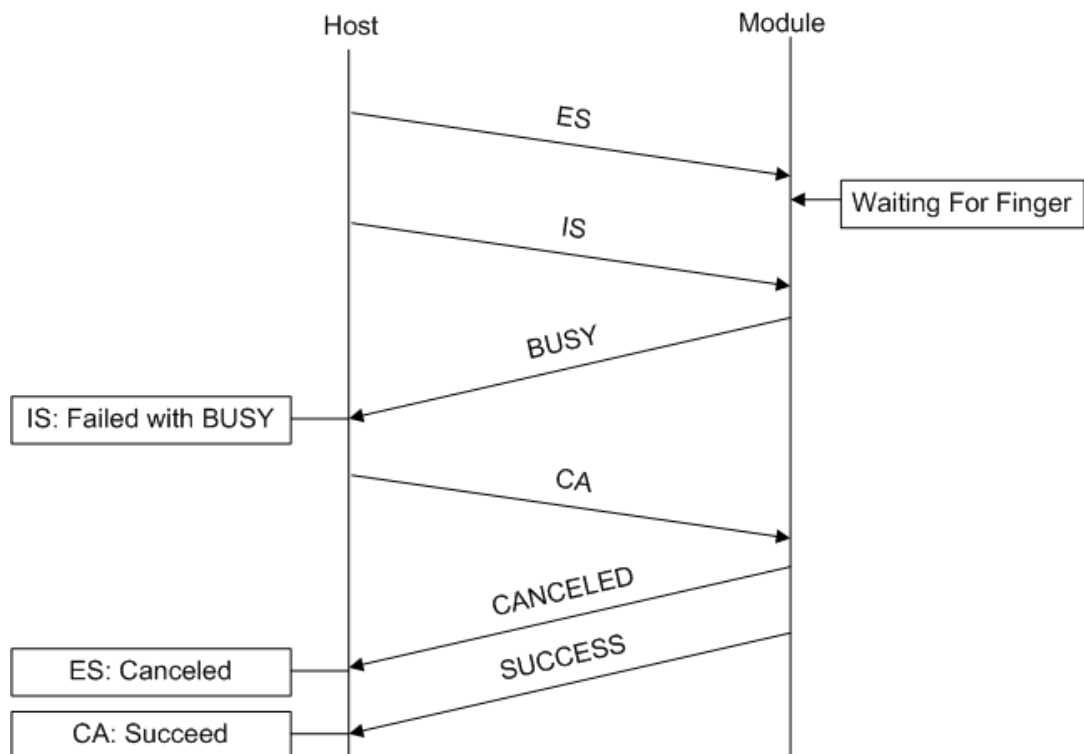
BUSY	0x34	Module is processing a command
------	------	--------------------------------

Error code

Error code	Description
SUCCESS	System status check is successfully completed.

CA : Cancel

Cancels the command which is being processed by the module. When the module is executing a command which takes long time to complete or needs user input to proceed, the status of the module will be changed into BUSY. In that case, only SR and SS commands can be processed. All the other commands will fail and BUSY error code will be returned. If users want to execute another command before finishing the current one, they can explicitly cancel it by CA command. Cancelable commands are as follows: ES, ESA, EW, EWA, VS, VW, IS, DS, DSA, DW, DWA, DAA, SI, ST, KS. The following diagram shows one example of CA command. In this example, IS was sent to the module before ES was finished, and failed with BUSY error code. After that, host canceled ES by CA command.



Request command

Field	Data	Description
Start code	0x40	
Command	0x60	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x60	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Cancel is successfully completed.

ID : Get Module ID

In a network environment, a host needs to know which modules participate in the network. ID is a special command for this purpose. A host sends ID command to all the modules using Broadcast Packet Protocol. Each module returns 4 byte response packet including its module ID. Since packet collision can occur in network environments, the host should repeat this process until no module returns response packets.

The exact procedure is as follows:

1. Host sends ID command and a list of user IDs which were received in previous ID requests. It also specifies maximum delay parameter to minimize the possibility of packet collisions.
2. Each module checks the user ID lists and returns its user ID if it is not included in the list. Before sending a response packet, each module will wait for a random period less than the maximum delay parameter.
3. Host receives response packets and adds new IDs to the list.
4. Repeat 1 ~ 3 until no response packet is received.

Request command

Field	Data	Description
Start code	0x41	
Command	0x85	
Param	Data Size	Number of received user IDs * 2. It would be 0 in the first request.
Size	Maximum Delay in milliseconds	Before sending a response packet, each module will wait for a random period less than this maximum value.
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then the user IDs which were received in previous requests, and finally the 0x0A.

Response command

Data	Description
Start code	1 byte(0x41)
Module ID	2 byte
Checksum	1 byte: checksum of preceding 3 bytes

S

Example

There are three modules with ID 1, 2, and 3 respectively. The host sends ID command.

Start	Terminal ID	Command	Data Size	Maximum Delay
0x41	0x00	0x85	0x00	0x3E8

Byte transmission order:

0x41, 0x00, 0x00, 0x85, 0x00, 0x00, 0x00, 0x00, 0xE8, 0x03, 0x00, 0x00, 0x00, 0xB1, 0x0A

Module 1 returns.

0x41, 0x01, 0x00, 0x42

Module 2 returns.

0x41, 0x02, 0x00, 0x43

Module 3 returns.

0x41, 0x03, 0x00, 0x44

The host receives the response packets of Module 1 and Module 2. The response packet of Module 3 is lost due to packet collision. The host sends ID command again with ID lists.

Start	Terminal ID	Command	Data Size	Maximum Delay
0x41	0x00	0x85	0x04	0x3E8

Byte transmission order:

0x41, 0x00, 0x00, 0x85, 0x04, 0x00, 0x00, 0x00, 0xE8, 0x03, 0x00, 0x00, 0x00, 0xB5, 0x0A

0x01, 0x00, 0x02, 0x00, 0x0A

Module 3 returns.

0x41, 0x03, 0x00, 0x44

UG : Upgrade Firmware

Upgrade firmware using Extended Data Transfer Protocol. If upgrade is interrupted by power loss or communication failure, the module will be in unusable state. Therefore, users should use this command with utmost caution.

Request command

Field	Data	Description
Start code	0x40	
Command	0x62	
Param	NULL	
Size	Size of firmware	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x62	
Param	NULL	
Size	NULL	
Error	0x61 0x6D 0x82	SUCCESS MEM_FULL DATA_ERROR
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Firmware upgrade is successfully completed.
MEM_FULL	Out of memory.
DATA_ERROR	The checksum of a data packet is incorrect.

Example

If the firmware size is 200KB and the data packet size is 16KB, the following packets are transferred between the host and the module.

(1) Request Packet

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x62	0x00	0x32000	0x00	0xC5	0x0A

(2) Response Packet

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x62	0x00	0x00	0x61	0x03	0x0A

(3) Data Packets

Start	Command	Num of Packet	Packet Index	Data Size	Flag	Checksum	End
0x40	0x62	0x0D	0x00	0x4000	0x00	0xEF	0x0A
0x40	0x62	0x0D	0x01	0x4000	0x00	0xF0	0x0A
...							
0x40	0x62	0x0D	0x0C	0x2000	0x00	0xDB	0x0A

These data packet headers are followed by 16KB data and 4 byte checksum. If the error code of the acknowledge packet is not DATA_OK(0x83), the host should stop the

upgrade process and start it again.

RS : Reset Module

Resets the module. There are two ways to reset the module. One is to use RS command and the other is to use GPIO input function. See GW and IW commands for the latter.

Request command

Field	Data	Description
Start code	0x40	
Command	0xD0	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xD0	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The module is reset successfully.

LM : Lock Module

Locks the module. When the module is locked, it returns LOCKED(0xA1) error code. Locking should be used with utmost caution. If users forget the master password after locking a module, it cannot be unlocked.

Request command

Field	Data	Description
Start code	0x40	
Command	0xB1	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xB1	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	The module is locked successfully.
---------	------------------------------------

UM : Unlock Module

Unlocks the module. To unlock, a 16 byte master password should be sent to the module. The default password is a string of 16 NULL characters.

Request command

Field	Data	Description
Start code	0x40	
Command	0xB0	
Param	NULL	
Size	Data size	16 byte password + 2 byte checksum = 18
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then 16 byte password, 2 byte checksum of the password, and finally the 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0xB0	
Param	NULL	
Size	NULL	
Error	0x61 0x6A 0x6B	SUCCESS NOT_MATCH TRY_AGAIN
Checksum	#	

End code	0x0A	
----------	------	--

Error code

Error code	Description
SUCCESS	The module is unlocked successfully.
NOT_MATCH	The master password is not correct.
TRY_AGAIN	The checksum of the password is not correct.

MP : Set Master Password

Changes the master password. The password is required for unlocking a locked module. The default password is a string of 16 NULL characters.

Request command

Field	Data	Description
Start code	0x40	
Command	0xB2	
Param	NULL	
Size	Data size	16 byte old password + 2 byte checksum + 16 byte new password + 2 byte checksum = 36
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then 16 byte old password, 2 byte checksum of the old password, 16 byte new password, 2 byte checksum of the new password, and finally the 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0xB2	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
	0x6A	NOT_MATCH

	0x6B	TRY_AGAIN
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The password is changed correctly.
NOT_MATCH	The old password is not correct.
TRY_AGAIN	The checksum of the password is not correct.

ES : Enroll by Scan

Enrolls a user's fingerprint.

There are three modes of fingerprint enrollment. ENROLL_MODE in system parameters determines which enroll mode is selected. In any modes, the size of fingerprint templates is same.

(1) Enroll Mode

- One Time(0x30): Enrolls a fingerprint template with one scanned image.
- Two Times(0x31 and 0x32): Enrolls a fingerprint template with two scanned images. In this mode, the two fingerprint images are compared to each other. If two images do not match, they are rejected. If they match, the one with better quality will be enrolled. By enhancing the quality of enrolled templates, the authentication performance will be improved compared to the One Time mode. This mode is also classified into two sub modes according to the way in which the second scanning is initiated. In 0x31, the module will start the second scanning automatically. In 0x32, the second scanning will be started after the host sends another request.
- Two Templates(0x41 and 0x42): Enrolls two fingerprint templates. The enrollment process is identical to the Two Times mode. However, instead of selecting one of the two templates, the module will enroll both of them. By enrolling two templates for each user ID, the authentication performance will be improved further than the Two Times mode. In this mode, one of the two stored templates may be automatically updated to reflect dynamical changes of user's finger skin. Each time when a user is verified, module decides if it will replace the existing template with the newly obtained one. For network applications in which a central server manages templates, special attention must be paid. In that case, automatic changes of a template may cause a synchronization problem with the server. This mode is also classified into two sub modes according to the way in which the second scanning is initiated. 0x41 is the default mode for FM .

(2) Enroll Options

Users can specify one of six options for fine tuning the enrollment process. If users do not specify any option – the FLAG field of the packet is NULL, the module does not care about duplication of a user ID and always creates or overwrites with new templates. The user ID “0x0000” is not allowed, since it is pre-assigned for internal use.

- **ADD_NEW:** Adds another fingerprint to the same user ID. The maximum number of templates per user is 10. By enrolling more templates, users can expect better authentication performance. FINGER_LIMIT error will be returned if the number exceeds the limit.
- **CHECK_ID:** Before enrolling, checks if the user ID has already some templates. If it does, EXIST_ID will be returned. This option is useful when users do not want to overwrite existing templates.
- **CHECK_FINGER:** Before enrolling, checks if the same fingerprint is already enrolled. If the identification succeeds, return EXIST_FINGER error. If the identification fails, continue enroll process with ADD_NEW option.
- **AUTO_ID:** The user ID will be assigned automatically by the module.
- **CHECK_FINGER_AUTO_ID:** Before enrolling, checks if the same fingerprint is already enrolled. If the identification succeeds, return EXIST_FINGER error. If the identification fails, continue enroll process with AUTO_ID option.
- **ADD_DURESS:** Adds another fingerprint as duress one to the specified user ID. Under duress, users can authenticate with duress finger to notify the threat. When duress finger is matched, the module will return DURESS_FINGER error code and write a log. Users can also setup output signals for duress events. When enrolling, the duress finger should not match with non-duress fingerprints of the same ID. If it is the case, EXIST_FINGER error code will be returned.

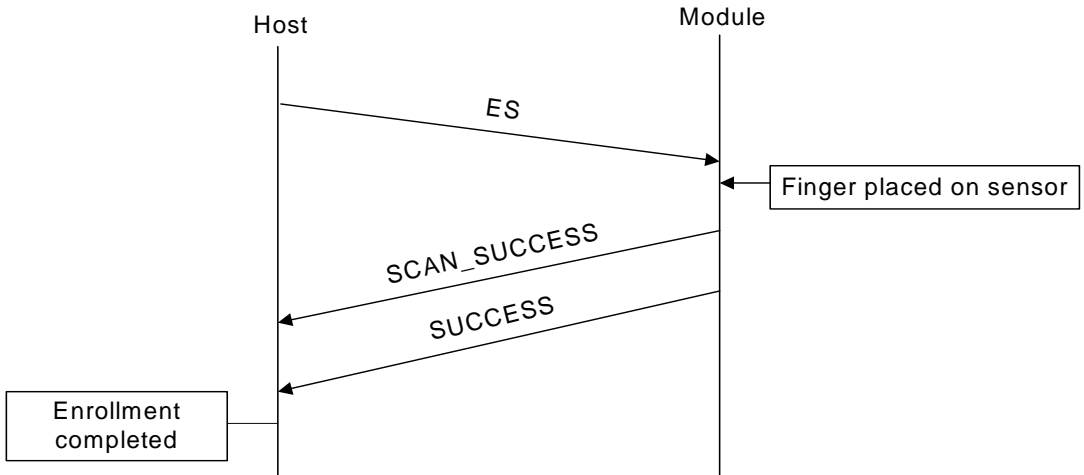
Commands like enroll, verify and identify may receive two response commands. The

first command comes in the intermediate stage, notifying image scanning completed and the other command notifies process result. If you do not want to receive the intermediate command, you can disable it by writing '0x30' to the system parameter SEND_SCAN_SUCCESS.

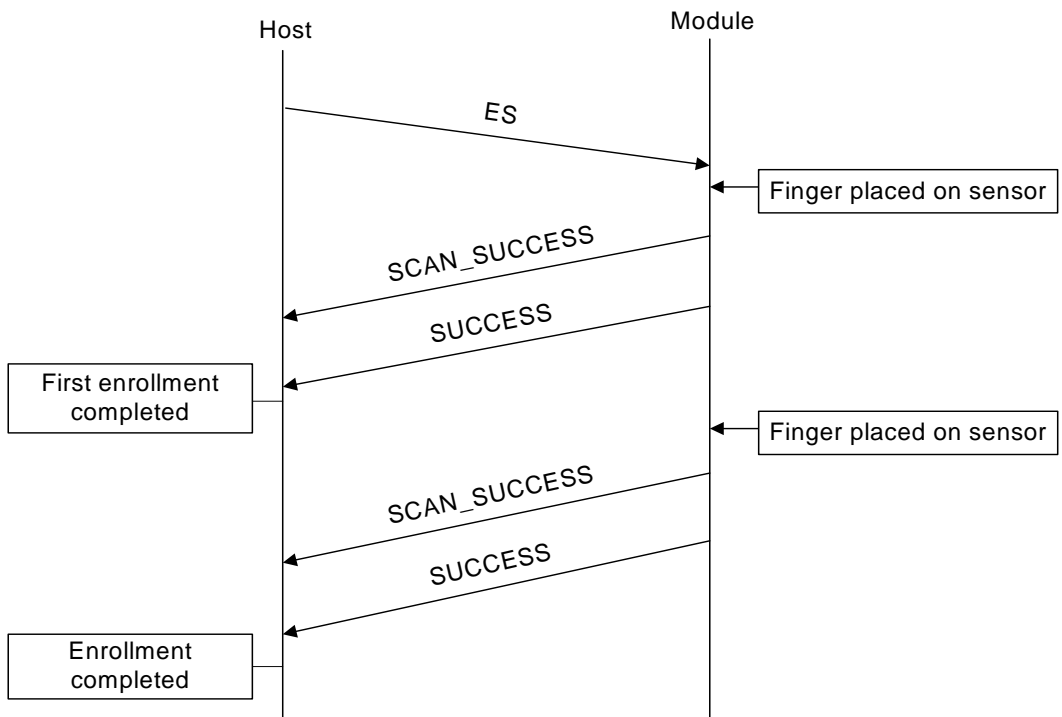
In the process of enrollment, if a user does not put his/her finger on the sensor, the module will send a timeout message and ends the enrollment process.

Timelines of ES:

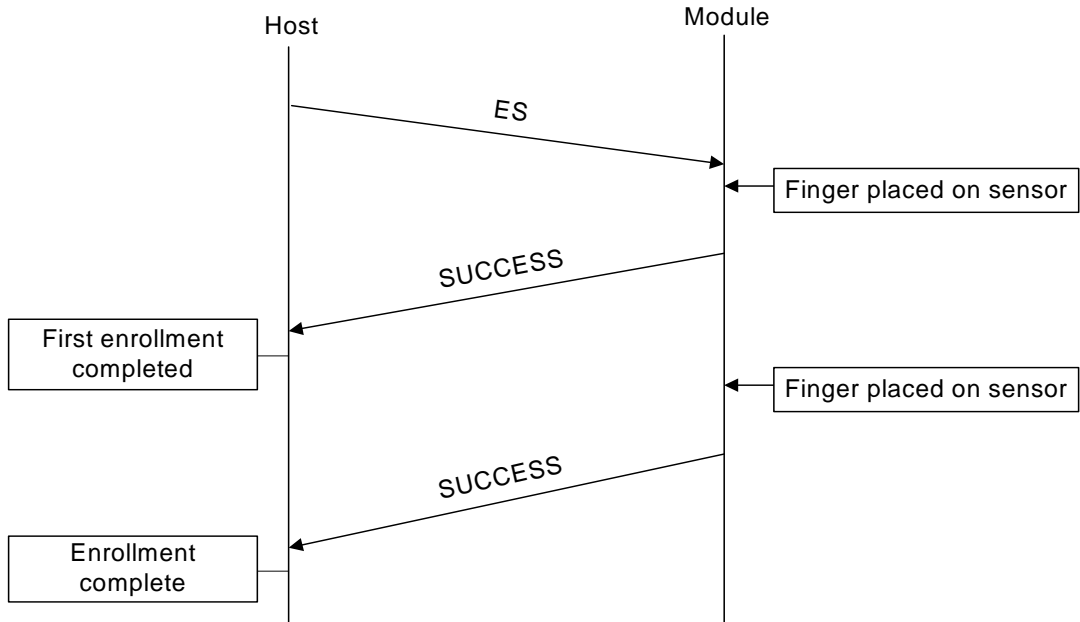
1. ENROLL_MODE = 0x30, SEND_SCAN_SUCCESS = 0x31



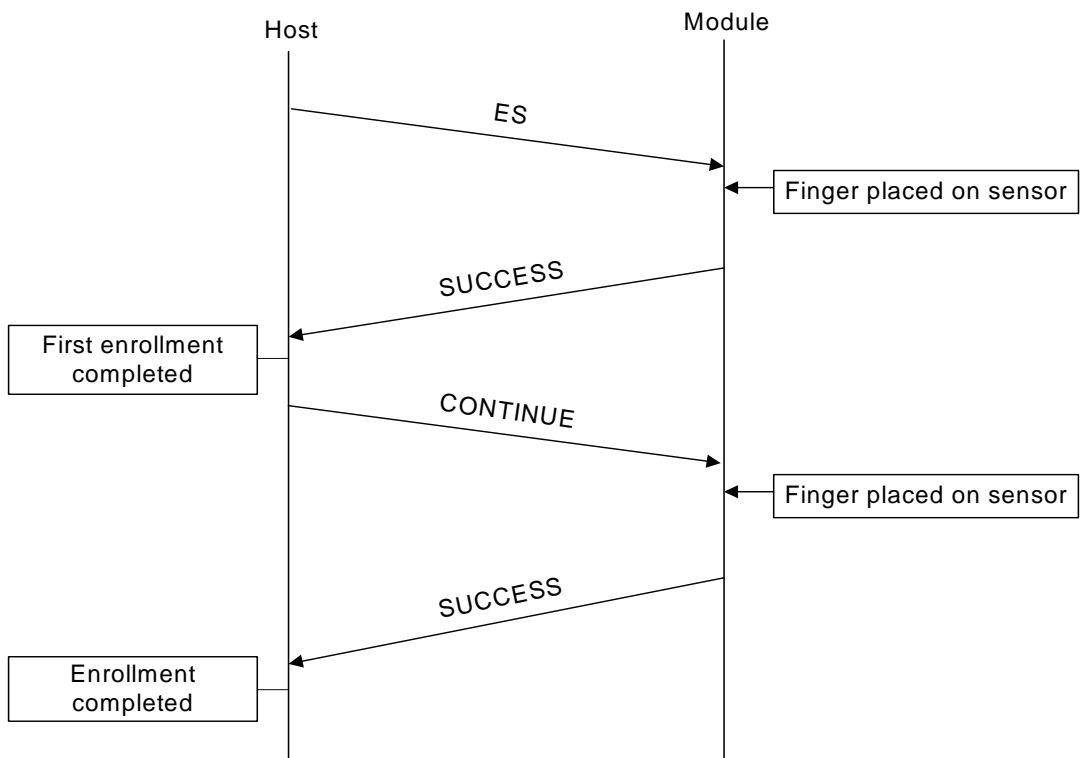
2. ENROLL_MODE = 0x31/0x41, SEND_SCAN_SUCCESS = 0x31



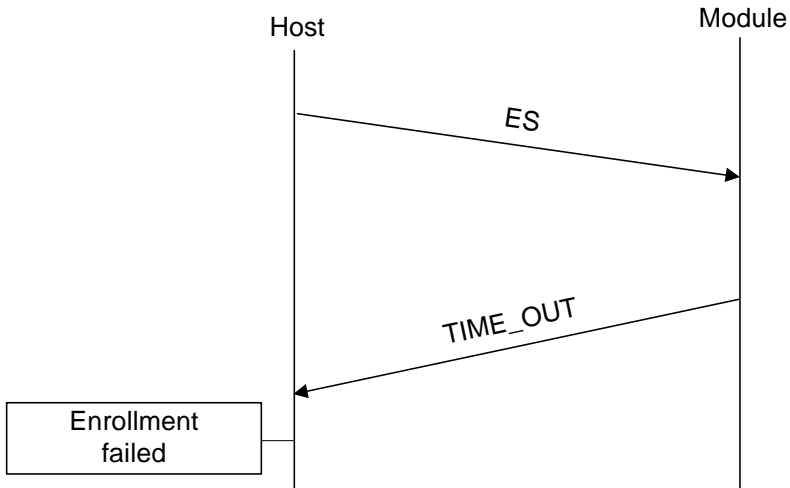
3. ENROLL_MODE = 0x31/0x41, SEND_SCAN_SUCCESS = 0x30



4. ENROLL_MODE = 0x32/0x42, SEND_SCAN_SUCCESS = 0x30

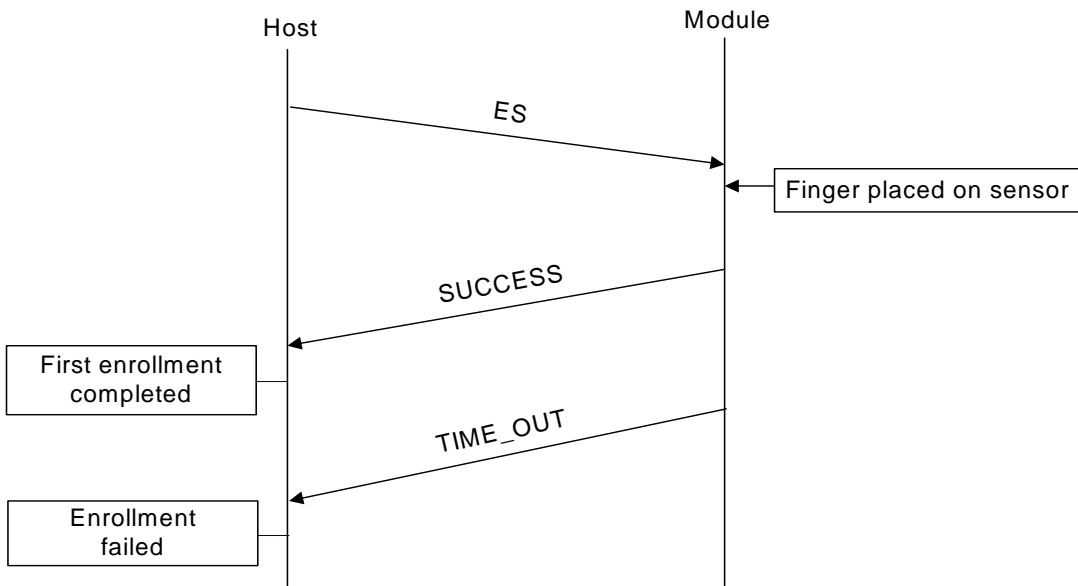


5. Time out occurred in the first enrollment



6. Time out occurred in the second enrollment

(When ENROLL_MODE = 0x31/0x41, SEND_SCAN_SUCCESS = 0x30)



Request command

Field	Data	Description
Start code	0x40	
Command	0x05	
Param	User ID	
Size	NULL	
Flag	0x71 0x79 0x74 0x70 0x84 0x85 0x92	ADD_NEW AUTO_ID CONTINUE CHECK_ID CHECK_FINGER ⁽¹⁾ CHECK_FINGER_AUTO_ID ⁽¹⁾ ADD_DURESS ⁽²⁾
Checksum	#	
End code	0x0A	

(1) These parameters are available for FM only.

(2) These parameters are available for FM only.

Response command

Field	Data	Description
Start code	0x40	
Command	0x05	
Param	User ID	
Size	Image Quality	Score 0 ~ 100
Error	(0x62) 0x63 0x6C 0x61 0x6B 0x6D	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT SUCCESS TRY_AGAIN MEM_FULL

	0x72	FINGER_LIMIT
	0x76	INVALID_ID
	0x6E	EXIST_ID
	0x86	EXIST_FINGER
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	An error occurred during enrollment process.
MEM_FULL	Flash memory is full.
FINGER_LIMIT	The number of fingerprints enrolled in same ID exceeds its limit (10).
INVALID_ID	The requested user ID is invalid. Note that '0x0000' cannot be used.
EXIST_ID	The requested user ID exists. (In case CHECK_ID flag is used)
EXIST_FINGER	The same finger is already enrolled. (In case CHECK_FINGER or CHECK_FINGER_AUTO_ID is used.) The same finger is already enrolled as non-duress.(In case ADD_DURESS option is given.)

All the error codes above are sent at the end of process, except "SCAN_SUCCESS," which appears in an intermediate stage.

Example

When enrolling a new fingerprint template with an user ID, '0x0123,'

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x05	0x0123	0x00	0x00	0x69	0x0A

The actual value input in the Param:

1 st place	2 nd place	3 rd place	4 th place
0x23	0x01	0x00	0x00

Byte transmission order

0x40, 0x05, 0x23, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x69, 0x0A

When adding a fingerprint template with an user ID, '0x0123,'

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x05	0x0123	0x00	0x71	0xDA	0x0A

Byte transmission order

0x40, 0x05, 0x23, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x71, 0xDA, 0x0A

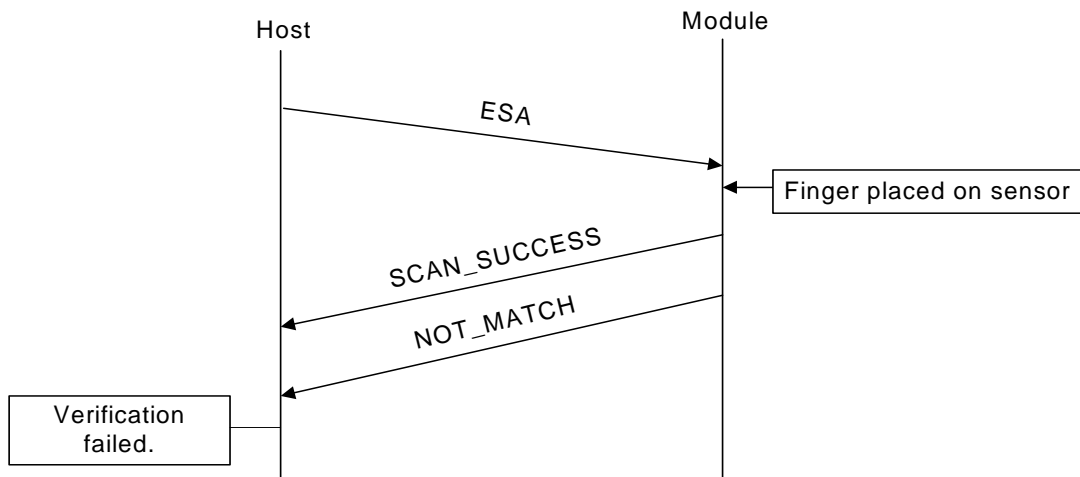
ESA : ES with Administrator's Verification

Enroll and Delete commands can change the fingerprint DB stored in the module. For some applications, it might be necessary to obtain administrator's permission before enrolling or deleting fingerprints. There are 5 commands which need administrator's verification before proceeding: ESA, EWA, DSA, DWA, and DAA. To process these commands, a user with proper administration level should verify himself first. If there is no user with corresponding administration level, these commands will fail with UNSUPPORTED error code. In case the verification fails, NOT_MATCH error code will be returned. The only exception is that ESA and EWA will succeed when the fingerprint DB is empty. In that case, the first user enrolled by ESA or EWA will have ADMIN_ALL level. See AW for details of administration levels.

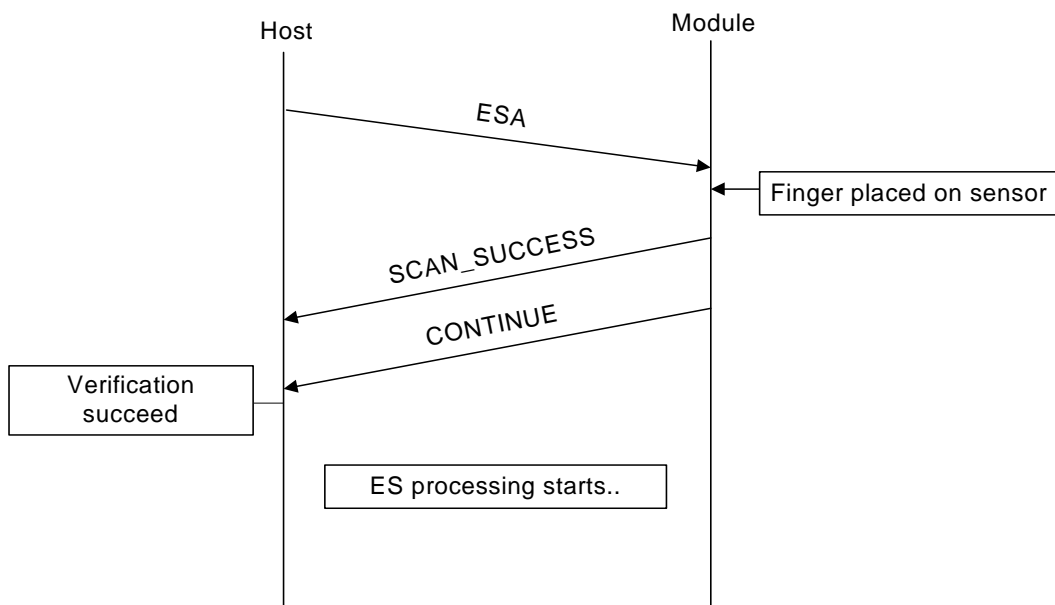
ESA is equal to ES except that it needs administrator's verification before enrolling a fingerprint. A user with ADMIN_ENROLL or ADMIN_ALL privileges should verify himself first to start the ES processing.

Timelines of ESA:

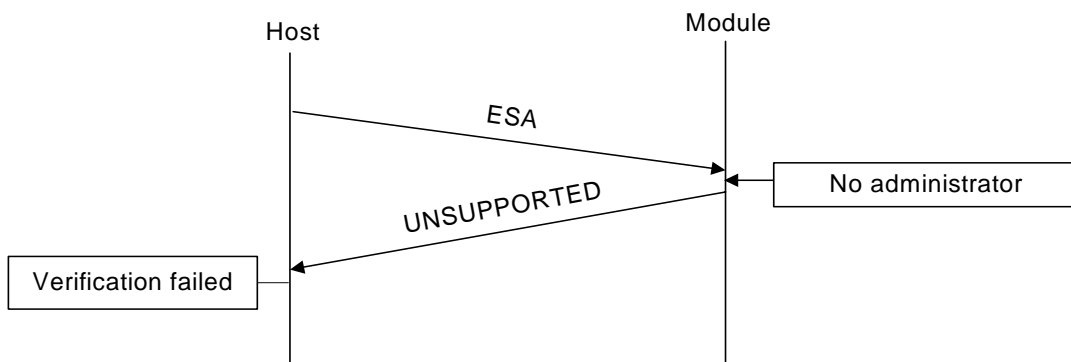
1. When administrator's verification fails (SEND_SCAN_SUCCESS = 0x31)



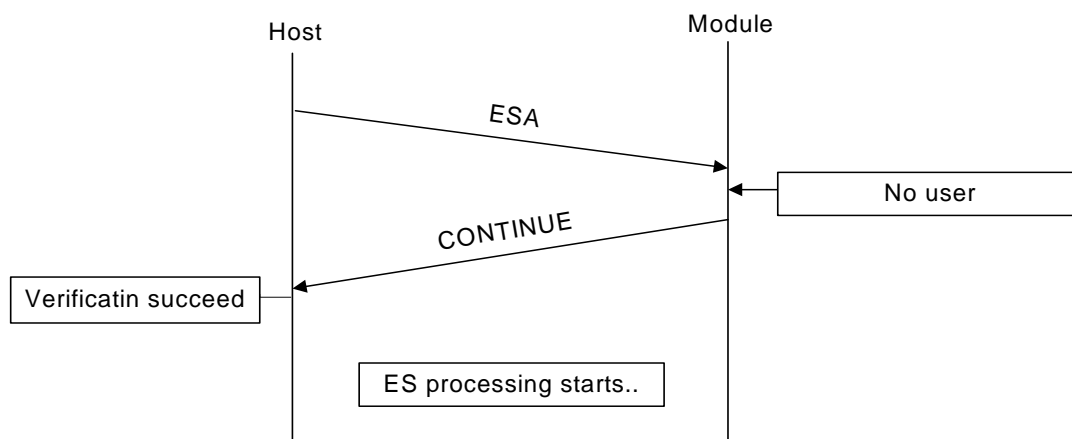
2. When administrator's verification succeeds (SEND_SCAN_SUCCESS = **0x31**),



3. When there is no user with corresponding administration levels (SEND_SCAN_SUCCESS = **0x31**),



4. When there is no user (SEND_SCAN_SUCCESS = **0x31**),



Request command

Field	Data	Description
Start code	0x40	
Command	0x70	
Param	User ID	
Size	NULL	
Flag	0x71 0x79 0x74 0x70 0x84 0x85 0x92	ADD_NEW AUTO_ID CONTINUE CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x70	

Param	User ID	
Size	Image Quality	Score 0 ~ 100
Error	(0x62) 0x63 0x74 0x6C 0x61 0x6B 0x6D 0x72 0x76 0x6E 0x6A 0x75 0x86	(SCAN_SUCCESS) SCAN_FAIL CONTINUE TIME_OUT SUCCESS TRY_AGAIN MEM_FULL FINGER_LIMIT INVALID_ID EXIST_ID NOT_MATCH UNSUPPORTED EXIST_FINGER
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
CONTINUE	Administrator's verification succeeds.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	An error occurred during enrollment process.
MEM_FULL	Flash memory is full.
FINGER_LIMIT	The number of fingerprints enrolled in same ID exceeds its limit (10).
INVALID_ID	The requested user ID is invalid. Note that '0x0000' cannot be used.
EXIST_ID	The requested user ID exists. (In case CHECK_ID flag is used)

EXIST_FINGER	The same finger is already enrolled. (In case CHECK_FINGER or CHECK_FINGER_AUTO_ID are used.)
NOT_MATCH	Administrator's verification fails.
UNSUPPORTED	There is no user with ADMIN_ALL or ADMIN_ENROLL levels.

All the error codes above are sent at the end of process, except "SCAN_SUCCESS" and "CONTINUE," which appear in an intermediate stage.

EI : Enroll by Image

Enrolls a user ID by transmitting the fingerprint image from the host. The fingerprint image data can be transmitted following the EI packet. The Enroll Mode for scanning fingerprint 1 or 2 times does not affect for this command and only one image is used for enrollment.

Request command

Field	Data	Description
Start code	0x40	
Command	0x06	
Param	User ID	
Size	Image size	
Flag	0x71 0x79 0x70 0x84 0x85 0x92	ADD_NEW AUTO_ID CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

First transmit the request command, then the raw data (fingerprint image) and finally the 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0x06	

Param	User ID	
Size	Image Quality	Score 0 ~ 100
Error	(0x62) 0x61 0x6B 0x6D 0x72 0x76 0x6E 0x86	(SCAN_SUCCESS) SUCCESS TRY_AGAIN MEM_FULL FINGER_LIMIT INVALID_ID EXIST_ID EXIST_FINGER
Checksum	#	
End code	0x0A	

Refer to ES for the error codes.

Example

When enrolling a fingerprint by transmitting image data of 8,000 (0x1F40) bytes with a user ID, '0x0123'

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x06	0x0123	0x1F40	0x00	0xC9	0x0A

+

Fingerprint Image	0x0A
-------------------	------

The actual value input in the Param

1 st place	2 nd place	3 rd place	4 th place
0x23	0x01	0x00	0x00

The actual value input in the Size

1 st place	2 nd place	3 rd place	4 th place
0x40	0x1F	0x00	0x00

Byte transmission order

0x40, 0x06, 0x23, 0x01, 0x00, 0x00, 0x40, 0x1F, 0x00, 0x00, 0x00, 0xC9, 0x0A, 0x**,
 0x**, ..., 0x**, 0x0A

EIX : EI with Extended Data Transfer Protocol

Enrolls a user ID by transmitting the fingerprint image from the host. The transfer of image conforms to the Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x80	
Param	User ID	
Size	Image size	
Flag	0x71 0x79 0x70 0x84 0x85 0x92	ADD_NEW AUTO_ID CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x80	
Param	User ID	
Size	Image Quality	Score 0 ~ 100
Error	(0x62)	(SCAN_SUCCESS)

	0x61	SUCCESS
	0x6B	TRY_AGAIN
	0x6D	MEM_FULL
	0x72	FINGER_LIMIT
	0x76	INVALID_ID
	0x6E	EXIST_ID
	0x82	DATA_ERROR
	0x83	DATA_OK
	0x86	EXIST_FINGER
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	An error occurred during enrollment process.
MEM_FULL	Flash memory is full.
FINGER_LIMIT	The number of fingerprints enrolled in same ID exceeds its limit (10).
INVALID_ID	The requested user ID is invalid. Note that '0x0000' cannot be used.
EXIST_ID	The requested user ID exists. (In case CHECK_ID flag is used)
DATA_ERROR	The checksum of a data packet is incorrect.
DATA_OK	The checksum of a data packet is correct.
EXIST_FINGER	The same finger is already enrolled. (In case CHECK_FINGER or CHECK_FINGER_AUTO_ID are used.)

All the error codes above are sent at the end of process, except “SCAN_SUCCESS” and “DATA_OK,” which appear in an intermediate stage.

ET : Enroll by Template

Enrolls a user ID by transmitting a fingerprint template from the host. The fingerprint template data can be transmitted following the ET packet.

Request command

Field	Data	Description
Start code	0x40	
Command	0x07	
Param	User ID	
Size	Template size	
Flag	0x71 0x79 0x70 0x84 0x85 0x92	ADD_NEW AUTO_ID CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

First transmit the request command, then the fingerprint template, and finally the 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0x07	
Param	User ID	
Size	NULL	

Error	(0x62) 0x61 0x6B 0x6D 0x72 0x76 0x6E 0x86	(SCAN_SUCCESS) SUCCESS TRY_AGAIN MEM_FULL FINGER_LIMIT INVALID_ID EXIST_ID EXIST_FINGER
Checksum	#	
End code	0x0A	

Refer to ES for the error codes.

Example

When enrolling a fingerprint by transmitting its fingerprint template of 400 (0x190) byte with a user ID, '0x0123':

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x07	0x0123	0x190	0x00	0x8E	0x0A

+

Fingerprint Template	0x0A
----------------------	------

The actual value input in the Param

1 st place	2 nd place	3 rd place	4 th place
0x23	0x01	0x00	0x00

The actual value input in the Size

1 st place	2 nd place	3 rd place	4 th place

0x90	0x01	0x00	0x00
------	------	------	------

Byte transmission order

0x40, 0x07, 0x23, 0x01, 0x00, 0x00, 0x90, 0x01, 0x00, 0x00, 0x00, 0x8E, 0x0A, 0x**,
0x**, ... , 0x**, 0x0A

ETX : ET with Extended Data Transfer Protocol

Enrolls a user ID by transmitting fingerprint templates from the host. Users can enroll multiple templates or a template multiple times. The transfer of template data conforms to the Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x87	
Param	User ID	
Size	(Number of templates << 24) (Number of enroll << 16) Template size	If Number of templates or Number of enroll is 0, it is assumed to be 1.
Flag	0x71 0x79 0x70 0x84 0x85 0x92	ADD_NEW AUTO_ID CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x87	

Param	User ID	
Size	Number of enrolled templates	
Error	(0x62) 0x61 0x6B 0x6D 0x72 0x76 0x6E 0x82 0x83 0x86	(SCAN_SUCCESS) SUCCESS TRY_AGAIN MEM_FULL FINGER_LIMIT INVALID_ID EXIST_ID DATA_ERROR DATA_OK EXIST_FINGER
Checksum	#	
End code	0x0A	

Refer to EIX for the error codes.

Example

- (1) When enrolling ID 10 which has three 384-byte templates:

Start	Command	Param	Size*	Flag	Checksum	End
0x40	0x87	0x0A	0x3000180	0x00	0x55	0x0A

*Size = (3 << 24) | 384 = 0x3000180

After the request packet, send three 384-byte templates by Extended Data Transfer Protocol. The number of enrolled templates will be 3.

- (2) When enrolling a 384-byte template two times to ID 16:

Start	Command	Param	Size*	Flag	Checksum	End
0x40	0x87	0x10	0x20180	0x00	0x5A	0x0A

*Size = $(2 \ll 16) | 384 = 0x20180$

After the request packet, send a 384-byte template by Extended Data Transfer Protocol.
The number of enrolled templates will be 2.

EW : Enroll by Wiegand ID

Enrolls a user's fingerprint using Wiegand ID.

Request command

Field	Data	Description
Start code	0x40	
Command	0x1C	
Param	NULL	
Size	NULL	
Flag	0x71 0x74 0x70 0x84 0x85 0x92	ADD_NEW CONTINUE CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x1C	
Param	User ID	
Size	Image Quality	Score 0 ~ 100
Error	(0x62) 0x63 0x6C	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT

	0x61	SUCCESS
	0x6B	TRY_AGAIN
	0x6D	MEM_FULL
	0x72	FINGER_LIMIT
	0x76	INVALID_ID
	0x6E	EXIST_ID
	0x86	EXIST_FINGER
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for Wiegand input or fingerprint input.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	An error occurred during enrollment process.
MEM_FULL	Flash memory is full.
FINGER_LIMIT	The number of fingerprints enrolled in same ID exceeds its limit (10).
INVALID_ID	The requested user ID is invalid. Note that '0x0000' cannot be used.
EXIST_ID	The requested user ID exists. (In case CHECK_ID flag is used)
EXIST_FINGER	The same finger is already enrolled. (In case CHECK_FINGER or CHECK_FINGER_AUTO_ID are used.)

All the error codes above are sent at the end of process, except "SCAN_SUCCESS," which appears in an intermediate stage.

EWA : EW with Administrator's Verification

Enrolls a user's fingerprint using Wiegand ID with administrator's verification. See ESA for details of administrator's verification.

Request command

Field	Data	Description
Start code	0x40	
Command	0x71	
Param	NULL	
Size	NULL	
Flag	0x71 0x74 0x70 0x84 0x85 0x92	ADD_NEW CONTINUE CHECK_ID CHECK_FINGER CHECK_FINGER_AUTO_ID ADD_DURESS
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x71	
Param	User ID	
Size	Image Quality	Score 0 ~ 100
Error	(0x62)	(SCAN_SUCCESS)

	0x63	SCAN_FAIL
	0x74	CONTINUE
	0x6C	TIME_OUT
	0x61	SUCCESS
	0x6B	TRY_AGAIN
	0x6D	MEM_FULL
	0x72	FINGER_LIMIT
	0x76	INVALID_ID
	0x6E	EXIST_ID
	0x6A	NOT_MATCH
	0x75	UNSUPPORTED
	0x86	EXIST_FINGER
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
CONTINUE	Administrator's verification succeeds.
TIME_OUT	Timeout for Wiegand input or fingerprint input.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	An error occurred during enrollment process.
MEM_FULL	Flash memory is full.
FINGER_LIMIT	The number of fingerprints enrolled in same ID exceeds its limit (10).
INVALID_ID	The requested user ID is invalid. Note that '0x0000' cannot be used.
EXIST_ID	The requested user ID exists. (In case CHECK_ID flag is used)
NOT_MATCH	Administrator's verification fails.
UNSUPPORTED	There is no user with ADMIN_ALL or ADMIN_ENROLL levels.

EXIST_FINGER	The same finger is already enrolled. (In case CHECK_FINGER or CHECK_FINGER_AUTO_ID are used.)
--------------	---

All the error codes above are sent at the end of process, except “SCAN_SUCCESS” and “CONTINUE,” which appear in an intermediate stage.

VS : Verify by Scan

Verifies if a fingerprint input on the sensor matches the enrolled fingerprint of the corresponding user ID.

Request command

Field	Data	Description
Start code	0x40	
Command	0x08	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x08	
Param	User ID	
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x63 0x6C 0x61 0x6B 0x69 0x6A	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT SUCCESS TRY_AGAIN NOT_FOUND NOT_MATCH

	0x90 0x91 0x94	REJECTED_ID DURESS_FINGER ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for the fingerprint input.
SUCCESS	Verification is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	Requested user ID is not found.
NOT_MATCH	The input fingerprint does not match the enrolled fingerprint of the corresponding user ID.
REJECTED_ID	Authentication mode is AUTH_REJECT or the ID is in the blacklist.
DURESS_FINGER	Duress finger is matched.
ENTRANCE_LIMIT	Authentication fails since the entrance limit is exceeded.

All the error codes above are displayed at the end of the process, except “SCAN_SUCCESS,” which appears at the intermediate stage.

VI : Verify by Image

Verifies if the fingerprint image transmitted from the host matches the enrolled fingerprint of the corresponding user ID. The fingerprint image can be transmitted following the VI packet.

Request command

Field	Data	Description
Start code	0x40	
Command	0x09	
Param	User ID	
Size	Image size	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then the fingerprint image, and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x09	
Param	User ID	
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x61 0x6B 0x69 0x6A	(SCAN_SUCCESS) SUCCESS TRY_AGAIN NOT_FOUND NOT_MATCH

	0x90 0x91 0x94	REJECTED_ID DURESS_FINGER ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Refer to VS for the error codes.

VIX: VI with Extended Data Transfer Protocol

Verifies if the fingerprint image transmitted from the host matches the enrolled fingerprint of the corresponding user ID. The transfer of image conforms to the Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x82	
Param	User ID	
Size	Image size	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x82	
Param	User ID	
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x61 0x6B 0x69 0x6A 0x82	(SCAN_SUCCESS) SUCCESS TRY_AGAIN NOT_FOUND NOT_MATCH DATA_ERROR

	0x83 0x90 0x91 0x94	DATA_OK REJECTED_ID DURESS_FINGER ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SUCCESS	Verification is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	Requested user ID is not found.
NOT_MATCH	The input fingerprint does not match the enrolled fingerprint of the corresponding user ID.
DATA_ERROR	The checksum of a data packet is incorrect.
REJECTED_ID	Authentication mode is AUTH_REJECT or the ID is in the blacklist.
DURESS_FINGER	Duress finger is matched.
ENTRANCE_LIMIT	Authentication fails since the entrance limit is exceeded.

All the error codes above are displayed at the end of the process, except “SCAN_SUCCESS” and “DATA_OK,” which appear at the intermediate stage.

VT : Verify by Template

Verifies if the fingerprint template transmitted from the host matches the enrolled fingerprint of the corresponding user ID. The fingerprint template can be transmitted following the VT packet.

Request command

Field	Data	Description
Start code	0x40	
Command	0x10	
Param	User ID	
Size	Template size	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then the fingerprint template, and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x10	
Param	User ID	
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x61 0x6B 0x69 0x6A	(SCAN_SUCCESS) SUCCESS TRY_AGAIN NOT_FOUND NOT_MATCH

	0x90 0x91 0x94	REJECTED_ID DURESS_FINGER ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Refer to VS for the error codes.

VW : Verify by Wiegand ID

Verifies if a fingerprint input on the sensor matches the enrolled fingerprint of the corresponding Wiegand ID.

Request command

Field	Data	Description
Start code	0x40	
Command	0x1D	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x1D	
Param	User ID	
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x63 0x6C 0x61 0x6B 0x69 0x6A	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT SUCCESS TRY_AGAIN NOT_FOUND NOT_MATCH

	0x90 0x91 0x94	REJECTED_ID DURESS_FINGER ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for Wiegand input or fingerprint input.
SUCCESS	Verification is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	Requested user ID is not found.
NOT_MATCH	The input fingerprint does not match the enrolled fingerprint of the corresponding user ID.
REJECTED_ID	Authentication mode is AUTH_REJECT or the ID is in the blacklist.
DURESS_FINGER	Duress finger is matched.
ENTRANCE_LIMIT	Authentication fails since the entrance limit is exceeded.

All the error codes above are displayed at the end of the process, except “SCAN_SUCCESS,” which appears at the intermediate stage.

VH : Verify Host Template by Scan

Transmits a fingerprint template from the host to the module and verifies if it matches the live fingerprint input from the sensor. As in the Smart Card, the VH command is used when the fingerprint template is not stored in the module but transmitted by the host.

Request command

Field	Data	Description
Start code	0x40	
Command	0x22	
Param	Number of templates	Number of templates verified. Valid value is 0 to 10. 0 means 1 template.
Size	Template size	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then each fingerprint templates followed by 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0x22	
Param	NULL	
Size	NULL	
Error	(0x62) 0x63 0x6C	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT

	0x61 0x6B 0x6A	SUCCESS TRY_AGAIN NOT_MATCH
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint template scanning by module is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint template matches the input fingerprint.
TRY_AGAIN	Fingerprint image is not good.
NOT_MATCH	Fingerprint template does not match the input fingerprint.

WSL : Write Security Level of a User

At default, the Security Level parameter is applied both to identification and verification. By using WSL command, different security level can be defined per each user. This security level is used only for verifying the specific user. The security level for identification is not changed by this command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x6B	
Param	User ID	
Size	Security level	0: Same as the Security Level parameter. 1: 1/1,000 2: 3/10,000 3: 1/10,000 4: 3/100,000 5: 1/100,000 6: 3/1,000,000 7: 1/1,000,000 8: 3/10,000,000 9: 1/10,000,000 10: 3/100,000,000 11: 1/100,000,000
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x6B	
Param	NULL	
Size	NULL	
Error	0x61 0x69 0x75 0x76	SUCCESS NOT_FOUND UNSUPPORTED INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Security level is written successfully.
NOT_FOUND	There is no such user ID.
UNSUPPORTED	Invalid security level.
INVALID_ID	Invalid user ID.

RSL : Read Security Level of a User

Reads the security level of a user set by WSL command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x6C	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x6C	
Param	NULL	
Size	Security Level	Same as WSL command
Error	0x61	SUCCESS
	0x69	NOT_FOUND
	0x76	INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Security level is read successfully.
NOT_FOUND	There is no such user ID.
INVALID_ID	Invalid user ID.

IS : Identify by Scan

Identifies the user ID among the enrolled fingerprints that matches the input fingerprint from the sensor. While VS verifies the fingerprint of a specific user ID, IS scans all the enrolled fingerprints within designated range of user ID. If no fingerprint is found which matches the input data, an error code, "NOT_FOUND," will appear on the error field.

Request command

Field	Data	Description
Start code	0x40	
Command	0x11	
Param	NULL Range of user ID	Search all enrolled user ID Lower 16 bit value denotes lower limit of user ID, and higher 16 bit value denotes upper limit of user ID.
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x11	
Param	User ID	Identified User ID
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62)	(SCAN_SUCCESS)

	0x63	SCAN_FAIL
	0x6C	TIME_OUT
	0x61	SUCCESS
	0x6B	TRY_AGAIN
	0x69	NOT_FOUND
	0x7A	TIMEOUT_MATCH
	0x90	REJECTED_ID
	0x91	DURESS_FINGER
	0x94	ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Identification is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	No matching template.
TIMEOUT_MATCH	Timeout for matching.
REJECTED_ID	Authentication mode is AUTH_REJECT or the ID is in the blacklist.
DURESS_FINGER	Duress finger is matched.
ENTRANCE_LIMIT	Authentication fails since the entrance limit is exceeded.

All the error codes above are displayed at the end of the process, except “SCAN_SUCCESS,” which appears at the intermediate stage.

II : Identify by Image

Identifies a corresponding user ID among all the enrolled fingerprints, which matches the transmitted fingerprint image from the host. The fingerprint image can be transmitted following the II packet.

Request command

Field	Data	Description
Start code	0x40	
Command	0x12	
Param	NULL Range of user ID	Search all enrolled user ID Lower 16 bit value denotes lower limit of user ID, and higher 16 bit value denotes upper limit of user ID.
Size	Image size	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then the fingerprint image(RAW data), and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x12	
Param	User ID	Identified User ID
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62)	(SCAN_SUCCESS)

	0x61	SUCCESS
	0x6B	TRY_AGAIN
	0x69	NOT_FOUND
	0x7A	TIMEOUT_MATCH
	0x90	REJECTED_ID
	0x91	DURESS_FINGER
	0x94	ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Refer to IS for the error codes.

II X : II with Extended Data Transfer Protocol

Identifies a corresponding user ID among all the enrolled fingerprints, which matches the transmitted fingerprint image from the host. The transfer of image conforms to the Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x81	
Param	NULL Range of user ID	Search all enrolled user ID Lower 16 bit value denotes lower limit of user ID, and higher 16 bit value denotes upper limit of user ID.
Size	Image size	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x81	
Param	User ID	Identified User ID
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x61 0x6B	(SCAN_SUCCESS) SUCCESS TRY_AGAIN

	0x69	NOT_FOUND
	0x7A	TIMEOUT_MATCH
	0x82	DATA_ERROR
	0x83	DATA_OK
	0x90	REJECTED_ID
	0x91	DURESS_FINGER
	0x94	ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SUCCESS	Identification is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	No matching template.
TIMEOUT_MATCH	Timeout for matching.
DATA_ERROR	The checksum of a data packet is incorrect.
DATA_OK	The checksum of a data packet is correct.
REJECTED_ID	Authentication mode is AUTH_REJECT or the ID is in the blacklist.
DURESS_FINGER	Duress finger is matched.
ENTRANCE_LIMIT	Authentication fails since the entrance limit is exceeded.

All the error codes above are displayed at the end of the process, except “SCAN_SUCCESS” and “DATA_OK,” which appear at the intermediate stage.

IT : Identify by Template

Identifies a corresponding user ID among all the enrolled fingerprints, which matches the transmitted fingerprint template from the host. The fingerprint template can be transmitted following the IT packet.

Request command

Field	Data	Description
Start code	0x40	
Command	0x13	
Param	NULL Range of user ID	Search all enrolled user ID Lower 16 bit value denotes lower limit of user ID, and higher 16 bit value denotes upper limit of user ID.
Size	Template size	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmit the request command, then the fingerprint template, and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x13	
Param	User ID	Identified User ID
Size	Sub-ID	Order of fingerprints enrolled in the same ID
Error	(0x62) 0x61	(SCAN_SUCCESS) SUCCESS

	0x6B	TRY_AGAIN
	0x69	NOT_FOUND
	0x7A	TIMEOUT_MATCH
	0x90	REJECTED_ID
	0x91	DURESS_FINGER
	0x94	ENTRANCE_LIMIT
Checksum	#	
End code	0x0A	

Refer to IS for the error codes.

DA : Delete All Templates

Deletes all user IDs and fingerprint templates stored in the module.

Request command

Field	Data	Description
Start code	0x40	
Command	0x17	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x17	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Deletion of all templates is successfully completed.

DAA : DA with Administrator's Verification

Deletes all user IDs and fingerprint templates with administrator's verification. See ESA for details of administrator's verification. When there is no user, SUCCESS will be returned immediately.

Request command

Field	Data	Description
Start code	0x40	
Command	0x74	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x74	
Param	NULL	
Size	NULL	
Error	(0x62) 0x74 0x61 0x6A 0x75	(SCAN_SUCCESS) CONTINUE SUCCESS NOT_MATCH UNSUPPORTED
Checksum	#	

End code	0x0A	
----------	------	--

Error code

Error code	Description
SUCCESS	Deletion of all templates is successfully completed.
CONTINUE	Administrator's verification succeeds.
NOT_MATCH	Administrator's verification fails.
UNSUPPORTED	There is no user with ADMIN_ALL or ADMIN_DELETE levels.

All the error codes above are displayed at the end of the process, except "CONTINUE," which appear at the intermediate stage.

DT : Delete Template

Deletes a user ID and its fingerprint template stored in the module. Users can also delete just one template by selecting DELETE_ONLY_ONE option. Or, users can delete the enrolled templates of multiple IDs by selecting DELETE_MULTIPLE_ID option.

Request command

Field	Data	Description
Start code	0x40	
Command	0x16	
Param	User ID	
Size	Sub index or Last user ID	Sub index if DELETE_ONLY_ONE option is given. Last user ID if DELETE_MULTIPLE_ID option is given.
Flag	NULL DELETE_ONLY_ONE(0x70)* DELETE_MULTIPLE_ID(0x71)*	
Checksum	#	
End code	0x0A	

*These options has been supported since V1.5 firmware of FM

Response command

Field	Data	Description
Start code	0x40	
Command	0x16	
Param	N/A	

Size	Number of deleted IDs if DELETE_MULTIPLE_ID option is given	
Error	0x61 0x69	SUCCESS NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Templates are deleted successfully.
NOT_FOUND	There is no matching user ID found.

S

Example

(1) To delete user ID 10:

Param	Size	Flag
0x0A	Null	Null

(2) To delete the second template of user ID 10:

Param	Size	Flag
0x0A	1	DELETE_ONLY_ONE(0x70)

(1) To delete user ID 10 ~ 20:

Param	Size	Flag
0x0A	0x14	DELETE_MULTIPLE_ID(0x71)

DS : Delete by Scan

Identifies the user ID among the enrolled fingerprints that matches the input fingerprint from the sensor and deletes found ID and its fingerprint templates stored in the module.

Request command

Field	Data	Description
Start code	0x40	
Command	0x1E	
Param	NULL Range of user ID	Search all enrolled user ID Lower 16 bit value denotes lower limit of user ID, and higher 16 bit value denotes upper limit of user ID.
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x1E	
Param	User ID	Deleted user ID
Size	NULL	
Error	(0x62) 0x63 0x6C 0x61 0x6B	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT SUCCESS TRY_AGAIN

	0x69	NOT_FOUND
	0x7A	TIMEOUT_MATCH
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	Requested user ID is not found.
TIMEOUT_MATCH	Timeout for matching.

All the error codes above are displayed at the end of the process, except “SCAN_SUCCESS,” which appears at the intermediate stage.

DSA : DS with Administrator's Verification

Deletes by scan with administrator's verification. See ESA for details of administrator's verification. You cannot delete an administration id using this command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x72	
Param	NULL Range of user ID	Search all enrolled user ID Lower 16 bit value denotes lower limit of user ID, and higher 16 bit value denotes upper limit of user ID.
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x72	
Param	User ID	Deleted user ID
Size	NULL	
Error	(0x62) 0x63 0x74 0x6C 0x61	(SCAN_SUCCESS) SCAN_FAIL CONTINUE TIME_OUT SUCCESS

	0x6B	TRY_AGAIN
	0x69	NOT_FOUND
	0x7A	TIMEOUT_MATCH
	0x6A	NOT_MATCH
	0x75	UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint scanning is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
CONTINUE	Administrator's verification succeeds.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint enrollment is successfully completed.
TRY_AGAIN	Fingerprint image is not good.
NOT_FOUND	Requested user ID is not found.
TIMEOUT_MATCH	Timeout for matching.
NOT_MATCH	Administrator's verification fails
UNSUPPORTED	There is no user with ADMIN_ALL or ADMIN_DELETE levels.* Cannot delete an administration id using DSA command.**

* If UNSUPPORTED is returned immediately after DSA command

** If UNSUPPORTED is returned after administrator's verification succeeds.

All the error codes above are displayed at the end of the process, except "SCAN_SUCCESS" and "CONTINUE," which appear at the intermediate stage.

DW : Delete by Wiegand ID

Deletes Wiegand ID and its fingerprint template stored in the module.

Request command

Field	Data	Description
Start code	0x40	
Command	0x1F	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x1F	
Param	User ID	
Size	NULL	
Error	(0x62) 0x61 0x69 0x6C	(SCAN_SUCCESS) SUCCESS NOT_FOUND TIMEOUT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Template delete is successfully completed.
NOT_FOUND	There is no matching user ID found.
TIMEOUT	Timeout for Wiegand input.

DWA : DW with Administrator's Verification

Deletes by Wiegand ID with administrator's verification. See ESA for details of administrator's verification. You cannot delete an administration id using this command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x73	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x73	
Param	User ID	
Size	NULL	
Error	(0x62) 0x74 0x61 0x69 0x6C 0x6A 0x75	(SCAN_SUCCESS) CONTINUE SUCCESS NOT_FOUND TIMEOUT NOT_MATCH UNSUPPORTED

Checksum	#	
End code	0x0A	

Error code

Error code	Description
CONTINUE	Administrator's verification succeeds.
SUCCESS	Template delete is successfully completed.
NOT_FOUND	There is no matching user ID found.
TIMEOUT	Timeout for Wiegand input.
NOT_MATCH	Administrator's verification fails.
UNSUPPORTED	There is no user with ADMIN_ALL or ADMIN_DELETE levels.* Cannot delete an administration id using DWA command.**

* If UNSUPPORTED is returned immediately after DSA command

** If UNSUPPORTED is returned after administrator's verification succeeds.

All the error codes above are displayed at the end of the process, except "CONTINUE," which appear at the intermediate stage.

LT : List User ID

Reads the list of user IDs enrolled in the module. With block index and block size parameters, users can receive part of enrolled IDs. For example, assume that there are 490 enrolled templates. If block size is 50 and block index is 0, only the first 50 IDs(1st ~ 50th) will be returned. If block index is 9, the last 40 IDs(451st ~ 490th) will be returned. If block size is 0, the module ignores block index and returns all the IDs.

Request command

Field	Data	Description
Start code	0x40	
Command	0x18	
Param	Block Index ⁽¹⁾	Starting from 0.
Size	Block Size ⁽¹⁾	If block size is 0, the module ignores the block index and returns all the IDs.
Flag	NULL	
Checksum	#	
End code	0x0A	

(1) These parameters are available FM only.

Response command

Field	Data	Description
Start code	0x40	
Command	0x18	
Param	Templates count	The number of templates returned
Size	Data size	(The number of templates returned) * 4
Error	0x61	SUCCESS

	0x76	INVALID_ID
Checksum	#	
End code	0x0A	

First transmits the response command and then the user ID.

User ID (4byte)	User ID (4byte)	...	User ID (4byte)	User ID (4byte)	0x0A
--------------------	--------------------	-----	--------------------	--------------------	------

Error code

Error code	Description
SUCCESS	Successfully completed.
INVALID_ID	Block index is out of range.

Example

If the user IDs to be returned are 0x0304, 0x0587, 0x8859

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x18	0x03	0x0C	0x61	0xC8	0x0A

The actual value input in the Param

1 st place	2 nd place	3 rd place	4 th place
0x03	0x00	0x00	0x00

The actual value input in the Size

1 st place	2 nd place	3 rd place	4 th place
0x0C	0x00	0x00	0x00

User ID Data

User ID	User ID	User ID	
0x0304	0x0587	0x8859	0x0A

Byte transmission order

0x40, 0x18, 0x03, 0x00, 0x00, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x61, 0xC8, 0x0A,

0x04, 0x03, 0x00, 0x00, 0x87, 0x05, 0x00, 0x00, 0x59, 0x88, 0x00, 0x00, 0x0A

LTX : LT with Extended Data Transfer Protocol

Reads the list of user IDs enrolled in the module with Extended Data Transfer Protocol. LTX returns 8 byte data for each template. The 2 byte checksum is calculated by summing up all the template data. This checksum can be used to check if two templates are identical.

User ID	(Security Level << 4) Duress	(Authentication Mode << 6) (Administration Level << 4) Sub ID	Checksum
4 bytes	1 byte	1 byte	2 bytes

Request command

Field	Data	Description
Start code	0x40	
Command	0x86	
Param	0	.
Size	Maximum length of data packet size	If 0, it is the total size of data
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x86	
Param	Templates count	The number of templates to be returned
Size	Data size	(The number of templates to be returned) * 8

Error	0x61 0x6D	SUCCESS MEM_FULL
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Successfully completed.
MEM_FULL	Out of memory.

Example

For example, Template 1 and Template 2 are enrolled to ID 1 and Template 1 is duress finger. Template 3 and Template 4 are enrolled to ID2 and the administration level of ID2 is ADMIN_ALL (0x03).

(1) Request Packet

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x86	0x00	0x00	0x00	0xC6	0x0A

Since size parameter is 0, the module will send all the data in one data packet.

(2) Response Packet

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x86	0x04	0x20	0x61	0x4B	0x0A

Template Count: 4, Data Size: 32

(3) Data Packet Header

Start	Command	Num of Packet	Packet Index	Data Size	Flag	Checksum	End
0x40	0x86	0x01	0x00	0x20	0x00	0xE7	0x0A

(4) Data Packet Body

	ID	Duress	(Admin Level << 4) Sub ID	Checksum
Template 1	0x01	0x01	0x00	##
Template 2	0x01	0x00	0x01	##
Template 3	0x02	0x00	0x30	##
Template 4	0x02	0x00	0x31	##

CT : Check User ID

Checks if the user ID exists in the module. Also returns the existing number of enrolled templates for the user ID.

Request command

Field	Data	Description
Start code	0x40	
Command	0x19	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x19	
Param	User ID	
Size	Number of Templates	Number of enrolled templates for the user ID
Error	0x6E 0x69	EXIST_ID NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
EXIST_ID	The user ID exists in the module.
NOT_FOUND	The user ID is not found in the module.

FP : Fix All Provisional Templates

Fix all provisional templates which are enrolled when the Provisional Enroll of system parameter is on.

Request command

Field	Data	Description
Start code	0x40	
Command	0x23	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x23	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
	0x69	NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	All provisional templates are fixed.
NOT_FOUND	No provisional template exists.

DP : Delete All Provisional Templates

Delete all provisional templates which are enrolled when the Provisional Enroll of system parameter is on.

Request command

Field	Data	Description
Start code	0x40	
Command	0x24	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x24	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
	0x69	NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	All provisional templates are deleted.
NOT_FOUND	No provisional template exists.

RI : Read Image

Reads the fingerprint image received from the sensor.

First execute such commands as ES, VS and IS in order to use this command. The fingerprint images in this process are transmitted in a raw data format as SEND_IMAGE structures, of which has a structure as follows. In order to save any fingerprint images in the module, first execute the ES, VS and IS commands. Without executing these commands, the module cannot be updated.

```
typedef struct SEND_IMAGE {
    int width;
    int height;
    int compressed;
    int encrypted;
    int binary;
    int img_len;
    int template_len;
    char buffer[BUFFER_LENGTH];
} image_t;
```

Member Variable	Description
Width (4byte)	Width of fingerprint images
Height (4byte)	Height of fingerprint images
Compressed (4byte)	Compression status – Currently not used In case of 4bit gray mode, starting point of actually transmitted image is designated. Lower word is x coordinate and higher word is y coordinate.
Encrypted (4byte)	Encryption status– Currently not used In case of 4bit gray mode, size of actually transmitted image is designated. Lower word is width and higher word is height.

Binary (4byte)	Image Format : 0 – gray, 1 – binary, 2 – 4bit gray
Img_len (4byte)	Size of the fingerprint image received from the sensor = width * height
Template_len (4byte)	Size of the fingerprint template – Currently not used
Buffer (1byte*BUFFER_LENGTH)	<p>Actual raw image data</p> <p>BUFFER_LENGTH=200*1024 (=200Kbyte) max</p> <p>1) When the image format is binary: 1 byte per 8 pixels</p> <p style="padding-left: 40px;">Size of data actually transmitted: $\text{Img_len}/8+1$</p> <p>2) When the image format is gray: 1 byte per 1 pixel</p> <p style="padding-left: 40px;">Size of data actually transmitted: $\text{Img_len} (= \text{Width} * \text{Height})$</p> <p>3) When the image format is 4bit gray: 1 byte per 2 pixel</p> <p style="padding-left: 40px;">Size of data actually transmitted: $\text{Actual width} * \text{height} / 2$</p>

Request command

Field	Data	Description
Start code	0x40	
Command	0x20	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x20	
Param	NULL	

Size	Data size	When image format is binary: 4*7 + (Img_len/8)+1 When image format is gray: 4*7 + Img_len When image format is 4 bit gray: 4*7 + actual width * actual height / 2
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

First transmits the response command, then the image data, and finally the 0x0A

Error code

Error code	Description
SUCCESS	Fingerprint image reading is successfully completed.

Example

Image reading request command:

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x20	0x00	0x00	0x00	0x60	0x0A

Example of a request command sent from the module:

- Infineon sensor: 224 * 288 pixels = 64512 (= 0xFC00) pixels

- Image format: binary
- Binary image data size: $(64512 / 8) + 1 = 8065$
- Total data size: $(4 \text{ byte} * 7) + 8065 \text{ byte} = 8093 (=0x1F9D) \text{ byte}$

Start	Command	Param	Size	Error	Checksum	End
0x40	0x20	0x00	0x1F9D	0x61	0x7D	0x0A

Width	Height	Compressed	Encrypted	Binary	Img_len	Template_len
0xE0	0x120	0x00	0x00	0x01	0xFC00	0x00

Buffer[0]	Buffer[1]	...		Buffer[8063]	Buffer[8064]	End
0xEF	0x2C	0x1A	0xFE	0x0A

Received data order

0x40, 0x20, 0x00, 0x00, 0x00, 0x00, 0x9D, 0x1F, 0x00, 0x00, 0x61, 0x7D, 0x0A,
 0xE0, 0x00, 0x00, 0x00, 0x20, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0xEF, 0x2C, ..., 0x1A, 0xFE, 0x0A

If the image format is binary, the data in the buffer of the SEND_IMAGE structures will be 1byte data packed from 8 pixels of the image raw data sent by the sensor. Below is an operation example to unpack the binary data with 1byte per 1 pixel.

```
Void img_display_from_buffer()
{
```

```
unsigned char *uncomp_buffer,*img_buf,*ptr;
int len,i;
unsigned long uncomp_len;
image_t receive_img;

// g_bin_buffer(buffer with received data inside)
// total_bin_size(total size of received data) are global variables
memcpy(&receive_img,(image_t*)g_bin_buffer,total_bin_size);

img_buf=new unsigned char[receive_img.img_len];
ptr=img_buf;
// Check if the image format is binary
if(receive_img.binary){
    // Data size sent in binary = (Actual image size)/8
    for(i=0;i<receive_img.img_len/8;i++){
        bit operation to divide 1 byte into 8 bytes
        *ptr=(receive_img.buffer[i] & 1)?255:0;
        ptr++;
        *ptr=(receive_img.buffer[i]>>1 & 1)?255:0;
        ptr++;
        *ptr=(receive_img.buffer[i]>>2 & 1)?255:0;
        ptr++;
        *ptr=(receive_img.buffer[i]>>3 & 1)?255:0;
        ptr++;
        *ptr=(receive_img.buffer[i]>>4 & 1)?255:0;
        ptr++;
        *ptr=(receive_img.buffer[i]>>5 & 1)?255:0;
        ptr++;
        *ptr=(receive_img.buffer[i]>>6 & 1)?255:0;
```

```
        ptr++;
        *ptr=(receive_img.buffer[i]>>7 & 1)?255:0;
        ptr++;
    }
} else {
    // Put the image into the buffer, for it is not in binary format.
    memcpy(img_buf,receive_img.buffer,receive_img.img_len);
}
// img_buf
}
```

RIX : RI with Extended Data Transfer Protocol

Reads the fingerprint image received from the sensor. The transfer of image conforms to the Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x84	
Param	NULL	
Size	Maximum length	Maximum length of a data packet body
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x84	
Param	NULL	
Size	Image size	When image format is binary: $4*7 + (\text{Img_len}/8)+1$ When image format is gray: $4*7 + \text{Img_len}$ When image format is 4 bit gray: $4*7 + \text{actual width} * \text{actual height} / 2$
Error	0x61	SUCCESS

Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Fingerprint image reading is successfully completed.

SI : Scan Image

Scan live fingerprint image from the sensor.

If the command is successful, captured image data is sent from the module, whose data format is same as that of RI command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x15	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x15	
Param	NULL	
Size	Image size	Size of image data transmitted
Error	(0x62) 0x63 0x6C 0x61	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT SUCCESS
Checksum	#	
End code	0x0A	

First transmits the response command, then the image data, and finally the 0x0A

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint image scanning by module is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint image is successfully read.

SIX : SI with Extended Data Transfer Protocol

Scan live fingerprint image from the sensor. The transfer of image conforms to the Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x83	
Param	NULL	
Size	Maximum length	Maximum length of a data packet body
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x83	
Param	NULL	
Size	Image size	Size of image data transmitted
Error	(0x62) 0x63 0x6C 0x61	(SCAN_SUCCESS) SCAN_FAIL TIME_OUT SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint image scanning by module is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint image is successfully read.

RT : Read Template

Reads the fingerprint templates stored in the module.

When reading the fingerprint template of a specific user ID, input the user ID in the Param field. Input “NULL” in the Param to read the latest fingerprint template created in the sensor.

Request command

Field	Data	Description
Start code	0x40	
Command	0x14	
Param	NULL User ID	NULL denotes the latest fingerprint template created
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x14	
Param	NULL	
Size	Template size	
Error	0x61 0x69 0x74	SUCCESS NOT_FOUND CONTINUE
Checksum	#	

End code	0x0A	
----------	------	--

First transmit the response command, then the fingerprint template, and finally the 0x0A

Error code

Error code	Description
SUCCESS	Fingerprint template reading is successfully completed.
CONTINUE	When there are more users to be enrolled.
NOT_FOUND	There is no fingerprint template or matching user ID found.

If there are more than two fingerprint templates enrolled as one user ID, the host will receive two response commands. If the error code of the first response command is read CONTINUE, it means that there is one more fingerprint enrolled as the corresponding user ID. Then, the host will receive the next packet instantly.

For example, if there are two fingerprint templates enrolled as one user ID, the process order is as follows:

Response command + Template1 + 0x0A + Response command + Template2 + 0x0A

If the error code of the first response command is read "SUCCESS," it means that there are no more fingerprint templates enrolled as the corresponding user ID.

RTX : RT with Extended Data Transfer Protocol

Reads the fingerprint templates stored in the module. Unlike RT which will read all the templates enrolled to an ID, users can read one template at a time by selecting a sub index. The transfer of template data conforms to Data Transfer Protocol. See Appendix B Extended Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0x89	
Param	NULL User ID	NULL denotes the latest fingerprint template created
Size	(Sub index << 16) Maximum length of a data packet body	Sub index is from 0 to 9.
Flag	0 1	Ignore sub index and reads all the templates of the specified ID. Reads one template of the specified sub index.
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x89	
Param	User ID	
Size	(Number of templates to be	

	returned << 16) Template size	
Error	0x61 0x69 0x75	SUCCESS NOT_FOUND UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The templates are read successfully.
NOT_FOUND	There is no fingerprint template or matching user ID found.
UNSUPPORTED	Invalid parameters.

ST : Scan Template

Reads the fingerprint template of a live fingerprint on sensor. This command is used when a host wants to read the fingerprint template of a live fingerprint without enrolling or verifying process. The fingerprint template is not stored in the module.

Request command

Field	Data	Description
Start code	0x40	
Command	0x21	
Param	NULL	
Size	NULL	
Flag	NULL ADD_CHECKSUM(0x70)*	If ADD_CHECKSUM option is given, the module will insert 4 byte checksum before the end code(0x0A).
Checksum	#	
End code	0x0A	

*ADD_CHECKSUM has been supported since V1.5 firmware of FM

Response command

Field	Data	Description
Start code	0x40	
Command	0x21	
Param	Image quality score	0 ~ 100
Size	Template size	
Error	(0x62) 0x63	(SCAN_SUCCESS) SCAN_FAIL

	0x6C	TIME_OUT
	0x61	SUCCESS
	0x6B	TRY_AGAIN
Checksum	#	
End code	0x0A	

First transmit the response command, then the fingerprint template, and finally the 0x0A. If ADD_CHECKSUM option is given, 4 byte checksum will be inserted before 0x0A. The checksum is calculated over the whole template data.

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint template scanning by module is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint template is successfully read.
TRY_AGAIN	Fingerprint image is not good.

Example

Refer to ES and RT

KS : Scan Template with Challenge Data

This command is similar to ST, except that 4 bytes of challenge data is provided in the command. Then, the response packet includes the supplementary data, made by encrypting (challenge data + obtained template).

The size of the returned data will be multiple of 32 bytes, since supported AES encryption algorithm processes data based on 32 byte block.

If the module is not in encryption mode, the responding supplementary data will be non-encrypted one of (challenge data + template data). The supplementary data should be followed by EOP.

Request command

Field	Data	Description
Start code	0x40	
Command	0x35	
Param	Challenge data	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x35	
Param	NULL	
Size	Template size	
Error	(0x62)	(SCAN_SUCCESS)

	0x63 0x6C 0x61 0x6B	SCAN_FAIL TIME_OUT SUCCESS TRY_AGAIN
Checksum	#	
End code	0x0A	

First transmit the response command, then the fingerprint template, and finally the 0x0A

Error code

Error code	Description
SCAN_SUCCESS	Fingerprint template scanning by module is successfully completed.
SCAN_FAIL	Sensor or fingerprint input has failed.
TIME_OUT	Timeout for fingerprint input.
SUCCESS	Fingerprint template is successfully read.
TRY_AGAIN	Fingerprint image is not good.

KW : Write Encryption Key

Changes encryption key, which is 32 bytes. In order to change the key, there should be no stored fingerprint data in the module in order to avoid ambiguity problem. After using the KW command, the SF command should be used to save encryption key in flash memory.

The request command should be followed by new key data of 32 bytes and packet end code. Thus, 33 bytes data (new key (32byte) + EOP(0x0A)) should be provided as the supplementary data.

If the request is unsuccessful, UNSUPPORTED code is returned.

Default encryption key is { 0x01, 0x00, , 0x00 }

Request command

Field	Data	Description
Start code	0x40	
Command	0x34	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmits the request command, then the key data, and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x34	

Param	NULL	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Key is successfully written.
UNSUPPORTED	Enrolled fingerprint exists.

ML : Get Size of User Memory

Returns size of user memory in Bytes.

Request command

Field	Data	Description
Start code	0x40	
Command	0x31	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x31	
Param	Size of user memory	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Successfully completed.

MW : Write to User Memory

Writes data to user memory. The SF command should be used to save in flash memory

Request command

Field	Data	Description
Start code	0x40	
Command	0x32	
Param	Offset	First position to be written
Size	Length	Data length
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmits the request command, then user data, and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x32	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
	0x75	UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	User data is successfully written.
UNSUPPORTED	Offset exceeds size of user memory.

MR : Read from User Memory

Reads data from user memory.

Request command

Field	Data	Description
Start code	0x40	
Command	0x33	
Param	Offset	First position to be read
Size	Length	Data length
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x33	
Param	NULL	
Size	Size	Actual size to be read
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

First transmits the response command, then user data, and finally the 0x0A

Error code

Error code	Description
------------	-------------

SUCCESS	User data is successfully read.
UNSUPPORTED	Offset exceeds size of user memory.

TW : Write Current Time

Writes current date and time. The year of the date parameter can be from 0 to 89 and will be added to 2000. For example, if the year is 2005, the year field of the date should be 5.

Request command

Field	Data	Description
Start code	0x40	
Command	0x3A	
Param	(DD<<16) (MM<<8) YY	Date
Size	(ss<<16) (mm<<8) hh	Time
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x3A	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Current time is successfully written.
---------	---------------------------------------

TR : Read Current Time

Reads current date and time.

Request command

Field	Data	Description
Start code	0x40	
Command	0x3B	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x3B	
Param	(DD<<16) (MM<<8) YY	Date
Size	(ss<<16) (mm<<8) hh	Time
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Deletion of all templates is successfully completed.
---------	--

LN : Get Number of Log Data

Returns number of log data.

Request command

Field	Data	Description
Start code	0x40	
Command	0x3C	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x3C	
Param	Number of stored log data	
Size	Number of total log data	Stored + empty
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Successfully completed.

LR : Read Log Data

Reads log data.

The structure of log data is as follows.

Field	Size	Description
Event	1 byte	Output function ID (See OW command)
Source	1 byte	0x01 : Host port 0x02 : Auxiliary port 0x03 : Wiegand input 0x04 : IN0 port 0x05 : IN1 port 0x06 : IN2 port 0x07 : Freescan
Date	3 byte	YY MM DD
Time	3 byte	hh mm ss
User ID	4 byte	User ID
Reserved	4 byte	This field can be set by LC command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x3D	
Param	Start index	
Size	Requested count	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x3D	
Param	NULL	
Size	Actual count	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

First transmits the response command, then log data (actual count x 16 bytes), and finally the 0x0A

Error code

Error code	Description
SUCCESS	Log data is successfully read.
UNSUPPORTED	Start index or requested size is invalid.

LD : Delete Log Data

Deletes log data. The requested count should be a multiple of 256. If it is not a multiple of 256, the next multiple number of logs will be deleted. For example, if the requested count is 300, 512 logs will be deleted.

Request command

Field	Data	Description
Start code	0x40	
Command	0x3E	
Param	Requested count	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x3E	
Param	Actual count to be deleted	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Log data is successfully deleted.
---------	-----------------------------------

LC : Set/Get the Custom Log Field

There is a 4 byte reserved field in each log record. LC command is used for setting or getting this custom value. If the custom field is not set, the reserved field will be filled with NULL. This command would be most useful for Time & Attendance applications, in which custom log records are necessary.

User can set 4 types of custom log field.

Custom Log Field	Description
0x00: Default	If the log is generated by Packet Protocol commands or freescan, this value will be used.
0x01: IN0	If the log is generated by an Input port, the respective values will be used.
0x02: IN1	
0x03: IN2	

Request command

Field	Data	Description
Start code	0x40	
Command	0x3F	
Param	Custom field value	
Size	Custom field type	0x00: Default 0x01: IN0 0x02: IN1 0x03: IN2
Flag	0 1	Read custom value Write custom value
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x3F	
Param	Custom field value	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Custom field is read or written successfully.
UNSUPPORTED	Invalid field type.

RCL : Read Current Log in Cache

The module has a 256 byte log cache, which can hold up to 16 recent log records. The log cache is cleared as soon as it is read by RCL command. By using this command, users can monitor the status of the module in real time.

The data transfer conforms to the Extended Data Transfer protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0xEC	
Param	NULL	
Size	Data packet size	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xEC	
Param	Number of log records	
Size	Log data size	Number of log records * 16
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Log data is successfully read.

CCL : Clear Log Cache

Clears the log cache. This command can be used when starting real time monitoring.

Request command

Field	Data	Description
Start code	0x40	
Command	0xEB	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xEB	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Loc cache is cleared.

WW : Write Wiegand Configuration (Deprecated)

Writes configuration of Wiegand I/O. If Wiegand input is set to a function, the module will start the function automatically on detecting a Wiegand input.

The function for Wiegand I/O is as follows.

Port	Function ID	Function Name
Input	0x00	No action
	0x12	Enroll Given ID
	0x22	Verify Given ID
	0x42	Delete Given ID
Output	0x5A	Output Matched ID

Since firmware V1.4, WW, WR, WG, and WS commands are deprecated. It is highly recommended that extended Wiegand protocol is used in place of these commands. See Appendix C. Extended Wiegand Protocol and WWX, WRX, WGX, WSX, WPW, and WPR commands.

Request command

Field	Data	Description
Start code	0x40	
Command	0x41	
Param	(FC Bits<<16) (Format<<8) Function ID	Format is always 0 for Wiegand 26 bits
Size	FC code	
Flag	0x00	Wiegand input port
	0x01	Wiegand output port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x41	
Param	NULL	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Configuration of Wiegand is successfully written.
UNSUPPORTED	Port or format is invalid.

Example

Set Wiegand input function to Verify Given ID(0x22), FC bits to 8, FC code to 1.

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x41	0x080022	0x01	0x00	0xAC	0x0A

Byte transmission order

0x40, 0x41, 0x22, 0x00, 0x08, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0xAC, 0x0A

WR : Read Wiegand Configuration (Deprecated)

Reads configuration of Wiegand I/O.

Request command

Field	Data	Description
Start code	0x40	
Command	0x42	
Param	NULL	
Size	NULL	
Flag	0x00 0x01	Wiegand input port Wiegand output port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x42	
Param	(FC Bits<<16) (Format<<8) Function ID	Format is always 0 for Wiegand 26 bits
Size	FC code	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Configuration of Wiegand is successfully read.
UNSUPPORTED	Port is invalid.

WG : Get Wiegand Input (Deprecated)

Reads current state of Wiegand input port.

Request command

Field	Data	Description
Start code	0x40	
Command	0x43	
Param	0 1	Clear state after read Remain state
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x43	
Param	Wiegand ID	
Size	FC code	
Error	0x61 0x69	SUCCESS NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	State of Wiegand input port is successfully read.
NOT_FOUND	No Wiegand input.

WS : Set Wiegand Output (Deprecated)

Output specified ID to Wiegand output port.

Request command

Field	Data	Description
Start code	0x40	
Command	0x44	
Param	Wiegand ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x44	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Wiegand ID is successfully output.

WM : Map Wiegand ID to Input Function

Maps specified Wiegand ID to an input function. When a Wiegand card reader detects the mapped IDs, the corresponding input function will be executed. For example, if Wiegand ID 10 was mapped to 0x10(Enroll by Scan), ES command will be processed. There can be at most 16 Wiegand mappings. The functions to which Wiegand ID can be mapped are as follows:

Function ID	Function Name
0x10	Enroll by Scan
0x11	Enroll by Wiegand ID
0x40	Delete by Scan
0x41	Delete by Wiegand ID
0x49	Delete All
0x1A	ES with Administrator's Verification
0x1B	EW with Administrator's Verification
0x4A	DS with Administrator's Verification
0x4B	DW with Administrator's Verification
0x4C	DA with Administrator's Verification

Request command

Field	Data	Description
Start code	0x40	
Command	0x68	
Param	Input Function	
Size	Wiegand ID	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x68	
Param	NULL	
Size	NULL	
Error	0x61 0x6D 0x75	SUCCESS MEM_FULL UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Wiegand ID is successfully mapped to the function.
MEM_FULL	No more mapping is allowed.
UNSUPPORTED	The input function is not supported.

WL : List Wiegand ID Mapping

Lists Wiegand IDs which are mapped to input functions.

Request command

Field	Data	Description
Start code	0x40	
Command	0x69	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x69	
Param	Number of mapping	
Size	Data size	(Number of mapping) * 5
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

First transmits the response command, then mapping data (number of mapping x 5 bytes), and finally the 0x0A. The mapping data consists of 4 byte Wiegand ID and 1 byte input function. For example, Wiegand ID 0x2450 is mapped to DS(0x40), and Wiegand ID 0x100 is mapped to ES(0x10), the mapping data will be as follows:

0x50 0x24 0x00 0x00 0x40 0x00 0x01 0x00 0x00 0x10

Error code

Error code	Description
SUCCESS	Wiegand mappings are successfully transferred.

WC : Clear Wiegand ID Mapping

Clears all the Wiegand ID mappings to input functions.

Request command

Field	Data	Description
Start code	0x40	
Command	0x6A	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x6A	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	All Wiegand ID mappings are cleared.

WWX : Write Wiegand Configuration

Set Wiegand format. See Appendix C. Extended Wiegand Protocol for details of format definition data.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC0	
Param	(Pulse Width in usec << 16) (Pulse interval in usec)	If 0, use default values: 50usec width and 2,000usec interval
Size	The size of format definition data	0 if the format is 26 bit standard
Flag	0x01 0x02 0x03	26 bit standard Pass through format Custom format
Checksum	#	
End code	0x0A	

First transmit the request command, then the format definition data, and finally the 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0xC0	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS

	0x75	UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Wiegand format is successfully written.
UNSUPPORTED	Invalid format data.

WRX : Read Wiegand Configuration

Read Wiegand format definition data. See Appendix C. Extended Wiegand Protocol for details of format definition data.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC1	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC1	
Param	(Wiegand format << 24) (Pulse width in usec << 16) (Pulse interval in usec)	
Size	The size of format definition data	0 if the format is 26 bit standard
Error	0x61 0x69	SUCCESS NOT_FOUND
Checksum	#	
End code	0x0A	

First transmits the response command and then the format definition data, and finally 0x0A.

Error code

Error code	Description
SUCCESS	Wiegand format data is successfully read.
NOT_FOUND	Wiegand format is not configured by WWX.

WGX : Get Wiegand Input

Reads Wiegand input string. Wiegand input string is converted to a 64 bit integer with the first bit as the most significant bit. For example, if 26 bit Wiegand input is 00000000 00000000 00110010 00(Site code is 0 and ID is 100), the higher 32 bit is 0x00000000 and the lower 32 bit is 0x000000c8. If flag is set to 0, the module will wait for new Wiegand input. If it is set to 1, the module will read the latched input.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC2	
Param	0 1	Clear state after read Remain state
Size	NULL	
Flag	0 1	Read new Wiegand input Read latched input
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC2	
Param	Higher 32 bit of Wiegand input	
Size	Lower 32 bit of Wiegand input	
Error	0x62 0x61	(SCAN_SUCCESS) SUCCESS

	0x69 0x6C	NOT_FOUND TIMEOUT
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SCAN_SUCCESS	New Wiegand input is detected.
SUCCESS	Wiegand input is successfully read.
NOT_FOUND	Wiegand format is not configured by WWX.
TIMEOUT	Timeout for Wiegand input.

WSX : Set Wiegand Output

Write Wiegand output string. As in WRX, the Wiegand string is represented by a 64 bit integer.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC3	
Param	Higher 32 bit of Wiegand output	
Size	Lower 32 bit of Wiegand output	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC3	
Param	NULL	
Size	NULL	
Error	0x61 0x6C	SUCCESS NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Wiegand output is successfully written.
NOT_FOUND	Wiegand format is not configured by WWX.

WFW : Set Wiegand Field

Set an alternative value for a Wiegand field. If this value is set, the field is always overwritten by this value. For example, this command can be used to set alternative site code for 26 bit standard format. This command is only applicable to 26 bit standard and Custom format.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC4	
Param	Alternative value	
Size	Field index	In 26 bit standard format, site code is index 0. If field index is set to 0xFF, it represents the FAIL ID. See WPW command.
Flag	0 1	Set alternative value. Clear the alternative value.
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC4	
Param	NULL	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED

Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Alternative value is successfully written.
UNSUPPORTED	Invalid index or value.

WFR : Get Wiegand Field

Read an alternative value of a Wiegand field. This command is only applicable to 26 bit standard and Custom format.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC5	
Param	NULL	
Size	Field index	In 26 bit standard format, site code is index 0. If field index is set to 0xFF, it represents the FAIL ID.
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC4	
Param	Alternative value	
Size	NULL	
Error	0x61 0x69 0x75	SUCCESS NOT_FOUND UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Alternative value is successfully read.
NOT_FOUND	Alternative value is not set for the specified field.
UNSUPPORTED	Invalid index.

WPW : Write Wiegand I/O Setting

Set Wiegand input function, output mode, and advanced options. Available values are as follows. Advanced options are not applicable to Pass Through mode.

Input function	
0x00	Disable
0x12	Enroll Given ID
0x22	Verify Given ID
0x42	Delete Given ID

Output mode	
0x00	Disable
0x01	Enable for Wiegand input only
0x02	Enable for all inputs

Advanced options	
0x01	When matching fails, inverse parity bits
0x02	When matching fails, output the fail ID set by WFW command

Request command

Field	Data	Description
Start code	0x40	
Command	0xC6	
Param	(Input function << 16) (Output mode)	

Size	Advanced options 0x01 0x02	Inverse parity on fail Fail ID
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC6	
Param	NULL	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Wiegand I/O is successfully configured..
UNSUPPORTED	Invalid setting value.

WPR : Read Wiegand I/O Setting

Read Wiegand I/O port settings configured by WPW command.

Request command

Field	Data	Description
Start code	0x40	
Command	0xC7	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xC7	
Param	(Input function << 16) (Output mode)	
Size	Advanced options	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	I/O settings are successfully read.

IW : Write Input Configuration

FM modules provide 3 input ports, IN0, IN1, and IN2. Users can assign an input function to each input port. If an input port is activated, the assigned function will be executed by the module.

Users can also assign tamper switch to an input port. When the tamper switch is on, Tamper Switch On(0x64) event occurred. When it gets off, Tamper Switch Off(0x65) event occurred. Both events are recorded in log, too.

The assignable functions for input ports are as follows.

Function ID	Description
0x00	No action
0x10	Enroll by Scan
0x30	Identify by Scan
0x40	Delete by Scan
0x49	Delete All
0x11	Enroll by Wiegand ID
0x21	Verify by Wiegand ID
0x41	Delete by Wiegand ID
0x1A	ES with Administrator's Verification
0x1B	EW with Administrator's Verification
0x4A	DS with Administrator's Verification
0x4B	DW with Administrator's Verification
0x4C	DA with Administrator's Verification
0x60	Cancel
0x66	Tamper Switch In
0x69	Reset Module

Request command

Field	Data	Description
Start code	0x40	
Command	0x47	
Param	Function ID	
Size	Minimum time	10 ~ 10000 msec If the input signal is shorter than this minimum time, it will be ignored.
Flag	0x00, 0x01, 0x02	IN0, IN1, IN2 port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x47	
Param	NULL	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Configuration of input port is successfully written.
UNSUPPORTED	Port ID is invalid.

Example

Assign Enroll by Scan(0x10) to IN1 with minimum time 100ms.

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x47	0x10	0x10	0x01	0xA8	0x0A

Byte transmission order

0x40, 0x47, 0x10, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x01, 0xA8, 0x0A

IR : Read Input Configuration

Reads configuration of an input port.

Request command

Field	Data	Description
Start code	0x40	
Command	0x48	
Param	NULL	
Size	NULL	
Flag	0x00, 0x01, 0x02	IN0, IN1, IN2 port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x48	
Param	Function ID	
Size	Minimum time	10 ~ 10000 msec
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Configuration of input port is successfully read.

UNSUPPORTED	Port ID is invalid.
-------------	---------------------

IG : Get Input State

Reads current state of input port.

Request command

Field	Data	Description
Start code	0x40	
Command	0x49	
Param	0 1	Clear state after read Remain state
Size	NULL	
Flag	0x00, 0x01, 0x02	IN0, IN1, IN2 port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x49	
Param	0 1	Inactive Active
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Current state of input port is successfully read.
UNSUPPORTED	Port ID is invalid.

OW : Write Output Configuration

FM modules provide 3 output ports and 3 LED ports; OUT0, OUT1, OUT2 and LED0, LED1, LED2 respectively. Users can assign multiple output events to each output or LED port. If one of the given events occurs, the configured signal will be output to the port.

The assignable events for output port are as follows.

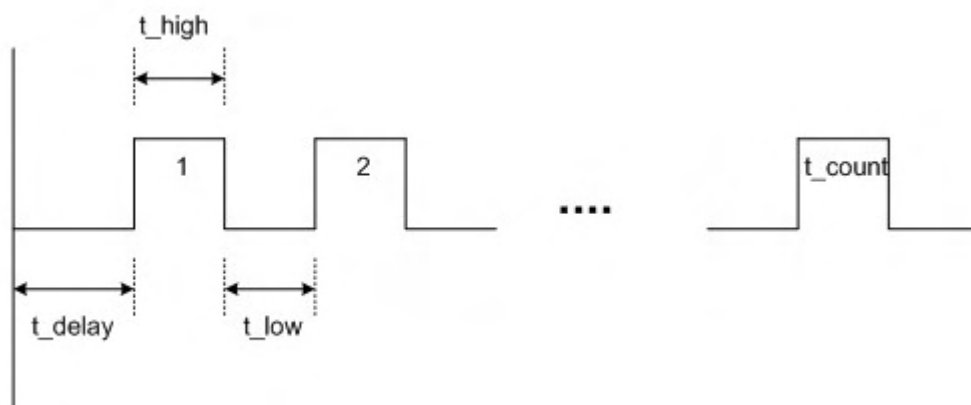
Category	Event ID	Description
Clear	0x00	Clear all functions
Enroll	0x13	Enroll Wait Wiegand
	0x14	Enroll Wait Finger
	0x15	Enroll Processing
	0x16	Enroll Bad Finger
	0x17	Enroll Success
	0x18	Enroll Fail
Verify	0x23	Verify Wait Wiegand
	0x24	Verify Wait Finger
	0x25	Verify Processing
	0x26	Verify Bad Finger
	0x27	Verify Success
	0x28	Verify Fail
	0x62	Verify Duress Finger
Identify	0x34	Identify Wait Finger
	0x35	Identify Processing
	0x36	Identify Bad Finger
	0x37	Identify Success
	0x38	Identify Fail
	0x63	Identify Duress Finger
Delete	0x43	Delete Wait Wiegand

	0x44	Delete Wait Finger
	0x45	Delete Processing
	0x46	Delete Bad Finger
	0x47	Delete Success
	0x48	Delete Fail
Detect	0x54	Detect IN0
	0x55	Detect IN1
	0x56	Detect IN2
	0x57	Detect Wiegand input
	0x58	Detect Finger
End	0x59	End Processing
Tamper Switch	0x64	Tamper Switch On
	0x65	Tamper Switch Off
System	0x6A	System Started
	0x6B	Reset by Watchdog Timer

Request command

Field	Data	Description
Start code	0x40	
Command	0x4A	
Param	(T_delay<<16) (T_count<<8) Event ID	If T_count is 0, the output signal will be repeated indefinitely until other events occur. If Event ID is 0, all the events assigned to this port will be cleared.
Size	(T_low<<16) (T_high)	If T_low and T_high are both 0, the specified event will be deleted.
Flag	0x00, 0x01, 0x02 0x03, 0x04, 0x05	OUT0, OUT1, OUT2 port LED0, LED1, LED2 port
Checksum	#	

End code	0x0A	
----------	------	--



Response command

Field	Data	Description
Start code	0x40	
Command	0x4A	
Param	NULL	
Size	NULL	
Error	0x61 0x6D 0x75	SUCCESS MEM_FULL UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Configuration of output port is successfully written.
MEM_FULL	Number of output configuration exceeds its limit (200 configurations).

UNSUPPORTED	Port ID is invalid.
-------------	---------------------

Example

(1) Assign Enroll Wait Finger(0x14) to LED0. The led will blink indefinitely with 500ms period; t_delay(0), t_high(250), t_low(250), t_count(0).

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x4A	0x14	0xFA00FA	0x03	0x95	0x0A

Byte transmission order

0x40, 0x4A, 0x14, 0x00, 0x00, 0x00, 0xFA, 0x00, 0xFA, 0x00, 0x03, 0x95, 0x0A

(2) Assign Enroll Success(0x17) to LED1. The led will be on for 500ms; t_delay(0), t_high(500), t_low(0), t_count(1).

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x4A	0x0117	0x01F4	0x04	0x9B	0x0A

Byte transmission order

0x40, 0x4A, 0x17, 0x01, 0x00, 0x00, 0xF4, 0x01, 0x00, 0x00, 0x04, 0x9B, 0x0A

OR : Read Output Configuration

Reads the configuration of an output event assigned to the given port. Users can get the list of events assigned to an output port using OL command.

Request command

Field	Data	Description
Start code	0x40	
Command	0x4B	
Param	Event ID	
Size	NULL	
Flag	0x00, 0x01, 0x02 0x03, 0x04, 0x05	OUT0, OUT1, OUT2 port LED0, LED1, LED2 port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x4B	
Param	(T_delay<<16) (T_count<<8)	
Size	(T_low<<16) (T_high)	
Error	0x61 0x69 0x75	SUCCESS NOT_FOUND UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Configuration of output port is successfully read.
NOT_FOUND	Specified event is not assigned to the specified port.
UNSUPPORTED	Port ID is invalid.

OL : Read Output Configuration List

Reads the list of events assigned to an output port.

Request command

Field	Data	Description
Start code	0x40	
Command	0x4C	
Param	NULL	
Size	NULL	
Flag	0x00, 0x01, 0x02 0x03, 0x04, 0x05	OUT0, OUT1, OUT2 port LED0, LED1, LED2 port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x4C	
Param	NULL	
Size	Number of enabled events	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

First transmits the response command, then enabled event IDs, and finally the 0x0A

Error code

Error code	Description
SUCCESS	Enabled events of output port are successfully read.
UNSUPPORTED	Port ID is invalid.

OS : Set Output State

Activate or deactivate specified output ports.

Request command

Field	Data	Description
Start code	0x40	
Command	0x4D	
Param	0x00 0x01	Deactivate Activate
Size	NULL	
Flag	0x00, 0x01, 0x02 0x03, 0x04, 0x05	OUT0, OUT1, OUT2 port LED0, LED1, LED2 port
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x4D	
Param	NULL	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Output is successfully completed.
UNSUPPORTED	Port ID is invalid.

GW : Write GPIO Configuration

Writes configuration of GPIO ports.

GW command should be followed by supplementary data and packet end code. Composition of supplementary data should be same as GR's response.

When configuring Wiegand interface, one command for one GPIO among the defined pair is enough for configuration.

In order to transfer the GPIO configuration between the module and the host, 8 byte supplementary data is used for each input function, output function, or wiegand configurations. 4 words (2 bytes) constitute the GPIO data, whose composition is as follows.

Word	Input	Output	Wiegand
1	Function	Function	Bits
2	Activation	Level	Facility bits
3	0 / interval	Interval	ID bits
4	0	Blink period	Facility code

Third and fourth words in input mode are useless, except the “delete all by confirm” function, which requires third word for timeout interval.

Refer to Appendix A. for detail description about functions.

For WAIT and PROCESSING output functions, interval and blinking is meaningless, but the interval value should be positive value for effective configuration.

Request command

Field	Data	Description
Start code	0x40	
Command	0x37	
Param	GPIO ID (mode << 16)	

Size	# of supplementary data	
Flag	NULL	
Checksum	#	
End code	0x0A	

First transmits the request command, then supplementary, and finally the 0x0A

Response command

Field	Data	Description
Start code	0x40	
Command	0x37	
Param	GPIO ID	
Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Configuration of GPIO port is successfully written.
UNSUPPORTED	Supplementary data is invalid.

Example

When setting GPIO_0(0x0) as an input port(0x01) for enroll function(0x01) with active

high activation(0x01),

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x37	0x010000	0x01	0x00	0x79	0x0A

+

Function(0x01)	Activation(0x01)	0	0	0x0A
----------------	------------------	---	---	------

The Param(GPIO ID | (mode<<16)):

Upper 16 bit for Mode	Lower 16 bit for GPIO ID
0x0001	0x0000

The 8 byte supplementary data:

1 st word for Function	2 nd word for Activation	3 rd word	4 th word
0x0001	0x0001	0x0000	0x0000

Byte transmission order

0x40, 0x37, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x79, 0x0A

0x01, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0A

When setting GPIO_7(0x7) as an output port(0x02) for Beep function(0x8D) with active low activation(0x84) and 100ms interval,

Start	Command	Param	Size	Flag	Checksum	End
0x40	0x37	0x020007	0x01	0x00	0x81	0x0A

+

Function(0x8D)	Level(0x84)	Interval(0x0a)	Blink Period(0x0)	0x0A
----------------	-------------	----------------	-------------------	------

The Param(GPIO ID | (mode<<16)):

Upper 16 bit for Mode	Lower 16 bit for GPIO ID
0x0002	0x0007

The 8 byte supplementary data:

1 st word for Function	2 nd word for Level	3 rd word for Interval	4 th word for Blinking

0x008D	0x0084	0x000a	0x0000
--------	--------	--------	--------

Byte transmission order

0x40, 0x37, 0x07, 0x00, 0x02, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x81, 0x0A

0x8D, 0x00, 0x84, 0x00, 0x0A, 0x00, 0x00, 0x00, 0x0A

GR : Read GPIO Configuration

Reads configuration of GPIO ports.

The response packet is followed by supplementary data (8 * size) + packet end code (0x0A) if mode is not equal to disabled mode. The number of supplementary data is always 1 for the input mode and it is the number of active output functions for the output mode, respectively. For the shared I/O mode, first supplementary data is input function.

Request command

Field	Data	Description
Start code	0x40	
Command	0x36	
Param	GPIO ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x36	
Param	GPIO ID (mode << 16)	
Size	# of supplementary data	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	

End code	0x0A	
----------	------	--

First transmits the response command, then supplementary data, and finally the 0x0A.

Error code

Error code	Description
SUCCESS	Configuration of GPIO port is successfully read.
UNSUPPORTED	Supplementary data is invalid.

Compatibility

NOT FOR FM

GC : Clear GPIO Configuration

Clears all configurations of GPIO ports.

Request command

Field	Data	Description
Start code	0x40	
Command	0x38	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x38	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	All configurations of GPIO is successfully cleared.

Compatibility NOT FOR

FM

GD : Set Default GPIO Configuration

Sets default of all GPIO ports.

Request command

Field	Data	Description
Start code	0x40	
Command	0x39	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x39	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Successfully completed.

Compatibility

NOT FOR FM

AW : Write Administration Level of a User

Assigns an administration level to a user. The default administration level of a user is ADMIN_NONE except when the first user is enrolled by ESA or EWA. In that case, the first user will have ADMIN_ALL level. See ESA for the relationship between administration levels and commands with administrator's verification.

Admin. Level	Value	Commands
ADMIN_NONE	0x00	
ADMIN_ENROLL	0x01	ESA, EWA
ADMIN_DELETE	0x02	DAA, DSA, DWA
ADMIN_ALL	0x03	ESA, EWA, DAA, DSA, DWA

Request command

Field	Data	Description
Start code	0x40	
Command	0x65	
Param	User ID	
Size	Administration Level	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x65	
Param	NULL	

Size	NULL	
Error	0x61 0x69 0x75 0x76	SUCCESS NOT_FOUND UNSUPPORTED INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Administration level is successfully written.
NOT_FOUND	The user id is not found.
UNSUPPORTED	The administration level is not supported.
INVALID_ID	The user id is invalid.

AR : Read Administration Level of a User

Reads the administration level of a user.

Request command

Field	Data	Description
Start code	0x40	
Command	0x66	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x66	
Param	NULL	
Size	Administration Level	
Error	0x61	SUCCESS
	0x69	NOT_FOUND
	0x76	INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Administration level is successfully read.
NOT_FOUND	The user id is not found.
INVALID_ID	The user id is invalid.

AC : Clear All the Administration Levels

Clears all the administration levels to ADMIN_NONE.

Request command

Field	Data	Description
Start code	0x40	
Command	0x67	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0x67	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Administration levels are successfully cleared.

UW: Write Authentication Mode of a User

Specifies the authentication mode of a user. AUTH_BYPASS can be used for 1:1 matching, when it is necessary to allow access without matching fingerprints. AUTH_REJECT can be used for disabling some IDs temporarily. The default authentication mode is AUTH_FINGERPRINT.

Mode	Value	Description
AUTH_FINGERPRINT	0x00	Fingerprint authentication.
AUTH_BYPASS	0x01	Authentication will succeed without matching fingerprints.
AUTH_REJECT	0x03	Authentication will always fail.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA3	
Param	User ID	
Size	Authentication Mode	AUTH_FINGERPRINT AUTH_BYPASS AUTH_REJECT
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA3	

Param	NULL	
Size	NULL	
Error	0x61 0x69 0x75 0x76	SUCCESS NOT_FOUND UNSUPPORTED INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Authentication mode is successfully written.
NOT_FOUND	The user id is not found.
UNSUPPORTED	The authentication mode is invalid.
INVALID_ID	The user id is invalid.

UR : Read Authentication Mode of a User

Reads the authentication mode of a user.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA4	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA4	
Param	NULL	
Size	Authentication Mode	
Error	0x61	SUCCESS
	0x69	NOT_FOUND
	0x76	INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Authentication mode is successfully read.
NOT_FOUND	The user id is not found.
INVALID_ID	The user id is invalid.

UC : Reset Authentication Modes of All Users

Resets the authentication modes of all users to AUTH_FINGERPRINT.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA5	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA5	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Authentication modes are successfully reset.

UL : List User ID by Authentication Mode

Reads the user IDs with specified authentication mode.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA6	
Param	NULL	
Size	Authentication mode	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA6	
Param	Number of user ID	Number of user IDs with the specified authentication mode
Size	Data size	(The number of user ID) * 4
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

First transmits the response command and then the user IDs.

User ID (4byte)	User ID (4byte)	...	User ID (4byte)	User ID (4byte)	0x0A
--------------------	--------------------	-----	--------------------	--------------------	------

Error code

Error code	Description
SUCCESS	Successfully completed.

ABL : Add a User to the Blacklist

Adds a user to the blacklist. When a user ID is added to the blacklist, authentication will always fail regardless of fingerprint matching result. The blacklist takes precedence over the authentication mode of a user. For example, though the authentication mode of a user is AUTH_BYPASS, the authentication would fail if it is in the blacklist. The blacklist can store up to 1022 user IDs.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF3	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF3	
Param	User ID	
Size	Number of IDs in the blacklist	
Error	0x61 0x6D	SUCCESS MEM_FULL
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The id is added successfully.
MEM_FULL	The blacklist is full.

DBL : Delete a User ID from the Blacklist

Deletes a user from the blacklist.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF4	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF4	
Param	User ID	
Size	Number of IDs in the blacklist	
Error	0x61 0x69	SUCCESS NOT_FOUND
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The user id is deleted successfully.

NOT_FOUND	The user id is not found in the blacklist.
-----------	--

RBL : Read the Blacklist

Reads the contents of the blacklist.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF5	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF5	
Param	Number of user ID	Number of user IDs in the blacklist
Size	Data size	(The number of user ID) * 4
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

First transmits the response command and then the user IDs.

User ID (4byte)	User ID (4byte)	...	User ID (4byte)	User ID (4byte)	0x0A
--------------------	--------------------	-----	--------------------	--------------------	------

Error code

Error code	Description
SUCCESS	Successfully completed.

CBL : Clear the Blacklist

Clears the blacklist.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF6	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF6	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The blacklist is cleared successfully.

WME : Write Entrance Limit

Specifies how many times the user is permitted to access per day. The available options are between 0 and 7. The default value is 0, which means that there is no limit. If the user tries to authenticate after the limit is reached, ENTRANCE_LIMIT error will be returned.

To check the limit, the module keeps track of entrance counts of each user. The entrance counts are cleared in the following cases.

- (1) At midnight.
- (2) When the module is reset.
- (3) When new entrance limit is written by WME command.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF0	
Param	User ID	
Size	Entrance limit	0(Infinite) 1 ~ 7
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF0	

Param	NULL	
Size	NULL	
Error	0x61 0x69 0x75 0x76	SUCCESS NOT_FOUND UNSUPPORTED INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Entrance limit is successfully written.
NOT_FOUND	The user id is not found.
UNSUPPORTED	The entrance limit is invalid.
INVALID_ID	The user id is invalid.

RME : Read Entrance Limit

Reads the entrance limit of a user.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF1	
Param	User ID	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF1	
Param	Entrance count	The number of entrances for today
Size	Entrance limit	The entrance limit written by WME
Error	0x61 0x69 0x76	SUCCESS NOT_FOUND INVALID_ID
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Entrance limit is successfully read.
NOT_FOUND	The user id is not found.
INVALID_ID	The user id is invalid.

CME : Clear Entrance Limit

Clears the entrance limits of all users.

Request command

Field	Data	Description
Start code	0x40	
Command	0xF2	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xF2	
Param	NULL	
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	The entrance limits of all users are cleared.

Appendix A. GPIO Configuration

Overview

NOT FOR FM MODULE

Because all GPIO function can be configured using PC interface program, a general user does not need to know the method stated in this appendix. After GPIO function is tested using PC interface program, the specific configuration can be embedded in the module before delivery for volume order.

GPIO's can be configured as input, output, shared IO, Wiegand input, or Wiegand output. In the input mode, enroll, identify, and delete function are supported. In the output mode, the port can send requested output patterns corresponding to the various events, such as match success, enroll fail, and so on.

1. GPIO Index

Eight GPIO's are available on FM modules, which are configurable. Each port is indexed by its ID ranged from 0 to 7 as described in the Table 1.

GPIO Name	ID
GPIO_0	0
GPIO_1	1
GPIO_2	2
GPIO_3	3
GPIO_4	4

GPIO_5	5
GPIO_6	6
GPIO_7	7

Table 1. GPIO index

Supported pin modes are input, output, shared IO, wiegand input and wiegand output. GPIO of shared IO mode waits input events in normal state and also sends responding output. Input modes, including general input and shared IO, are configurable on GPIO_0 to GPIO_3. Output mode is configurable on any GPIO port. Wiegand modes are restricted on special ports. Wiegand output mode is only configurable on the couple of GPIO_4 and GPIO_5, and wiegand input is configurable on the couple of GPIO_2 and GPIO_3, respectively. Port of lower ID is correspondent to bit 0 and port of higher ID is correspondent to bit 1 for Wiegand interfaces.

When the module is operated in network mode, GPIO_7 is internally controlled to support the network interfaces of RS422 or RS485. So, GPIO_7 is not available for general purpose in network mode.

Supported modes for the GPIO's are summarized in Table 2.

Ports	Output	Input	Shared IO	Wiegand In	Wiegand Out
GPIO_0 ~ 1	O	O	O		
GPIO_2 ~ 3	O	O	O	O	
GPIO_4 ~ 5	O				O
GPIO_6 ~ 7	O				

Table 2. Supported modes

2. GPIO mode

Configured mode of GPIO is identified by its mode value. GPIO mode value is listed in Table 3.

Mode	Value
None(disable)	0
Input	1
Output	2
Shared IO	3
Wiegand input	4
Wiegand output	5

Table 3. Gpio mode value

Wiegand modes are enabled for the specific couples. And, if one of Wiegand couple is re-configured as other mode, then the other one is automatically reset to disabled mode.

1) Input mode

Input mode is defined by two factors, function and activation method. GPIO configured as input mode processes one of input functions, enrollment, identification, delete all, and delete all by confirm when the module detects activation at the pin. Four activation methods are selectable including active high, active low, falling edge, and rising edge activations.

Table 4 and 5 shows the index value for input function and activation method.

Input function	Corresponding command	Index value
Enroll	ES	0x01
Identify	IS	0x02
Delete all	DA	0x03
Delete all by confirm	DA	0x04
Cancel	CA	0x06
Enroll with admin's verification	ESA	0x07
Delete with admin's verification	DSA	0x08

Delete all with admin's verification	DAA	0x09
Reset	RS	0x0A

Table 4. Input functions

Activation	Index value
Active high	0x01
Active low	0x02
Rising edge	0x03
Falling edge	0x04

Table 5. Input activation method

“Delete all by confirm” function is for erasing all fingerprint data after confirmation by accepting repeated input within the pre-specified interval. So, this function requires another parameter of maximum waiting interval. This function is also related with a specific output function, named DELETE_WAIT, which sends blinking output in waiting repeated input.

2) Output mode

The GPIO in the output mode can support various output patterns for different functions. For example, GPIO_0 can be configured to send 1 pulse on enroll success and 2 pulses on match success.

There are four issues for output configuration: When, Which level, How long, and steady or blinking. In order to support these requirements, output function, level, interval, and blinking period can be configured.

There are twelve selectable output functions, which are listed in Table 6.

Event Name	Description	Event ID
ENROLL_WAIT_FINGER	Waiting finger scan on enroll	0x81
ENROLL_PROCESSING	Enroll process is going on	0x82
ENROLL_SUCCESS	Enroll success	0x83
ENROLL_FAIL	Enroll fail	0x84
MATCH_WAIT_FINGER	Waiting finger scan on match	0x85
MATCH_PROCESSING	Match process is going on	0x86
MATCH_SUCCESS	Match success	0x87
MATCH_FAIL	Match fail	0x88
DELETE_WAIT	Waiting confirm on 'delete all by confirm' input	0x89
DELETE_PROCESSING	Delete processing is going on	0x8a
DELETE_SUCCESS	Delete success	0x8b
DELETE_FAIL	Delete fail	0x8c
BEEP	Beep function	0x8d

Table 6 Supported output functions

BEEP function is special function with fixed output pattern on complex output events, which is useful for buzzer control. It will respond on various events with pre-defined patterns as follows:

- End of finger scan : 1 short beep
- Processing success : 1 long beep
- Insufficient fingerprint data : 2 beeps
- Processing failure : 3 beeps

Selectable output levels are listed in Table 7.

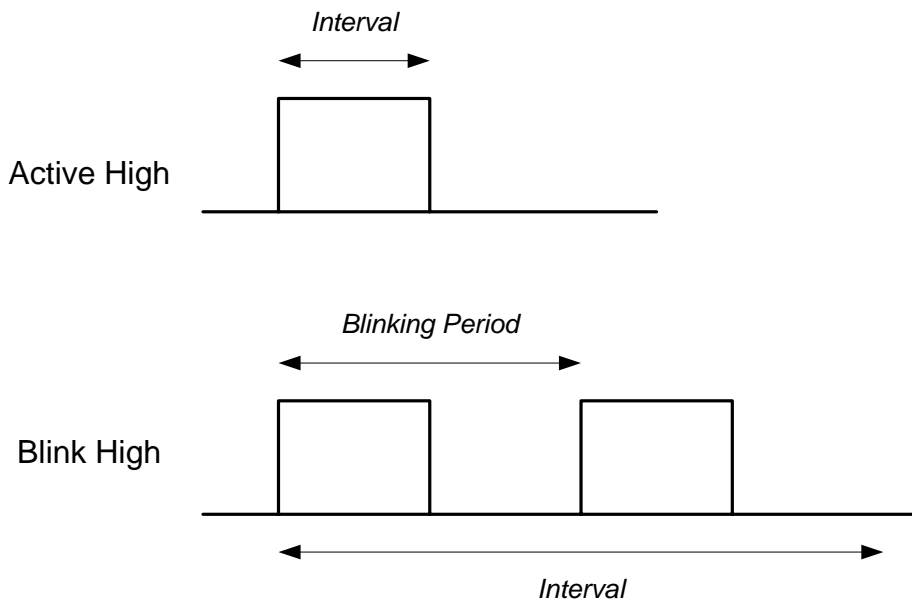
Name	Description	Index value
ACTIVE_HIGH	Active high level	0x82

ACTIVE_LOW	Active low level	0x84
HIGH_BLINK	Active high + blinking	0x83
LOW_BLINK	Active low + blinking	0x85

Table 7. Output levels

Notice that all the enabled functions for the same GPIO should have same inactive level, all high or all low. That is, if the inactive level is low, then all should be configured as ACTIVE_HIGH or HIGH_BLINK.

Output interval is also configurable. If the output is blinking mode, blinking period should be specified. Time units for these values are 10ms and the range is 1 to 0xFFFF. Notice that blinking period should not exceed the interval time. Blink count is obtained through dividing interval by blinking period. The following figure shows examples of output signals.



3) Wiegand mode

Wiegand mode requires four configuration parameters: total bits, facility bits, id bits, and facility code. Currently, the firmware only supports standard 26 bit Wiegand interfaces. When identification or verification is successful, Wiegand output is generated combining facility code and user ID. Similarly, Wiegand input is applied, the module carries out verification process

(VS) based on the ID code.

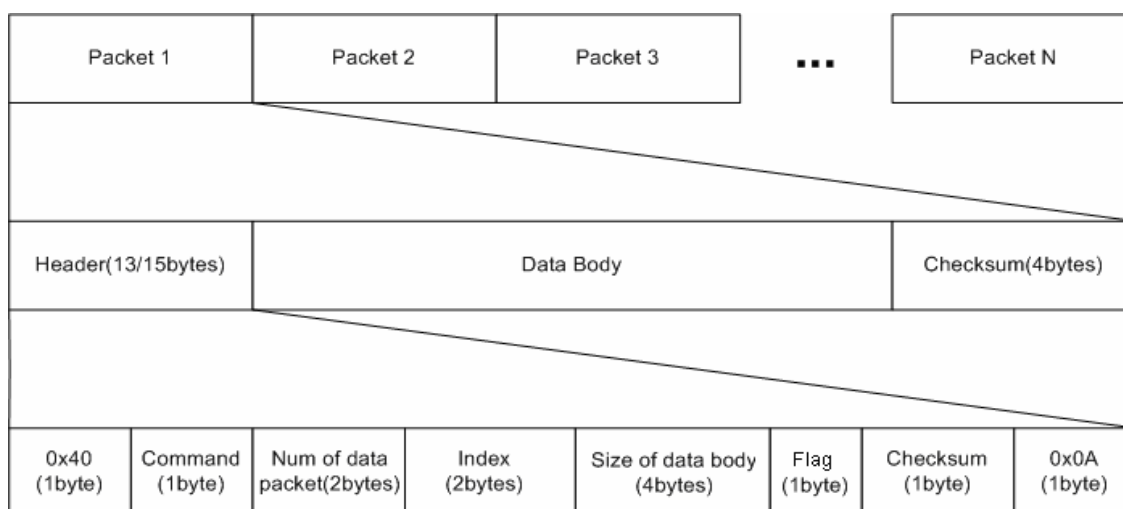
Appendix B. Extended Data Transfer Protocol

Overview

Dividing large data into small blocks can reduce communication errors between the host and the module. It is also suitable for embedded systems with limited resources. Data Transfer Protocol is an extension of Packet Protocol to provide a reliable and customizable communication for large data.

Data Packet Protocol

In Data Transfer Protocol, data is divided into multiple data packets. And a data packet consists of fixed-length header, variable-length data body, and 4 byte checksum. The following figure shows the data packet structure in Packet Protocol. If Network Packet Protocol is used, the start code will be 0x41, and 2 byte terminal ID will be inserted between the start code and the command byte. The checksum field is the sum of each byte in data body.



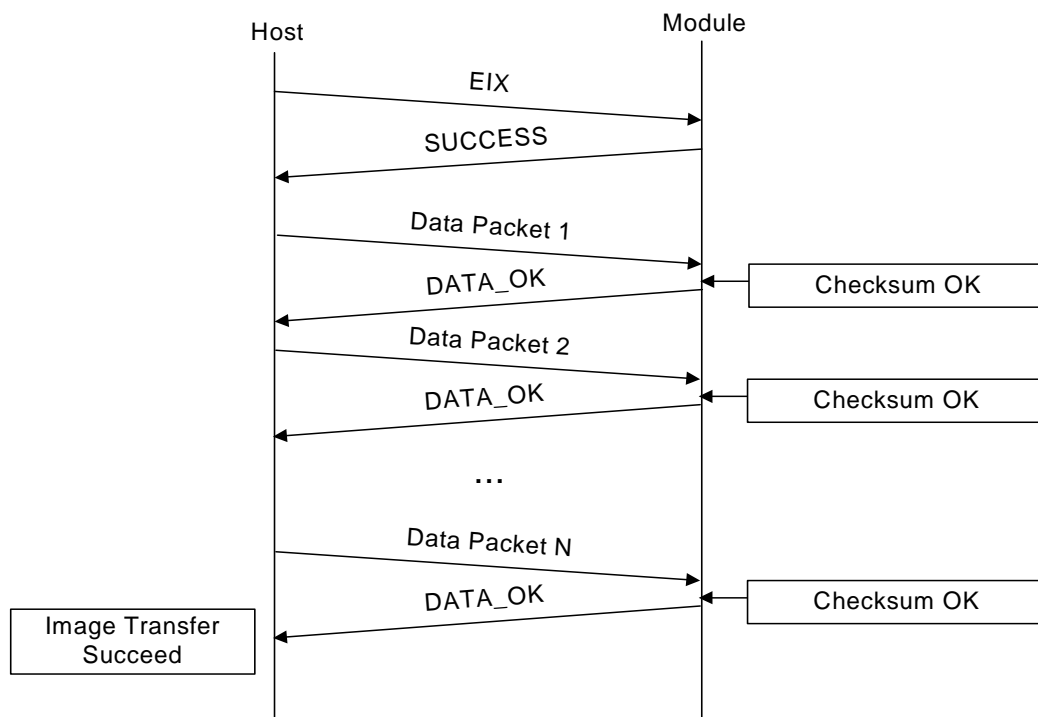
For example, if EIX command sends an image data of 16384 bytes into 3 packets of 4096, 4096, and 8192 bytes, the headers are as follows:

	Start	Command	Num of Packet	Index	Data Size	Flag	Checksum	End
Packet 1	0x40	0x80	0x03	0x00	0x1000	0x00	0xD3	0x0A
Packet 2	0x40	0x80	0x03	0x01	0x1000	0x00	0xD4	0x0A
Packet 3	0x40	0x80	0x03	0x02	0x2000	0x00	0xE5	0x0A

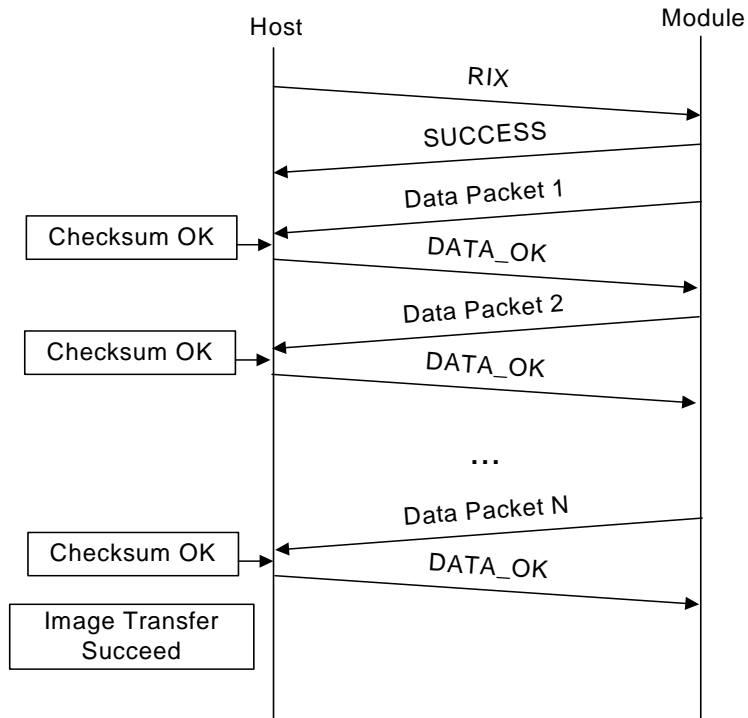
After receiving each data packet, the receiver has to send an acknowledgement packet notifying if the previous transfer is successful. If the received data packet has correct checksum, DATA_OK(0x83) will be returned. Otherwise, DATA_ERROR(0x82) will be returned. If the sender receives DATA_ERROR, it should quit sending the data immediately.

Example

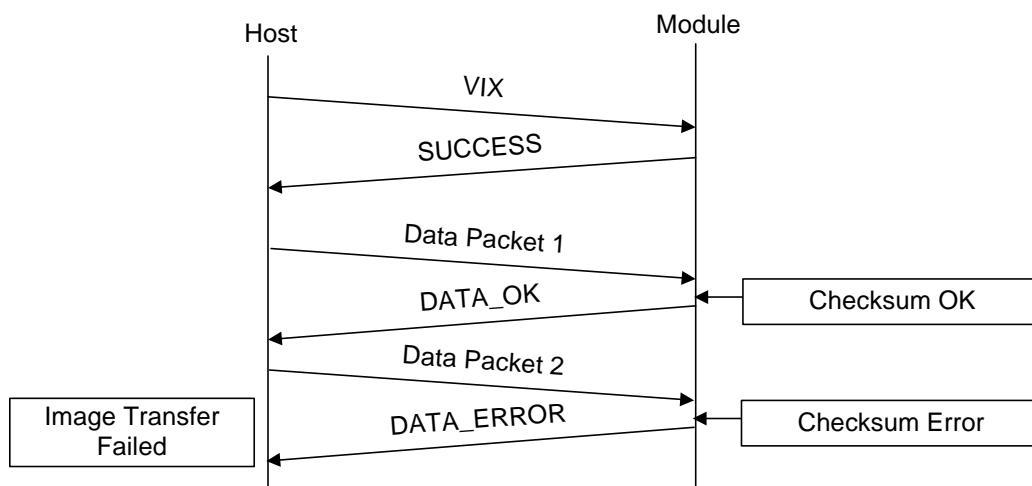
(1) When EIX succeeds in sending an image,



(2) When RIX succeeds in receiving an image,



(3) When VIX fails,



Appendix C. Extended Wiegand Protocol

Overview

Extended Wiegand Protocol supports up to 64 bit Wiegand formats. The only constraint is that the ID field is limited to 32 bits. It also supports advanced options such as Fail ID and Inverse Parity on Fail. There are three ways of configuring Wiegand formats; 26 bit standard, Pass Through, and Custom format.

26 bit standard

The 26 bit standard format is most widely used and consists of 8 bit site code and 16 bit ID. Users can set an alternative site code and enable advanced options.

Pass Through format

Pass Through format is used when only the format of ID field is known. When Wiegand input string is detected, the module extracts ID bits and starts verification with the ID. If the verification succeeds, the module outputs the Wiegand input string as unchanged. Parity check and advanced options are ignored in this format. By definition, Pass Through format is only useful when the matching is initiated by Wiegand input. If the matching is initiated by Packet Protocol or GPIO input, the bits other than ID field are set to 0.

The definition data for Pass Through format is as follows;

Field	Byte	Description
Total bits	1	1 ~ 64
Number of ID fields	1	In most cases, it will be 1. However, if ID bits are composed of several fields, it would be larger than 1.
1 st ID field start bit	1	Start bit index of 1 st ID field
1 st ID field length	1	Number of 1 st ID bits

...		
Nth ID field start bit	1	Start bit index of Nth ID field
Nth ID field length	1	Number of Nth ID bits

For example, assume that 32 bit pass through format is composed as follows;

XXXXXXXX IXXXXXXXX XXXXXXXX IXXXXXXXX (left most bit is 0th bit, BIT0)

I: Id field, X: Unknown field

The definition data would be,

Total Bits	Num of ID fields	1 st ID field start index	1 st ID field length	2 nd ID field start index	2 nd ID field length
0x20	0x02	0x01	0x0E	0x13	0x0C

Custom format

When users know all the information of a Wiegand format, Custom format can be defined. When Wiegand input string is detected, the module checks the parity bits first. If all the parity bits are correct, the module extracts ID bits and starts verification with the ID. Users can also set alternative values of each field and enable advanced options such as Fail ID. If the verification succeeds, the module outputs a Wiegand string. The output string may be different from the input string according to the alternative values and advanced options.

The definition data of Custom format is as follows;

Field		Byte	Description
Total Bits		1	1 ~ 64
Number of fields		1	1 ~ 16
1 st Field	Start bit index	1	Start bit index of the field
	Length	1	Number of bits: 1 ~ 32
...			
Nth Field	Start bit index	1	Start bit index of the field
	Length	1	Number of bits: 1 ~ 32

Number of parity bits		1	1 ~ 8
1 st Parity	Type	1	0: Even, 1: Odd
	Bit position	1	Bit index of the parity bit
	Bit mask	8	Bit mask for identifying the bits which are included in calculating the parity
...			
Nth Parity	Type	1	0: Even, 1: Odd
	Bit position	1	Bit index of the parity bit
	Bit mask	8	Bit mask for identifying the bits which are included in calculating the parity
Bit mask for ID field		2	Bit mask for identifying the fields which are part of ID

For example, assume that 44 bit custom format is composed as follows;

EAAAAAAAA IIIIIIII IIIIIIII BBBBBI IIIIII IIO (left most bit is 0th bit, BIT0)

E: Even parity for BIT1 ~ BIT22

O: Odd parity for BIT23 ~ BIT42

I: ID bits(Field1 and Field 3), A: Field 0, B: Field 2

The field definition data would be,

Field	Byte	Value
Total Bits	1	0x2C
Number of fields	1	0x04
Field 0	Start bit index	1
	Length	1
Field 1	Start bit index	1
	Length	1
Field 2	Start bit index	1
	Length	1

Field 3	Start bit index	1	0x1F
	Length	1	0x0C
Number of parity bits		1	0x02
1 st Parity	Type	1	0x00: Even
	Bit position	1	0x00
	Bit mask	8	0x7FFFFE: Bit1 ~ Bit 22
2 nd Parity	Type	1	0x01: Odd
	Bit position	1	0x2B
	Bit mask	8	0x07FFFF800000: Bit 23 ~ Bit 42
Bit mask for ID field		2	0x0A: Field 1 and Field 3

Appendix D. Packet Protocol for BioEntry™

Overview of BioEntry™

BioEntry™ is an advanced biometric access reader equipped with fingerprint recognition engine and standard Wiegand interface. BioEntry™ can practically replace legacy and simple readers and be instantly added onto existing access control systems as well as new installations.

BioEntry™ Smart is a fingerprint smart card reader that seamlessly integrates fingerprint and smart card reader into one device. BioEntry™ Smart is designed to replace existing access readers like proximity or magnetic readers without additional wiring. Fingerprint template is stored in each user's smart card and there is no need to store fingerprint data in a reader itself. This eliminates the burden of template management and networking readers.

BioEntry™ Pass is a fingerprint access reader equipped with fast one to many fingerprint identification engine. Enrolled with more than hundreds of users, identification can be done in less than one second.

FM Module vs. BioEntry™ (BioPass)

BioEntry™ is based on FM modules and shares most of the packet protocol commands.

Aside from the commands added for BioEntry™ Smart, there are following differences between the two.

(1) System Parameters

- To prevent ID collision in network environment, a unique Module ID is assigned to each BioEntry™ reader. Users cannot change the Module ID(0x6D) system parameter.
- Firmware Version(0x6E) parameter starts with 'P' for BioEntry™ Pass and 'S' for BioEntry™ Smart.
- Auxiliary port is reserved for Smartcard functionality. Therefore, users should not change the Baudrate2(0x72) system parameter.

(2) IO Configurations

- Input ports: IN2 is reserved for Tamper Switch and cannot be configured otherwise.
- Output ports: OUT2 is reserved for internal use and cannot be configured otherwise.
- LED ports: LED0 is connected to green LED. LED1 is connected to red LED. When both LED0 and LED1 are on, the LED on the reader will be seen as yellow. LED2 port is connected to an internal speaker and used for beep sound.

(3) Miscellaneous

- CARD_ERROR(0xA0) error code is added for BioEntry™ Smart.
- ACCESS_NOT_GRANTED(0x93) error code is added.
- The following output events are added for BioEntry™ Smart.
 - DETECT_SMARTCARD(0x70)
 - BAD_SMARTCARD(0x71)
 - WAIT_SMARTCARD(0x72)

Commands for BioEntry™ Smart

The following commands are added to BioEntry™ Smart.

Category	Name	Code	Description
Card Management	CR	0xA0	Read templates from a Smartcard
	CW	0xA1	Write templates to a Smartcard
	CF	0xAE	Format a Smartcard
	CC	0xA2	Configure the input function of Smartcard
	CG	0xA8	Get the input function of Smartcard
	VC	0xA7	Verify by Smartcard
	ECX	0xAF	Enroll templates to a Smartcard
Site Key & Layout Management	CKW	0xAA	Write site key
	CKR	0xAB	Read key options
	CLW	0xAD	Write card layout
	CLR	0xAC	Read card layout

CR : Read Smartcard

Reads the contents of a Smartcard. The contents consist of a 22 byte header and one or two template data. The structure of the header is as follows.

Field	Size	Description
CSN	4 bytes	Card serial number
Wiegand String	8 bytes	Wiegand string is converted into a 64 bit integer before written to a Smartcard. For example, assume that a 26 bit standard Wiegand string is 0000000000000000000010101 – facility code is 0 and the ID is 10. It can be converted to 0x0000000000000015 in 64 bit integer format. Then, the 64 bit integer will be written into the Smartcard in little-endian order.
Version	1 byte	Version of the Smartcard
Command Type	1 byte	See BioEntry Operation Manual for types of command cards
Security Level	1 byte	0: same as the security level set in the reader 1: 1/10,000 2: 3/100,000 3: 1/100,000 4: 3/1,000,000 5: 1/1,000,000 6: 3/10,000,000 7: 1/10,000,000 8: 3/100,000,000 9: 1/100,000,000 10: bypass. Pass this user without authentication of fingerprints.
Number of Templates	1 byte	Number of enrolled templates(0 ~ 2)
Template 1 Duress	1 byte	1: template 1 is of a duress finger 0: template 1 is not of a duress finger

Template 1 Size	1 byte	The size of template 1
Template 2 Duress	1 byte	1: template 2 is of a duress finger 0: template 2 is not of a duress finger
Template 2 Size	1 byte	The size of template 2

Request command

Field	Data	Description
Start code	0x40	
Command	0xA0	
Param	NULL	
Size	NULL	
Flag	Bit mask for data to be read 0x01: Header 0x02: Template 1 0x04: Template 2	For example, to read the header and template 1 only, the Flag should be 0x03. If it is 0, it is same as 0x01.
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA0	
Param	NULL	
Size	Size of data to be returned	
Error	0x61 0x6C 0xA0	SUCCESS TIMEOUT CARD_ERROR
Checksum	#	
End code	0x0A	

First transmits the response command, then card data, and finally the 0x0A.

Error code

Error code	Description
SUCCESS	Smartcard is successfully read.
TIMEOUT	Smartcard hasn't been detected for the timeout period.
CARD_ERROR	There is an error in reading the Smartcard.

CW : Write Smartcard

Writes templates to a Smartcard.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA1	
Param	Template index	0 for the first template and 1 for the second template
Size	Template size	
Flag	NULL 0x01	Normal finger Duress finger
Checksum	#	
End code	0x0A	

First transmits the request command, then template data, and finally the 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0xA1	
Param	Template index	
Size	NULL	
Error	0x61 0x6B 0x6C 0x75 0xA0	SUCCESS TRY_AGAIN TIMEOUT UNSUPPORTED CARD_ERROR

Checksum	#	
End code	0x0A	

First transmits the response command, then supplementary data, and finally the 0x0A.

Error code

Error code	Description
SUCCESS	Template is written successfully.
TRY_AGAIN	Template data is not received.
TIMEOUT	Smartcard hasn't been detected for the timeout period.
UNSUPPORTED	Invalid template index.
CARD_ERROR	There is an error in writing templates to the Smartcard.

CF : Format Smartcard

Formats a Smartcard. Formatting clears the header and nullifies the template data.

Request command

Field	Data	Description
Start code	0x40	
Command	0xAE	
Param	NULL	
Size	NULL	
Flag	0x00 0x01	Format header and template data. Format template data only.
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xAE	
Param	NULL	
Size	NULL	
Error	0x61 0x6C 0xA0	SUCCESS TIMEOUT CARD_ERROR
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Smartcard is formatted successfully.
TIMEOUT	Smartcard hasn't been detected for the timeout period.
CARD_ERROR	There is an error in formatting the Smartcard.

CC : Configure Card Input Function

Configures the input function which will be executed whenever a Smartcard is detected. There are three input functions available.

Function	Description
Disable	Do nothing.
Verify ID only	Ignore the templates on the card and try to match with the templates enrolled with the ID in the BioEntry reader.
Verify Template	Try to match with the templates enrolled in the card.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA2	
Param	NULL	
Size	NULL	
Flag	0x00 0x01 0x02	Disable Verify ID only Verify Template
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA2	
Param	NULL	

Size	NULL	
Error	0x61 0x75	SUCCESS UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Input function is configured successfully.
UNSUPPORTED	Invalid input function.

CG : Get Card Input Function

Reads the input function of Smartcard.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA8	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA8	
Param	NULL	
Size	Input function	0x00 – Disable 0x01 – Verify ID only 0x02 – Verify Template
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
------------	-------------

SUCCESS	Input function is read successfully.
---------	--------------------------------------

VC : Verify by Smartcard

Verifies if a fingerprint input on the sensor matches the templates in a Smartcard.

Request command

Field	Data	Description
Start code	0x40	
Command	0xA7	
Param	NULL	
Size	NULL	
Flag	0x00 0x01	Verify Template Verify ID only
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xA7	
Param	User ID	
Size	NULL	
Error	0x61 (0x62) 0x6A 0x6B 0x6C 0x75 0xA0	SUCCESS (SCAN_SUCCESS) NOT_MATCH TRY_AGAIN TIMEOUT UNSUPPORTED CARD_ERROR

Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Verification succeeds.
SCAN_SUCCESS	Smartcard or fingerprint is detected successfully.
NOT_MATCH	Verification failed.
TRY_AGAIN	Fingerprint image is not good.
TIMEOUT	Smartcard or fingerprint has not been detected for the timeout period.
UNSUPPORTED	Invalid option.
CARD_ERROR	There is an error in reading the Smartcard.

ECX : Enroll Templates to Smartcard

Enrolls templates to a Smartcard. It also initializes the header data on the card. A Smartcard should be enrolled first by this command to be detected by BioEntry readers. The transfer of template data conforms to Data Transfer Protocol.

Request command

Field	Data	Description
Start code	0x40	
Command	0xAF	
Param	User ID	
Size	(Security level << 24) (Number of templates << 16) Template size	See CR command for available security levels
Flag	Bit mask 0x01: Template 1 is duress 0x02: Template 2 is duress 0x04: Use a Wiegand string directly	If 0x04 is set, 8 byte Wiegand string should be sent after template data. In this case, User ID is ignored.
Checksum	#	
End code	0x0A	

After the request command, the following data should be sent to the module by Extended Data Transfer Protocol.

- (1) Template data: Template size * Number of templates
- (2) Wiegand string: If 0x04 is set in the Flag, 8 byte Wiegand string should be sent.

Response command

Field	Data	Description
-------	------	-------------

Start code	0x40	
Command	0xAF	
Param	User ID	
Size	CSN	Card Serial Number
Error	0x61 0xA0 0x76 0x6B 0x6C 0x75 0x82 0x83	SUCCESS CARD_ERROR INVALID_ID TRY_AGAIN TIMEOUT UNSUPPORTED DATA_ERROR DATA_OK
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Templates are enrolled successfully.
CARD_ERROR	There is an error in writing the Smartcard.
INVALID_ID	User ID is invalid.
TRY_AGAIN	Invalid template data.
TIMEOUT	Smartcard has not been detected for the timeout period.
UNSUPPORTED	Invalid parameters or options.
DATA_ERROR	The checksum of a data packet is incorrect.
DATA_OK	The checksum of a data packet is correct.

CKW : Write Site Key

To prevent unauthorized access, Smartcards are encrypted with a 48 bit site key. For a BioEntry reader to decrypt a Smartcard, the site key stored in the reader should match with that of the card. Users can store as many as two site keys in the BioEntry reader and select two advanced options. If Use Secondary Key option is selected, the reader will try both the primary and secondary keys when decrypting a Smartcard. If it is not selected, the reader will try only the primary key. Auto Update option is useful when changing the keys of Smartcards. With this option on, the reader will re-encrypt a Smartcard with the primary key when detecting one which is encrypted with the secondary key.

Request command

Field	Data	Description
Start code	0x40	
Command	0xAA	
Param	Key type 0x00 0x01 0x02	Change the options only. Change the primary key. Change the secondary key.
Size	Data size	If Key type is 0, it will be 8. Otherwise, it will be 16.
Flag	Bit mask for options 0x01: Use Secondary Key 0x02: Auto Update	
Checksum	#	
End code	0x0A	

After the request command, the following data should be sent in sequence.

- (1) 6 byte old primary key and 2 byte checksum of it.

- (2) If Key type is not 0x00, 6 byte new primary/secondary key and 2 byte checksum of it.
- (3) 0x0A.

Response command

Field	Data	Description
Start code	0x40	
Command	0xAA	
Param	NULL	
Size	NULL	
Error	0x61 0x75 0x6A 0x6B	SUCCESS UNSUPPORTED NOT_MATCH TRY_AGAIN
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Key or option is changed successfully.
UNSUPPORTED	Invalid key type.
NOT_MATCH	The primary key is wrong.
TRY_AGAIN	Checksum error of key data.

CKR : Read Site Key Option

Reads the site key options.

Request command

Field	Data	Description
Start code	0x40	
Command	0xAB	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xAB	
Param	Bit mask for option	Same as set by CKW
Size	NULL	
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Options are read successfully.

CLW : Write Card Layout

A Mifare 1K card consists of 16 sectors and each sector has four 16 byte blocks. Since the last block of each sector is reserved for site key and the first sector is reserved for system use, there remains 15 sectors * 3 blocks * 16 = 720 bytes. Users can configure the layout of these remaining blocks with this command.

The structure of card layout data is as follows.

Data	Size	Default
Template Size	2 bytes	350
Header Block Index	1 byte	4*
Template 1 Start Block Index	1 byte	5
Template 1 Block Size	1 byte	22 blocks
Template 2 Start Block Index	1 byte	34
Template 2 Block Size	1 byte	22 blocks

* Sector 0: Block 0 ~ Block 3 (Block 3 is reserved for site key)

Sector 1: Block 4 ~ Block 7 (Block 7 is reserved for site key)

...

Sector 15: Block 60 ~ Block 63

Request command

Field	Data	Description
Start code	0x40	
Command	0xAD	
Param	NULL	
Size	Layout data size	7 bytes
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xAD	
Param	NUJLL	
Size	NULL	
Error	0x61	SUCCESS
	0x75	UNSUPPORTED
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Layout data is written successfully.
UNSUPPORTED	Invalid layout data.

CLR : Read Card Layout

Reads the card layout data written by CLW command.

Request command

Field	Data	Description
Start code	0x40	
Command	0xAC	
Param	NULL	
Size	NULL	
Flag	NULL	
Checksum	#	
End code	0x0A	

Response command

Field	Data	Description
Start code	0x40	
Command	0xAC	
Param	NULL	
Size	Layout data size	7 bytes
Error	0x61	SUCCESS
Checksum	#	
End code	0x0A	

Error code

Error code	Description
SUCCESS	Layout data is read successfully.

Contact Info