

ZKFinger SDK Manual

ZKFinger SDK

ZKTeco Inc.

All Right Reserved

ZKFinger SDK Copyright Declaration

ZKFinger SDK and the software development license agreement define your right related to the copyright of this development kit, and if you don't agree with the following agreement, please return the product to the place you bought it immediately.

I. Usage License

You can only copy one software package related to this SDK (including Setup.exe、License.rtf、Biokey.ocx、Manual.doc and pertinent samples) to one single PC, and without written license from ZKTECO CO.,LTD., content of the software package related to this SDK and this manual can not be copied or reprinted in its original form or any other forms by means of paper, electronics or any others.

II. Trademark Registration

ZKSoftware., ZKFinger, and RSoftware are registered trademarks owned by Zhongkong Automation System Inc., Ltd, protected by laws of People's Republic of China, and any illegitimate usage is forbidden. Microsoft is a registered trademark of Microsoft Corporation, and any other products and company names mentioned in this manual can be trademarks of respective owners.

If the content of this manual is changed, sorry for no further notice will be given.

ZKTeco Inc.

User Registration

After you have bought this product, please fill in *User Registration Form* carefully. If you send the completed form to our company by fax or email, you will become a legitimate user of this product, and you are able to obtain our total technical support service and information related to upgrading of this software version!

User Registration Form

1、 You bought the software on (Date): ___/___/___
at (Place): _____

2、 Your name: _____
Position: _____
Addressing: [] Sir [] Madam
Telephone: _____
E-mail: _____
Address: _____

3、 The name of your company: _____

The abbreviation of your company: _____

Company address:

_____ Province/City, _____ City/District, _____

Zip code: _____

Company Telephone: (____) _____

- Staff number: <100 persons
 101to 200 persons
 201to 500 persons
 501to 1000 persons
 above 1000 persons

Website: _____

Email address: _____

- 4、 Are you willing to receive our
- Product upgrading notice? Yes
- New product advertisement? Yes
- Technique information express? Yes
- Website upgrading notice? Yes

Or Email to: sdksupport@zkteco.com

Table of Contents

ZKFinger SDK Manual.....	1
1. ZKFinger Algorithm Description.....	1
2. ZKFinger SDK Architect.....	3
3. Software Installation.....	4
4. ActiveX Control Reference.....	5
4.1 Property.....	5
4.1.1 Active as Boolean.....	5
4.1.2 EngineValid as Boolean.....	5
4.1.3 EnrollIndex As Long.....	5
4.1.4 EnrollCount As Long.....	6
4.1.5 FPEngineVersion AS String.....	6
4.1.6 ImageHeight AS integer.....	6
4.1.7 ImageWidth AS integer.....	6
4.1.8 IsRegister As Boolean.....	6
4.1.9 OneToOneThreshold As Boolean.....	6
4.1.10 RegTplFileName As String.....	7
4.1.11 SensorCount As Long.....	7
4.1.12 SensorIndex AS Long.....	7
4.1.13 SensorSN As String.....	7
4.1.14 TemplateLen As Long.....	7
4.1.15 Threshold As Long.....	7
4.1.16 VerTplFileName As String.....	8
4.1.17 LastQuality As Long.....	8
4.1.18 LowestQuality As Long.....	8
4.1.19 FakeFunOn As Long.....	8

4.2 Method.....	9
Methods for the Control Interfaces of 1:1 and 1:N.....	9
Method for 1:N Control Interface.....	15
Methods for External Image File Interface.....	20
Methods for External Interface.....	21
Methods for EM/Mifare Card.....	23
4.3 Events.....	26
4.3.1 OnCapture(ActionResult AS Boolean, ATemplate).....	26
4.3.2 OnCaptureToFile(ActionResult AS Boolean).....	27
4.3.3 OnEnroll(ActionResult AS Boolean, ATemplate).....	27
4.3.4 OnEnrollToFile(ActionResult AS Boolean).....	27
4.3.5 OnFeatureInfo(AQuality As Long).....	27
4.3.6 OnImageReceived(byval AImageValid As Boolean).....	27
4.3.7 OnFingerTouching.....	28
4.3.8 OnFingerLeaving.....	28
5. Task Flow Description.....	29
6. Common Questions Description.....	32
6.1 The difference between 1:1 and 1:N Control.....	32
6.2 Read-in and Read-out Fingerprint Templates in Database.....	32
6.3 Software Doggle and Authorized License Documentation.....	34
6.4 The use of 1:N high-speed buffer.....	35
6.5 Using fingerprint images.....	35
6.6 Setting fingerprint identification threshold.....	36
6.7 Solutions to low-quality fingerprint templates for 1:N identification.....	36
7. ZKFinger SDK Development License Agreement.....	37
8. Software After-sale Service.....	43

1. ZKFinger Algorithm Description

ZKFinger algorithm is a kind of quick and accurate 1:1 and 1:N fingerprint identification algorithm, which is totally open to software developers and system integrators. If you use ZKFinger to identify fingerprints (2000-6000 pieces of fingerprints), you can complete the identification task easily within 1-5 seconds (the following tests require Pentium III 900MHz+ 128MB EMS memory) without categorizing fingerprints by names, PIN or any others in advance. ZKFinger algorithm has the following features:

- 1、 ZKFinger software development package can be quickly integrated to customers' systems, and can support any scanner device and fingerprint Sensor (Image quality ≥ 300 DPI) through open image process interface.
- 2、 By strainer mirrors and adequate valve values which are self-adaptive or can be easily matched, ZKFinger algorithm is able to weaken noise, increase the contrast degree of the bridge and vale, and even to capture whole or partial feature points from fingerprint of bad quality (fingerprint which is dirty, too dry or wet, broken, or with wounds, scars and marks) .
- 3、 ZKFinger algorithm identification supports the translation of fingerprints ($\geq 35\%$ of the fingerprint size) and circumrotation for 360 degree. Special technology is used to realize speedy verification when the fingerprint is translated or rotated for 360 degree (the average speed is 3000 pieces/second). Even when the fingerprint has few feature points (≤ 10 , normally fingerprint's feature points ≥ 15), this function can also be achieved.

- 4、 ZKFinger algorithm does not require global feature points (core point, triangular point), and identification can be completed by local feature points.
- 5、 Through classification algorithm (fingerprints are classified into five categories: arch category, left loop category, right loop category, tine arch category, and vortex category), ZKFinger use global feature ordering in advance, which accelerates the process of fingerprint verification remarkably.
- 6、 ZKFinger algorithm is quite concise: data only need 350K memory, so that they can easily be imported into embedded systems.

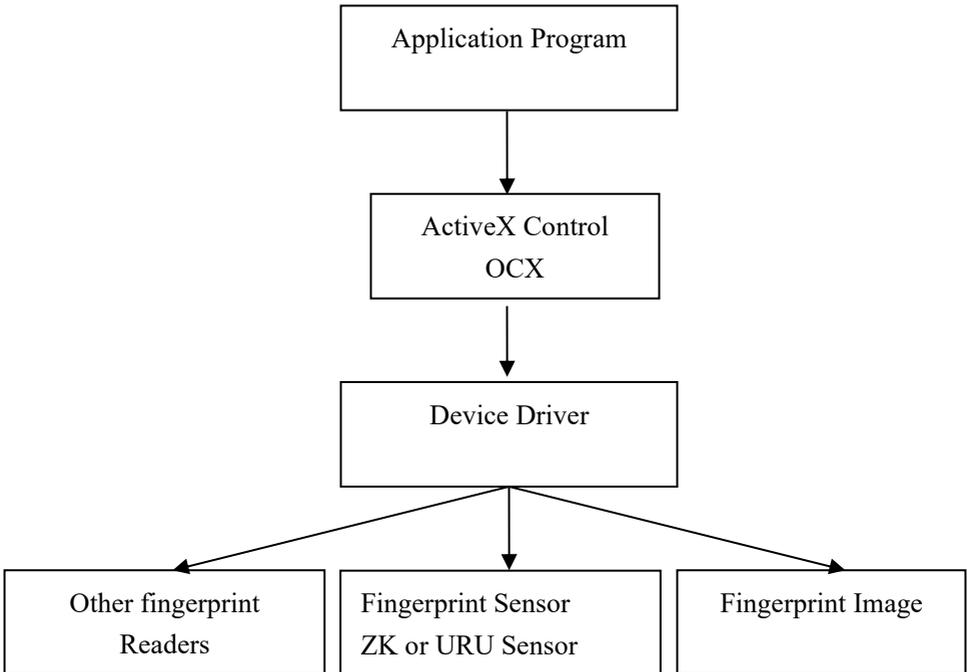
ZKFinger was used to test 2000 pieces of fingerprints collected from four Sensors (YLC, DFR200, U.are.U, and Authentec), every piece of fingerprint was verified with the other ones, and verification and test were carried out for altogether 4,000,000 times, and eventually the following test results were achieved:

Template size	310 or 1152 Byte
Rotation	0 – 360 degree
FAR	<= 0.001%
FRR	<= 2.0%
Registration time	0.5 second
Average verification speed	2500 pieces/second
Image quality	>=300DPI

2. ZKFinger SDK Architect

ZKFinger SDK Pro (Software Development Kit) mainly exists in the form of ActiveX, and users can develop application programs relative to fingerprint sensors by means of varied development languages (such as VC++, C++ Builder, Delphi, VB, Visual FoxPro, PB and so on).

SDK Architecture:



3. Software Installation

Before installing ZKFinger SDK, please make sure that your operation system and the configuration of your computer meet the requirements to run the software.

Before the installation, if your computer has been connected with a fingerprint Reader, you'd better pull it out.

1. Run setup.exe. Click Next button, and you can enter the following operation steps.
2. If it's windows 7 system, please install the patch in "Win7 Patch" folder.

4. ActiveX Control Reference

ZKFinger SDK Pro is divided into two controls: 1:1 and 1:N. For these two, their interfaces are basically of the same property and method, and the methods for the two interfaces only differ in the verification related to 1:N. In our further description, we will try to display the two controls comprehensively, and their differences will be labeled and illuminated.

VB language expression is used here, and fingerprint template Variant variable show as one-dimension byte arrays.

4.1 Property

4.1.1 Active as Boolean

Read only

It indicates whether the fingerprint Reader set by currentSensorIndex has got ready or not.

4.1.2 EngineValid as Boolean

Read only

It indicates whether the fingerprint identification system is performing normally or not. If the function initEngine has been used, effective result will be returned.

4.1.3 EnrollIndex As Long

Read only

The sampling order number at fingerprint registration, that is, the effective times for successful fingerprint registration at present.

4.1.4 EnrollCount As Long

The times for sampling fingerprints at registration, whose value ranges is 1 or 3 or 4.

4.1.5 FPEngineVersion AS String

It indicates the version of fingerprint algorithm. It's "9" (ZKFinger 9.0) as default. You can set "10" (ZKFinger 10.0). You should set it before InitEngine.

4.1.6 ImageHeight AS integer

Read only

It indicates the height of the fingerprint image.

4.1.7 ImageWidth AS integer

Read only

It indicates the width of the fingerprint image.

4.1.8 IsRegister As Boolean

Read only

It indicates whether a fingerprint is being registering or not.

4.1.9 OneToOneThreshold As Boolean

Set the identification threshold value (1-100) for ZKFinger low-speed fingerprint verification—one-to-one, and the default value is 10. The larger of the value, the lower the FAR and the higher the FRR are.

Note: 1:1 control doesn't have this property.

4.1.10 RegTplFileName As String

Set to save the file name of the fingerprint registration template when the event OnEnrollToFile is taking place.

4.1.11 SensorCount As Long

Read only

It indicates the number of fingerprint Readers which are connected to the computer, and if EngineValid is invalid, 0 is returned.

4.1.12 SensorIndex AS Long

Select the order number of the fingerprint head when multiple fingerprint Readers are connected. The serial number starts from 0, and if the number is smaller than 0, the fingerprint Reader will not work.

4.1.13 SensorSN As String

It indicates the serial number for hardware of the fingerprint Reader.

4.1.14 TemplateLen As Long

Read only

It indicates the byte length of the fingerprint registration template.

Note: The template length is 1152 bytes for 1:N, and the template length is 310 bytes for 1:1.

4.1.15 Threshold As Long

Set the verification and identification threshold value (1-100) for the fingerprint identification system, and the default value is 10. The larger the value, the lower the FAR and the higher the FRR are.

4.1.16 VerTplFileName As String

Set to save the file name of the fingerprint verification template when the event OnCaptureToFile is taking place.

4.1.17 LastQuality As Long

Read only

Lastest quality of fingerprint. You can get this property when event OnFeatureInfo triggered. When property LastQuality is less than LowestQuality, event OnFeatureInfo will return Insufficient feature points; -1 means suspicious fingerprint(fake finger).

4.1.18 LowestQuality As Long

Set allowed lowest quality of fingerprint. The default value is 60. When property LastQuality is less than LowestQuality, event OnFeatureInfo will return Insufficient feature points.

4.1.19 FakeFunOn As Long

Set on/of anti-fake function. The default value is 1. 1 means set on, 0 means set off.

4.2 Method

Methods for the Control Interfaces of 1:1 and 1:N

4.2.1 Sub BeginEnroll()

Begin to register fingerprints, and the event OnEnroll will take place when the registration completes.

4.2.2 Sub CancelEnroll()

Cancel the current status of fingerprint registration, that is, the operation started from BeginEnroll will be cancelled by this function.

4.2.3 Function DongleIsExist As Boolean

Examine whether Doggle is existing or not.

4.2.4 Function DongleSeed(Byval lp2 As Long, Byval p1, p2, p3, p4 As Integer) As Boolean

Obtain four 16-digital integral (p1, p2, p3, p4) return values for the seed code lp2. Doggle can compute a seed code by interior algorithm, which results in four return codes. Seed code algorithm is not open, and by examining whether the return codes are of the expected value we can examine whether Doggle is existing or not.

4.2.5 Function DongleUserID As Long

Read User ID in Doggle, and User ID will not repeat. Save it in specific location within Doggle.

This function has been canceled, here in order to compatible with before, we keep up this function.

4.2.6 Function DongleMemRead(Byval p1, p2 As Integer, buf) As Boolean

Read the p2 bytes started from p1 located in Doggle memory to Variant variable buf (one-dimension byte array). There are altogether 24 bytes in the memory, located from 0 to 23.

This function has been canceled, here in order to compatible with before, we keep up this function.

4.2.7 Function DongleMemWrite(Byval p1, p2 As Integer, buf) As Boolean

Write Variant variable buf (one-dimension byte array) to the p2 bytes started from p1 located in Doggle memory. There are altogether 24 bytes in the memory, located from 0 to 23.

This function has been canceled, here in order to compatible with before, we keep up this function.

4.2.8 Function GetTemplate()

Get the fingerprint template, which is obtained most recently.

4.2.9 Function GetFingerImage(Byval AFingerImage) As Boolean

Get the fingerprint image (BMP format), which is obtained most recently.

4.2.10 Function InitEngine() As Long

Initialize the fingerprint identification system. Property such as SensorCount, SensorSN, EngineValid, ImageHeight and ImageWidth will not return accurate results only after this function has been called. Return values:

0 Initialization succeeded.

1 The loading of the fingerprint identification driver failed.

2 Fingerprint Sensor has not been connected.

3 The fingerprint Reader appointed by the property SensorIndex dose not exist (**Note:** Set the property SensorIndex before calling the function).

Method EndEngine can be used to release the fingerprint device system.

4.2.11 Function VerFinger(byval regTemplate, verTemplate, AdoLearning As Boolean, byval AregFeatureChanged As Boolean) As Boolean

Compare whether the feature templates for two pieces of fingerprints are matched or not. Here, regTemplate represents fingerprint registration feature templates, verTemplate expresses fingerprint verification feature templates which are collected on the spot, AdoLearning denotes whether to carry out fingerprint feature template learning updating or not, and AregFeatureChanged shows whether the registration template regTemplate has been changed or not. True will be returned when the two pieces of fingerprints are matched, and False will be returned when not matched.

Explanation:

The fingerprint feature will vary to certain extent with the time, usually which will not pose an influence on the verification of fingerprints. While by fingerprint feature template learning updating, the system can obtain an integrated new template, so as to lower the FRR.

4.2.12 Function VerFingerFromFile(regTemplateFile As String, verTemplateFile As String , AdoLearning As Boolean, byval AregFeatureChanged As Boolean) As Boolean

Compare whether the feature templates for two pieces of fingerprints are matched or not. Here, regTemplate represents fingerprint registration feature templates, verTemplate expresses fingerprint verification feature templates which are collected on the spot, AdoLearning denotes whether to carry out fingerprint

feature template learning updating or not, and `AregFeatureChanged` shows whether the registration template `regTemplate` has been changed or not. True will be returned when the two pieces of fingerprints are matched, and False will be returned when not matched.

4.2.13 Function `VerRegFingerFile(RegTemplateFile As String, verTemplate, AdoLearning As Boolean, byval AregFeatureChanged As Boolean) As Boolean`

Compare whether the feature templates for two pieces of fingerprints are matched or not. Here, `regTemplate` represents previous fingerprint registration feature templates in the file specified by `FileName`, `verTemplate` expresses fingerprint feature templates which are collected on the spot, `AdoLearning` denotes whether to carry out fingerprint feature template learning updating or not, and `AregFeatureChanged` shows whether the registration template `regTemplate` has been changed or not. True will be returned when the two pieces of fingerprints are matched, and False will be returned when not matched.

4.2.14 Sub `PrintImageAt(HDC As OLE_HANDLE, X As Long, Y As Long, aWidth As Long, aHeight As Long)`

At the location specified by `(x,y)`, display the fingerprint image in accordance with the size specified by `(aWidth, aHeight)`. `HDC` is the `HDC` for the window in which the fingerprint will be shown.

4.2.15 Sub `PrintImageEllipseAt(HDC As OLE_HANDLE, X As Long, Y As Long, aWidth As Long, aHeight As Long, bkColor As OLE_COLOR)`

At the location specified by `(x,y)`, display the fingerprint image in accordance with the size specified by `(aWidth, aHeight)`. `HDC` is the `HDC` for the window in which the fingerprint will be shown. Here the fingerprint image is

surrounded by an ellipse.

4.2.16 Sub SaveBitmap(FileName As String)

Save the last fingerprint image to the bitmap file specified by FileName.

4.2.17 Sub SaveJPG(FileName As String)

Save the last fingerprint image as the Jpeg file that specified by FileName.

4.2.18 Function SaveTemplate(FileName As String, Template) As Boolean

Save the feature template for the Template fingerprint to the file specified by FileName.

4.2.19 function EncodeTemplate(ASour, var ADest As String) As Boolean

Transfer the Variant template ASour used by the control into the template string Adest, which is BASE64 formatted.

4.2.20 function DecodeTemplate(const ASour As String, ADest) As Boolean

Transfer the template string ASour which is BASE64 formatted into the Variant-typed template ASour used by the control.

The above-mentioned two methods are mainly used for saving database of templates. Variant-typed templates are saved in the manner of binary-formatted arrays, which are quite difficult for languages such as PB, VB, etc. Method EncodeTemplate can transfer Variant-typed codes into strings, and method DecodeTemplate can transfer string-typed codes into codes of Variant-typed. Here, we should pay attention that the template length will be increased after the template variable BASE64 code has been transferred into the string.

4.2.21 function EncodeTemplate1(ASour) As String

Transfer the Variant template ASour used by the control into the template string, which is BASE64 formatted.

4.2.22 function DecodeTemplate1(const ASour As String) As Variant

Transfer the template string ASour which is BASE64 formatted into the Variant-typed template used by the control.

4.2.23 Sub BeginCapture()

Set the current fingerprint device to begin to capture images, and method CancelCapture can be used to forbid the current device to capture images.

4.2.24 Sub EndEngine()

Release the fingerprint device initialized by method InitEngine, and method InitEngine can be utilized to re-initialize fingerprint device.

4.2.25 function VerFingerFromStr(regTemplateStr As String, verTemplateStr As String, AdoLearning As Boolean, byval AregFeatureChanged As Boolean) As Boolean

Compare whether the feature templates for two pieces of fingerprints are matched or not. Here, regTemplateStr(BASE64 formatted string) represents fingerprint registration feature templates, verTemplateStr(BASE64 formatted string) expresses fingerprint verification feature templates which are collected on the spot, AdoLearning denotes whether to carry out fingerprint feature template learning updating or not, and AregFeatureChanged shows whether the registration template regTemplate has been changed or not. True will be returned when the two pieces of fingerprints are matched, and False will be returned when not matched.

4.2.26 function GetTemplateAsString() As String

Get the fingerprint Verify or Register template, which is obtained most recently. It may be called on OnCapture , OnEnroll, OnCaptureToFile, OnEnrollToFile event. The return result is BASE64 formatted template string.

4.2.27 function ControlSensor(ACode As Long; AValue As Long)As Long

If ACode=11, control the green light, if it's 12, control the red light, if it's 13, control the beep.

If AValue=1, lights on, if it's 0, lights off.

Explanation:

Only support ZK4000 sensor and ZK8000 sensor currently.

Method for 1:N Control Interface

4.2.28 Function AddRegTemplateToFPCacheDB(fpcHandle As Long, FPID As Long, pRegTemplate) As Long

Add the fingerprint registration template pRegTemplate to the fingerprint sensor's high-speed buffer fpcHandle, and FPID, which must be larger than 0, is the (unique) label for adding registration template.

4.2.29 Function AddRegTemplateFileToFPCacheDB(fpcHandle As Long, FPID As Long, pRegTemplateFile As String) As Long

Add the previous fingerprint registration feature template in the file specified by pRegTemplateFile to the fingerprint identification high-speed buffer fpcHandle , and FPID, which must be larger than 0, is the (unique) label for adding the registration template. If the return value is 1, it indicates a success, and 0 indicates a failure.

4.2.30 Function CreateFPCacheDB As Long

Create the fingerprint identification high-speed buffer. As for 1:N identification, this function must be first called so as to obtain the fingerprint identification buffer handle.

Explanation:

ZKFinger 1:1 low-speed verification speed is rather slow (about 30ms for PII 233), so fingerprints of 1:1 genre added to the buffer by calling the function AddRegTemplateToFPCache can not be too many; otherwise, the verification speed will be impacted. By IsOneToOneTemplate, we can judge whether the fingerprint is of 1:1 type or not.

At the same time multiple buffers can be created for group comparison and others.

4.2.31 Sub FlushFPImages ()

Delete buffer images in the current fingerprint device.

4.2.32 Sub FreeFPCacheDB(fpcHandle As Long)

Release the fingerprint identification high-speed buffer. FpcHandle is the fingerprint identification buffer handle obtained by calling the function CreateFPCacheDB.

4.2.33 Function IdentificationFromFileInFPCacheDB (fpcHandle As Long, pVerTemplateFile As String, Byval Score As Long, Byval ProcessedFPNumber As Long) As Long

Compare all the registration templates in the fingerprint identification high-speed buffer fpcHandle with the fingerprint verification template in the file pVerTemplateFile. Score represents the highest score among ProcessedFPNumber times of verification, and ProcessedFPNumber shows the times of verification.

The fingerprint label will be returned if the identification is successful, and -1 is returned if failed.

Note:

During the process of identification, if the verification score is equal to or larger than the property Threshold, then it is considered that the verification is successful. In this case, no further verification will be carried out for the rest of fingerprint registration templates in the buffer, and this function returns the fingerprint label for the fingerprint registration template which is matched successfully;

If all the scores for the verification between all the fingerprint verification templates and all the fingerprint identification templates located in the fingerprint identification high-speed buffer, but meanwhile the highest score for the verification is equal to or larger than Score, then it is viewed that the verification is matched successfully. In this case, this function will return the label of the fingerprint registration template which gets the highest verification score, whose recommended value is 9.

4.2.34 Function IdentificationInFPCacheDB (fpCHandle As Long, pVerTemplate, Byval Score As Long, Byval ProcessedFPNumber As Long) As Long

Compare all the registration templates in the fingerprint identification high-speed buffer fpCHandle with the fingerprint verification template pVerTemplate. Score represents the highest score among ProcessedFPNumber times of verification, and ProcessedFPNumber shows the times of verification. The fingerprint label will be returned if the identification is successful, and -1 is returned if failed.

Note:

During the process of identification, if the verification score is equal to or larger than the property Threshold, then it is considered that the verification is successful. In this case, no further verification will be carried out for the rest of

fingerprint registration templates in the buffer, and this function returns the fingerprint label for the fingerprint registration template which is matched successfully;

If all the scores for the verification between all the fingerprint verification templates and all the fingerprint identification templates located in the fingerprint identification high-speed buffer, but meanwhile the highest score for the verification is equal to or larger than Score, then it is viewed that the verification is matched successfully. In this case, this function will return the label of the fingerprint registration template which gets the highest verification score, whose recommended value is 9.

4.2.35 Function IsOneToOneTemplate (Atemplate) As Boolean

Judge the current fingerprint feature template Atemplate is a ZKFinger 1:1 low-speed verification feature template.

4.2.36 Function ModifyTemplate(byval Atemplate, AOneToOne As Boolean) As Boolean

According AoneToOne modify the fingerprint feature template Atemplate to a ZKFinger 1:1 low-speed verification feature template or a high-speed verification feature template.

4.2.37 Function RemoveRegTemplateFromFPCacheDB (fpcHandle As Long, FPID As Long) As Long

Delete the fingerprint registration template which is labeled as FPID located in the fingerprint identification high-speed buffer fpcHandle. If the return value is 1, it indicates a success, and 0 represents a failure.

4.2.38 Sub CancelCapture()

Forbid the current fingerprint device to capture images, and the method BeginCapture can be used for the fingerprint device to begin to capture images.

4.2.39 Function AddRegTemplateStrToFPCacheDB(fpcHandle As Long, FPID As Long, ARegTemplateStr As String) As Long

Add the previous fingerprint registration feature template ARegTemplateStr which is BASE64 code string to the fingerprint identification high-speed buffer fpcHandle, and FPID, which must be larger than 0, is the (unique) label for adding the registration template. If the return value is 1, it indicates a success, and 0 indicates a failure.

4.2.40 Function IdentificationFromStrInFPCacheDB (fpcHandle As Long, AVerTemplateStr As String, Byval Score As Long, Byval ProcessedFPNumber As Long) As Long

Compare all the registration templates in the fingerprint identification high-speed buffer fpcHandle with the verification template AVerTemplateStr, which is BASE64 code formatted string. Score represents the highest score among ProcessedFPNumber times of verification, and ProcessedFPNumber shows the times of verification. The fingerprint label will be returned if the identification is successful, and -1 is returned if failed.

Note:

During the process of identification, if the verification score is equal to or larger than the property Threshold, then it is considered that the verification is successful. In this case, no further verification will be carried out for the rest of fingerprint registration templates in the buffer, and this function returns the fingerprint label for the fingerprint registration template which is matched successfully;

If all the scores for the verification between all the fingerprint verification

templates and all the fingerprint identification templates located in the fingerprint identification high-speed buffer, but meanwhile the highest score for the verification is equal to or larger than Score, then it is viewed that the verification is matched successfully. In this case, this function will return the label of the fingerprint registration template which gets the highest verification score, whose recommended value is 9.

4.2.41 Sub SetAutoIdentifyPara(AutoIdentify As Boolean, fpcHandle As Long, Score As Long)

Set the Internal Recognition Style AutoIdentify, fingerprint identification high-speed buffer fpcHandle and the SCORE which the same as the parameter Score of IdentificationFromInFPCacheDB method. Refer OnCapture event.

Methods for External Image File Interface

4.2.42 Function AddBitmap(BitmapHandle As OLE_HANDLE, ValidRectX1 As Long, ValidRectY1 As Long, ValidRectX2 As Long, ValidRectY2 As Long, DPI As Long) As Boolean

Enroll or verify the user with bitmap specified by BitmapHandle. The parameters ValidRectX1, ValidRectY1, ValidRectX2 and ValidRectY2 specify the effective area of the image. If the specified area is invalid, the whole image is effective. DPI specifies the resolution of the image.

4.2.43 Function AddImageFile(FileName As String, DPI As Long) As Boolean

Enroll or verify the user with fingerprint image (supports BMP or JPG format) specified by FileName. DPI specifies the resolution of the image.

Before using these two functions, if the image is used for fingerprint

enrollment, use BeginEnroll first and then set EnrollCount. Otherwise, if the image is used for verification, use BeginCapturefirst and then set AddImageFile or AddBitmap, and the system will trigger OnEnroll or OnCapture event.

The two external image interfaces are not support in ZKFinger SDK Lite Version.

Methods for External Interface

4.2.44 Function CreateFPCacheDBEx As Integer

The function is similar to CreateFPCacheDB, except that it can create both 9.0 and 10.0 fingerprint identification high speed cache at the same time.

4.2.45 Sub FreeFPCacheDBEx(fpcHandle As Long)

The function is similar to FreeFPCacheDB, except that it can create both 9.0 and 10.0 fingerprint identification high speed cache at the same time.

4.2.46 Function AddRegTemplateStrToFPCacheDBEx(fpcHandle As Integer, FPID As Integer, ARegTemplateStr As Stirng, ARegTemplate10Str As String) As Long

The function is similar to AddRegTemplateStrToFPCacheDB, except that it can create both 9.0 and 10.0 fingerprint identification high speed cache at the same time.

4.2.47 Function AddRegTemplateToFPCacheDBEx (fpcHandle As Integer, FPID As Integer, pRegTemplate, pRegTemplate10) As Long

The function is similar to AddRegTemplateStrToFPCacheDBEx, except that it uses Variant fingerprint template.

4.2.48 Function AddRegTemplateFileToFPCacheDBEx (fpcHandle As Integer, FPID As Integer, pRegTemplateFile As String, pRegTemplateFile10 As String) As Long

The function is similar to AddRegTemplateStrToFPCacheDBEx, except that it will upload the fingerprint template from the file.

4.2.49 Function RemoveRegTemplateFromFPCacheDBEx(fpcHandle As Long, FPID As Long) As Long

The function is similar to RemoveRegTemplateFromFPCacheDB, except that it will clear both 9.0 and 10.0 fingerprint template at the same time.

4.2.50 Function GetTemplateEx(AFPEngineVersion As String) As Variant

The function is similar to GetTemplate, except that it can import AFPEngineVersion parameter to get 9.0 or 10.0 fingerprint template.

If AFPEngineVersion is 9, get the 9.0 fingerprint template. If it's 10, get the 10.0 fingerprint template.

4.2.51 Function GetTemplateAsStringEx(AFPEngineVersion As String) As String

The function is similar to GetTemplateEx, except that the return template string is converted to BASE64 format.

Methods for EM/Mifare Card

4.2.52 Function MF_GET_SNR(commHandle As Long, deviceAddress As Long, mode As Byte, rDM_halt As Byte, ByRef snr As Byte, ByRef value As Byte) As Boolean

Return a single or multiple cards of 1 byte, card number of 4 bytes.

commHandle: Communication serial port of handling, 0.

deviceAddress: 0.

mode: 0x26 mode control can only operate one card at once, 0x52 mode control can operate multiple cards at once.

API_halt: Whether to need halt card, 00 means don't need and 01 means need.

snr: The returned single or multiple cards of 1 byte(snr[0] is error code when read card failed)

value: The returned card number of 4 bytes.

Explanation:

Only support ZK8000 sensor currently.

4.2.53 Function MF_GetSerNum(commHandle As Long, deviceAddress As Long, ByRef buffer As Byte) As Boolean

Read the reader address of 1 byte and the reader serial number of 8 bytes.

commHandle: Communication serial port of handling, 0.

deviceAddress: 0.

buffer: The reader-writer address of buffer[0], the 8 bytes reader-writer serial number of buffer[1...8].

Explanation:

Only support ZK8000 sensor currently.

4.2.54 Function MF_SetSerNum(commHandle As Long, deviceAddress As Long, ByRef newValue As Byte, ByRef buffer As Byte) As Boolean

Set the 8 bytes reader-writer serial number.

commHandle: Communication serial port of handling, 0.

deviceAddress: 0.

newValue: The reader-writer serial number of 8 bytes.

buffer: The returned error code when operate failed.

Explanation:

Only support ZK8000 sensor currently.

4.2.55 Function MF_GetVersionNum(commHandle As Long, deviceAddress As Long, ByRef versionNum As Byte) As Boolean

Read the version number of reader.

commHandle: Communication serial port of handling, 0.

deviceAddress: 0.

versionNum: The version number of reader-writer, versionNum[0] is error code when operate failed.

Explanation:

Only support ZK8000 sensor currently.

4.2.56 Function MF_PCDRead(commHandle As Long, deviceAddress As Long, mode As Byte, blkIndex As Byte, blkNum As Byte, ByRef key As Byte, ByRef buffer As Byte) As Boolean

Read the data of card.

commHandle: Communication serial port of handling, 0.

deviceAddress: 0.

Mode: 0(keyA+single card) 1(keyA+multi cards) 2(keyB+single card) 3(keyB+multi cards).

blkIndex: block index.

blkNum: number of block.

key: 6 bytes key, return 4 bytes of card number when read successful.

buffer: buffer[0] is error code when read failed, while it's the read data when read successful.

Explanation:

Only support ZK8000 sensor currently.

Error code:

0x80: Parameters set successful.

0x81: Parameters set failed.

0x82: Communication timeout.

0x83: Card doesn't exist.

0x84: Data error of receiving card.

0x87: Unknown error.

0x85: Parameter or command format input error.

0x8A: error when initialize the card.

0x8B: The error serial number got in preventing conflict.

0x8C: Password authentication failed.

0x90: Card doesn't support this command.

0x91: Command format error.

0x92: The FLAG parameter in command doesn't support the OPTION mode.

0x93: The BLOCK to be operated doesn't exist.

0x94: The object to be operated has been locked, and it can't be modified.

0x95: The lock operation failed.

0x96: The write operation failed.

4.2.57 Function MF_PCDWrite(commHandle As Long, deviceAddress As Long, mode As Byte, blkIndex As Byte, blkNum As Byte, ByRef key As Byte, ByRef buffer As Byte) As Boolean

Write the data to card.

commHandle: Communication serial port of handling, 0.

deviceAddress: 0.

mode: 0(keyA+single card) 1(keyA+multi cards) 2(keyB+single card)
3(keyB+multi cards).

blkIndex: block index.

blkNum: number of block.

key:6 bytes key, return 4 bytes of card number when read successful.

buffer: The data to be written, buffer[0] is error code when read failed(Reference 4.2.56).

Explanation:

Only support ZK8000 sensor currently.

4.3 Events

4.3.1 OnCapture(ActionResult AS Boolean, ATemplate)

When the AutoIdentify (set by SetAutoIdentifyPara method) value set to False, Capture the fingerprint verification template Atemplate for verification. ActionResult =true indicates that the fingerprint template is obtained successfully, and False represents a failure.

When the AutoIdentify (set by SetAutoIdentifyPara method) value set to True, the ATemplate is fingerprint identification result. The result is one-dimension array, please refer the following definition:

ATemplate[0] ID value, ID value is -1 if failed

ATemplate[1] return one-to-many recognition score

ATemplate[2] the processed fingerprint number for 1:N

ATemplate[3] the processed fingerprint number for 1:1

4.3.2 OnCaptureToFile(ActionResult AS Boolean)

Obtain the fingerprint verification template for verification, and the template is saved in a file, whose name is set by the property VerTplFileName. ActionResult =true indicates that the fingerprint template is obtained successfully, and False represents a failure.

4.3.3 OnEnroll(ActionResult AS Boolean, ATemplate)

Call this event when the user fingerprint registration completes. ActionResult =true indicates that the registration is successful, and the fingerprint feature template can be captured by using pTemplate property; False represents a failure.

4.3.4 OnEnrollToFile(ActionResult AS Boolean)

Call this event when the user fingerprint registration completes. ActionResult =true indicates that the registration is successful, and the fingerprint feature template is saved in a file, whose name is set as the property RegTplFileName; False represents a failure.

4.3.5 OnFeatureInfo(AQuality As Long)

Obtain the fingerprint's initial feature. Quality represents the quality of this fingerprint's feature, and it may have the following values:

- 0: Good fingerprint feature
- 1: Insufficient feature points
- 2: Other reasons resulting in the incapability to capture the fingerprint feature.
- 1: Suspicious fingerprint

4.3.6 OnImageReceived(byval AImageValid As Boolean)

Call this event if the device receives the fingerprint image, or the fingerprint

image is added by AddImageFile and AddBitmap. AimageValid indicates whether to extract a template or not. If it is set as False, the system will not extract templates after it has captured the fingerprint image.

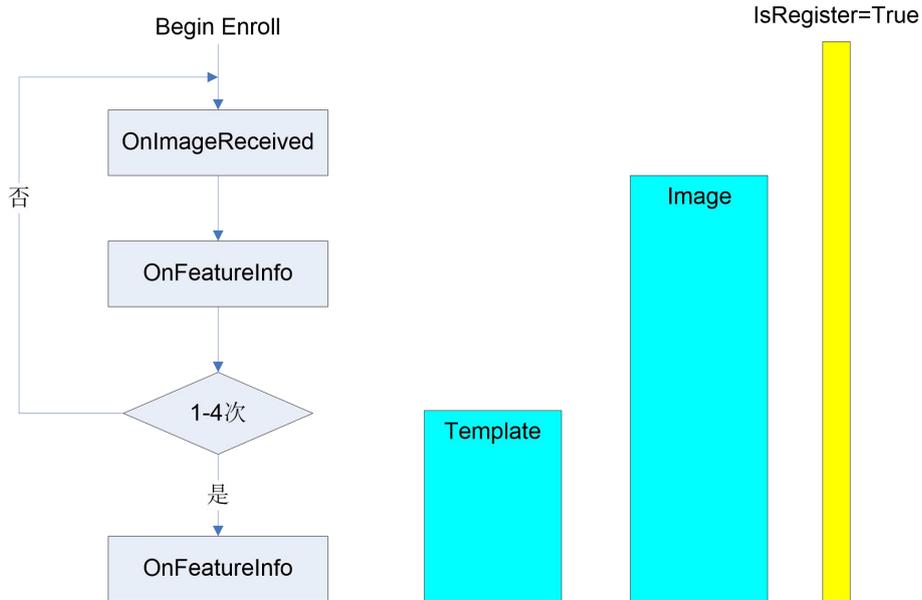
4.3.7 OnFingerTouching

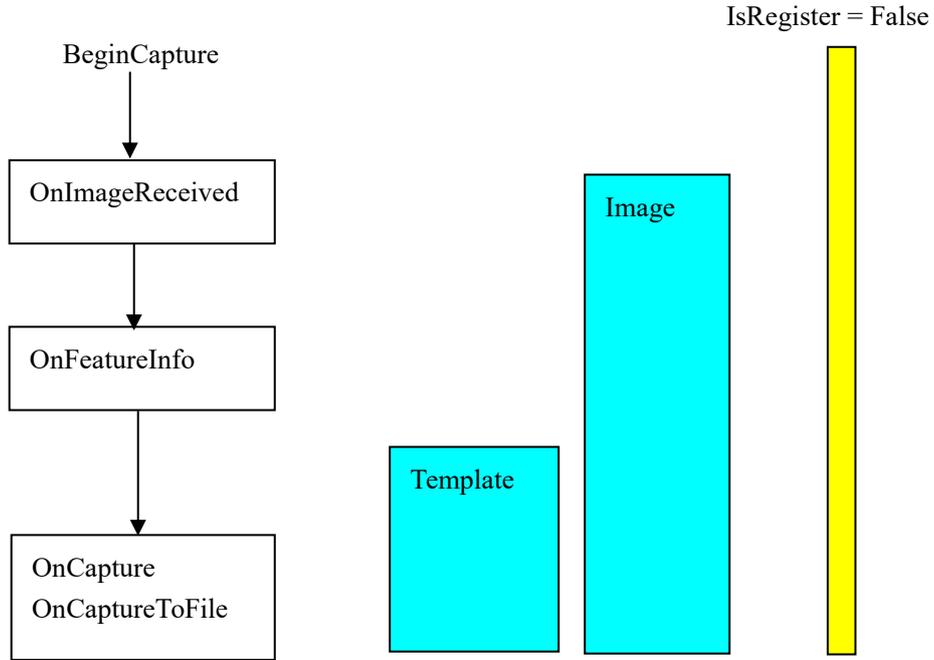
Call this event when press finger on sensor.

4.3.8 OnFingerLeaving

Call this event when removed finger from sensor.

5. Task Flow Description





Task Flow Description:

After the fingerprint Sensor has entered the working status, call `BeginEnroll` to enter fingerprint registration status, and call `BeginCapture` to enter fingerprint verification status. The working mode is based on event motivation, and events can be triggered in an order as shown in the above sketch graph.

It is usually needed to press the same finger for 1 to 4 times for fingerprint registration, and a fingerprint registration template will be obtained after the identification system integrates them. The times number needed for pressing the finger at registration is defined by the control property `EnrollCount`, and events `OnEnroll` and `OnEnrollToFile` will be triggered if the defined times number is arrived at.

At fingerprint verification, events `OnCapture` and `OnCaptureToFile` will be triggered after pressing the finger. At this moment, `VerFinger` or `IdentificationInFPCacheDB` can be called to carry out 1:1 or 1:N verification.

It should be paid attention to that event `OnFeatureInfo` will be triggered when a finger is being pressed for each time. If the fingerprint template of the finger which is being pressed is not qualified, this time the image captured is invalid, and the finger should be re-pressed.

6. Common Questions Description

6.1 The difference between 1:1 and 1:N Control

1 : 1 Control is mainly used for development projects which need 1:1 verification, and usually the currently-verified client's PIN should be entered in advance, and after that one or several templates he/she has registered are obtained for verification; while 1:N Control does not require entering the client's PIN, and this client can be identified by the client's fingerprint out of the registered fingerprint templates.

1:1 Control mainly aims to achieve a high pass rate and a relatively high accuracy rate; 1:N Control is principally designed to obtain a high verification speed and a relatively high accuracy rate.

1 : 1 Control's maximum template length is only 310 bytes, while that for 1:N Control is 1152 bytes. Because 1:N requires high-speed verification, and a remarkably low RAR, relatively more template feature information should be saved.

6.2 Read-in and Read-out Fingerprint Templates in Database

In SDK fingerprint templates are saved and called by means of Variant variable. What is stored is one-dimension binary arrays, which can not be read-in and read-out by SQL sentences as for character strings. There are two solutions:

- 1、 Method EncodeTemplate and method DecodeTemplate are able to make BASE64 code transfer between Variant variables and string variables. One point, which should be improved, is that after variables being transferred into strings, the template length will be increased by about 1/3.
- 2、 Directly work on Variant variables. An example is shown as the following:
Delphi, CB:

```

procedure TFPPProcess.SaveFPData(AQuery: TADOQuery; AFingerID: Integer; AFPData:
OleVariant);
var
  pData: PChar;
begin
  with AQuery do begin
    Close;
    SQL.Clear;
    SQL.Add('SELECT * FROM zkFingerPrint WHERE FingerID = ' + IntToStr(AFingerID));
    Open;
    if IsEmpty then
      Append
    else
      Edit;
    FieldByName('FingerID').Value := AFingerID;
    //Save the fingerprint template
    with TBlobStream(CreateBlobStream(FieldByName('Template'), bmWrite)) do begin
      pData := VarArrayLock(AFPData);
      try
        Write(pData^, VarArrayHighBound(AFPData, 1) - VarArrayLowBound(AFPData, 1) + 1);
      finally
        VarArrayUnlock(AFPData);
      end;
      Free;
    end;
    Post;
    Close;
  end;
end;
procedure TFPPProcess.GetFPData(AQuery: TADOQuery; AFingerID: Integer; var AFPData:

```

```
OleVariant);  
var  
    pData: PChar;  
begin  
    with AQuery do begin  
        Close;  
        SQL.Clear;  
        SQL.Add('SELECT * FROM zkFingerPrint WHERE FingerID = ' + IntToStr(AFingerID));  
        Open;  
        //read-out data  
        if not IsEmpty then  
            with TBlobStream(CreateBlobStream(FieldByName('Template'), bmRead)) do begin  
                AFPData := VarArrayCreate([0, Size + 1], varByte);  
                pData := VarArrayLock(AFPData);  
                try  
                    Read(pData^, Size);  
                finally  
                    VarArrayUnlock(AFPData);  
                end;  
                Free;  
            end;  
        Close;  
    end;  
end;
```

For other languages, please refer to the technical discussion forum on www.zkteco.com.

6.3 Software Doggle and Authorized License Documentation

The running of SDK requires software Doggle and authorized license

documentation. The difference between software Doggle and authorized license documentation is that software Doggle is independent of the fingerprint Sensor which is being used, while authorized license documentation corresponds to the fingerprint Sensor which is being used. That is, software Doggle can be used with all fingerprint Sensors, but authorized license documentation can only be used with the authorized fingerprint Sensor.

This item has been canceled in the 4.0 Version

6.4 The use of 1:N high-speed buffer

For 1:N verification, it is needed to categorize the templates to be verified; at the same time, in order to achieve higher speed, SDK needs to create memory first, and then add registered fingerprints to the memory. In fact, high-speed buffer is a kind of memory. In practice, firstly we need to create the buffer by the method `CreateFPCahceDB`, and then by methods such as, `AddRegTemplateToFPCahceDB`, `RemoveRegTemplateFromFPCacheDB` and so on, to add or delete fingerprint registration templates, and lastly free the memory by the method `FreeFPCacheDB`.

We can create multiple high-speed buffers at the same time so as to realize such functions as grouping query.

6.5 Using fingerprint images

In projects where 1:1 Control is used, quite often you are required to save fingerprint images, or to obtain plane fingerprint images directly from scanning. Thus, 1:1 Control SDK provides methods, which are capable of capturing fingerprint registration templates directly from plane fingerprint images, such as `AddImageFile`. But what should be paid attention to is that images should be captured correctly, and their resolution must not be less than 350DPI.

Note: SDK in Lite Version dose not provide these methods.

6.6 Setting fingerprint identification threshold

In 1:1 Control, the recommendation value for the property Threshold is 10. In this case, FAR is about 0.01%, and FRR ranges from 1.5% to 2%.

In 1:1 Control, the recommendation value for the property Threshold is 10. In this case, FAR is about 0.001%, FRR is about 3%, and the recommendation value for the property OneToOneThreshold is 10.

6.7 Solutions to low-quality fingerprint templates for 1:N identification

In 1:N Control, at fingerprint registration, the system will automatically label fingerprints by categories in terms of quality, and save them in templates. Templates of low quality are titled as ZKFinger 1:1 low-speed verification feature templates, and 1:1 in this term is a concept quite different from 1:1 in 1:1 Control, which should not be mixed up. Registration templates of high quality are labeled as ZKFinger high-speed verification feature templates.

In normal application environments, about 5% of the total fingerprint registration templates will be labeled as low-speed verification feature templates. The method `IsOneToOneTemplate` can be used to judge whether the template is a low-speed verification feature template, and the method `ModifyTemplate` can be used to modify grouping labels which distinguish templates of high quality from those of low quality forcefully.

ZKFinger 1:1 low-speed verification is quite slow (about 30ms for PII 233), thus not too many low-speed verification feature templates can be added to the high-speed buffer by the method `AddRegTemplateToFPCache`; otherwise the verification speed will be impacted.

7. ZKFinger SDK Development License Agreement

All these who order and use the products (including, but not limited to, ZKFinger development kit, discs, CD-ROM, Doggle and Development Manual) provided by Beijing Zhongkong Automation System Inc., Ltd (or its subsidiary company, as “ZKTeco Inc.” mentioned below), the subsidiary of ZKTeco Inc., Ltd, are subject to the terms and conditions set up in this agreement (ZKFinger is the registered trade mark of the nuclear technology of ZKTeco).

If you open the sealed kit consisting of ZKFinger Development Kit and Development Manual, or you install the below-mentioned software or use this software on your computer, or you are using any products of ZKTeco, you will accept this agreement without option, and at the same time, you must observe the terms and conditions set up in this agreement.

If you do not want to be subject to the terms and conditions set up in this agreement, you must return ZKFinger Development Kit and Development Manual to ZKTeco Inc. immediately (at least within 7 days after you have received this information kit), and your will receive the refund.

1. Title and Property

This is a license agreement but not a sales agreement. You are authorized by ZKTeco Inc. here, thus from an individual or a company juridical person, you receive the license (the license to transfer and the right to resell are subject to the terms and conditions clearly set up hereinto) to use the products of ZKTeco Inc, which is not exclusive and can not be transferred, and these licenses and rights are stated by the terms and conditions herein.

The software components of ZKTeco’s products, including any editing,

correcting, modifying, adding or updating ZKTeco Development Manual, any other files or user guide related to the software, will reserve the property for ZKTeco Inc.

All software (including, but not limited to: task products derived from the software codes and the second part of this agreement) consist of or express the pertinent intelligence property (including, but not limited to: copyright, commercial secrets, trade marks, and so on), and ZKTeco Inc.'s Development Manual and any other files are, and should be only possessed by ZKTeco Inc. Under any laws and rules, any content in this agreement does not construct the wavering of ZKTeco Inc.'s intelligence property

2. License

You are allowed with limit to use the software, which must be stated by this agreement and must also be executable file formatted.

- a) You can install and use the software in the computer in your office.
- b) You are allowed to have copies of the software of reasonable number, which can not surpass three.
- c) You can connect and copy the software to your computer's programs, only for the purpose of the development of this computer's programs, as described in ZKTeco Inc.'s Development Manual; but, any parts connected and copied to another computer is viewed as derived products, and the terms and conditions in this agreement should also be observed.

3. License Transfer

According to the description in the second part, in you computer's programs, after you plug in the software, you may integrate the plugged-in software and the software purchased from ZKTeco Inc., and then transfer the license to a seller or a

user in accordance with the terms and conditions in this agreement. Before transferring the license, you should add the clauses related to guaranty, waiver and license stated by ZKTeco Inc. in this agreement to the new agreement signed by you and the seller or the user, or you provide the above-mentioned clauses to the seller or the user directly.

4. Forbiddance

You are granted some rights as clarified in the above Section 1 and Section 2, but you should agree not to:

- d) Use, modify , insert or transfer the software or any other products of ZKTeco Inc. (including, but not limited to, test modules), except authorization specifically stated in this agreement;
- e) Sell, transfer license, and lease, transfer, mortgage or share your rights to others under this agreement;
- f) Modify, disassemble, decompile, reverse-engine, delete or add original codes of this software or codes intended to deduce this software;
- g) Put this software on a server, so that others are able to obtain it from a public network;
- h) Use the backup of this software or documentation copies (or let others have this kind of copies) for any purpose, except replacing the original copy which has been broken or has malfunctions.

If you are a member of the European Union, this agreement will not influence your legitimate rights specified in the Manual on Computer Software Protection authorized by EC Committee. If you are looking for any information related to this guide, you may contact ZKTeco Inc.

5. Limited Guaranty

ZKTeco Inc. makes a twelve-month guaranty (guaranty period), that:

- i) When the software is delivered to you, and if your computer's hardware environment and the operation system are matched with the designed ones, the software will work in the manner as described in Development Manual.
- j) Magnetic media which solidify this software do not have remarkable flaws in terms of the material and the technique.
- k) ZKFinger Doggle dose not have remarkable flaws in terms of the material and the technique.

6. Non-guaranty

ZKTeco Inc. dose not guarantee that all its products will meet your requirements and dose not guarantee that its operation will not be interrupted or no error will take place.

To the degree within the range of laws, in particular ZKTeco Inc. dose not undertake any special guaranty not set out herein, and any connotative guaranty. The guaranties include, but are not limited to, connotative guaranty related with sales and for any specific purpose.

7. Preparation Limit

If the guaranty is violated, the only responsibility ZKTeco Inc. should take is to replace or repair the product. If any product or its parts do not meet the above-mentioned limited guaranty requirements, free service will be provided. Repair Guaranty Claim Form must be filled in during the guaranty period, or persuasive proof, in addition to the Repair Guaranty Claim Form should be submitted to ZKTeco Inc. on the day when the malfunction is discovered. All the products should be returned to ZKTeco Inc. from the place where the products were purchased, and the returning side should be responsible for the transportation and guaranty fees. The product and its parts should be returned

together with your receipt copies.

8. **Removing indirect damages**

The both parties admit of the inherent intricacy of this software and the products of ZKTeco Inc., and acknowledge that the software and the products can not be perfect without any false. ZKTeco Inc. will not be responsible for any cost or damage (including indirect and specific damages) brought to you, your seller, your software users, or the third party, caused by any behavior and conduct whether it is violation and negligence or not according this agreement or any other specifications. The cost and damages include, but are not limited to: any commercial cost, beneficial damage, the damage or loss of data, or the loss of files, due to the usage of this software, or products of ZKTeco, or your software programs, even ZKTeco Inc. has been recommended about the possibility of the damages.

9. Responsibility Limit

Under certain circumstances, in spite of the terms and conditions set out in this agreement, if ZKTeco Inc. is still responsible for the damage caused by any malfunction and disqualification of the product, then the total responsibility fee for each product which has problems, will not surpass the fee paid by you to ZKTeco Inc. for the purchase of the product.

10. **No other guaranty**

Except what has been specified herein, ZKTeco Inc. will not provide any other specific or connotative guaranty, and will not be responsible for any products described in the prelude of this agreement, including their quality, performance, and the flexibility for a specific purpose.

11. Termination

If you can not abide by the terms and conditions of this agreement, your license and this agreement will be terminated. In this case, according to this agreement of ZKTeco Inc.:

- l) The license authorized by this agreement to you will be invalid, and you can not continue to use (including transferring the license) the licensed software and any other licensed products.
- m) You should immediately return to ZKTeco Inc. any tangible properties and their copies which represent ZKTeco Inc.'s intelligence property, or you should delete the electronic forms of this type of information.

In spite of the termination of this agreement, Clause 1, 4, 5, 6, 7, 8, 9, 10 and 11 will preserve.

12. Domination Law and Jurisdiction Right

This agreement is under the supervision of the laws of the People's Republic of China, and only Chinese courts possess the jurisdiction right for any conflict and discord aside from this agreement.

ZK Automation System Ltd., Inc., Beijing

8. Software After-sale Service

Thanks for your attention to our products. For the convenience that we are capable of providing you with our perfect service, please log on our technical forum, and complete registration information, so that we can contact you in time.

We are available from 9:00 a.m. to 6 p.m. from Monday to Friday, there are persons on duty on Saturdays, and statutory holidays and Sundays are excluded.

Your calls or emails are welcome at any time, and we are ready to solve problems for you.

Before you are prepared to call us, please first confirm that you have completed all operation procedures according to the manual, and that you have close any other application programs you have used.

We can be contact at:

Address: No. B02,Room 20-22 the Third period of Software Park,Jimei District, Xiamen,Fujian China

Zip Code: 361000

Telephone: 0592-7791134 ext. 3127

E-mail: sdksupport@zkteco.com

If you have got any technical problems concerning this set of products, please prepare the following information, so that we are able to remove problems for you within the shortest period:

- 1、 Software name
- 2、 The configuration of your computer (including the brand, computer type, CPU, EMS memory, disc driver, main board brand, and so on).
- 3、 XP\Vista\7\8\10 or any other environments
- 4、 any application programs you are using
- 5、 Detailed description about the problem

You are welcomed to log on our website: www.zkteco.com, visit our technical forum and submit your questions and suggestion. We will try to provide you with satisfactory replies in the soonest manner.