UTP-BUILDING ACCESS CONTROL USING FINGERPRINT IDENTIFICATION SENSOR

By

FAIZUL AKMAL B. HAMIDI

FINAL REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

UTP-BUILDING ACCESS CONTROL USING FINGERPRINT IDENTIFICATION SENSOR

by

Faizul Akmal b Hamidi

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

DR ABDALLAH BELAL ADAM
Senior Lecturer
Electrical & Electronics Engineering Program
New Academic Block 23-3-25
Universiti Teknologi PETRONAS
31750 Tronoh
Perak Darul Ridzuan, MALAYSIA.

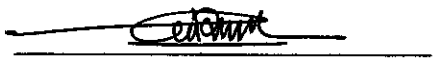
Dr. Abdallah Belal Adam
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2005

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Faizul Akmal b Hamidi

ABSTRACT

Today, security and identification of people are greater challenge in the whole world. Fingerprint sensor is one of the tools that have and may have remarkable contribution in overcoming these challenges. In University Technology Petronas (UTP) right now, the staffs and workers are using card to access the building and fill their name in the book for the working time sheet. This application is not suitable while UTP is one of the best universities in Malaysia and provided the most modern building and labs. This application is also not appropriate because it is hard for the administrator to identify which staffs didn't come to the office and to record the working time sheet for the staffs. There is actual need in innovating smart building access control using the fingerprint sensor. The sensor used in this project was U.are.U fingerprint sensor and the software was the BioKit SDK. The database and the related GUI layout interface were created using the Visual Basic and the Microsoft Access. The 5 user's data including names, ID number and all related information plus the fingerprint string were collected by using the fingerprint sensor and stored in the database. To access the building, the person will be examined by pressing his/her finger on the sensor, which will be captured and compared the scan's picture with the stored string data. Upon this comparison the sensor will gives electrical signal, which confirmed or rejected the person fingerprint template. Then the electrical signal was sent to the circuit to activate the locking system of the door. According to it, the door will open or remain close, while the detail information including the time is displayed, saved and be ready for administration officer to check.

ACKNOWLEDGEMENTS

Upon completing the Final Year Project (FYP) of UTP-Building Access Control using Fingerprint Identification Sensor, the author would like to praise to the Al-Mighty Allah for giving him the chance to finish the project. Special appreciation and thankful also dedicated to the FYP supervisor, Dr. Abdallah Belal Adam for his supervision, commitment, professionalism, advice and guidance for completing this project. The gratitude is also dedicated to Mr. Bushara, a postgraduate student in Electrical and Electronic engineering of University Technology Petronas for helping the author a lot in the programming codes. Also special thank to the author's partner, Miss. Asiah Bt. Hanapi for being very helpful in supporting and giving the ideas to the author in finding and researches the project.

The author also wants to give deepest gratitude to Electrical and Electronics Department for the support and not forget also to Electrical Technician, Miss Siti Hawa, the author's parents and colleagues for all the encouragement. Last but not least, to those who help directly or indirectly in completing this project, thank you very much, your contributions are highly regarded and would be remembered.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

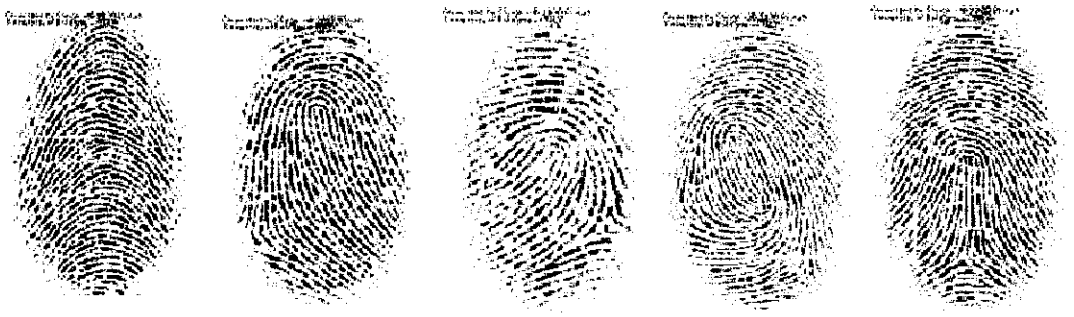
1.1 Background study

Biometrics is an automated system of recognizing a person based on the person's physical or behavioral characteristics. It is the same system that the human brain uses to recognize and distinguish one person from another. It is a system that recognizes a person based on "who" the person is and does not rely on "what a person is carrying" or "what a person knows." Things that a person can carry, such as keys and ID-badges, can be lost, stolen, and/or duplicated. Things that a person knows, such as passwords and pin-numbers, can be forgotten, stolen, or duplicated. Instead, biometrics relies on "who" a person is-on a unique immutable human characteristic that cannot be lost, forgotten, stolen or duplicated. Biometrics, therefore, provides the ultimate level of security, convenience and ease of use. It is this security and conveniences that fingerprint recognition system provide [1].

The project emphasizes on how the fingerprint identification system work and its application. The fingerprint identification and verification remain as one of the most prominent and reliable biometric identification methods. Fingerprints are used in variety of applications such as criminal investigations, access control system, national frontier control and many more.

A fingerprint is an impression of the underside of the end of a finger or thumbs; used for identification because the arrangement of ridges in any fingerprint is thought to be unique and permanent with each person [2]. A fingerprint is composed of valley and ridges lines. They follow a pattern. Everyone is known to have unique fingerprints. No two people have been found have the same fingerprinted even twins [3].

Figure 1 below shows the general shape of the pattern of fingerprint. The shape is classified according to five classes:



Arch Left loop Right loop Whorl Tented arch

Figure 1 : The pattern of fingerprint [4]

The second features are the cores and deltas. The core is located by the square while the delta is located by a triangle shows in the Figure 2.

Figure 2 : The core and delta [4]

From the references [4], it draws and described the differences between the shapes. It mentioned that, for the arch shape, it has no delta and cores. The right loop shape only has one core and the delta was situated at the right side while the left loop, the delta was situated at the left side. The whorl shape has two cores and two deltas and for the tented arch, it has one core while the delta is at the middle.

The minutiae are characterized by both their X-Y coordinates and the angle of the general direction of the ridge in this point characterized the minutiae.

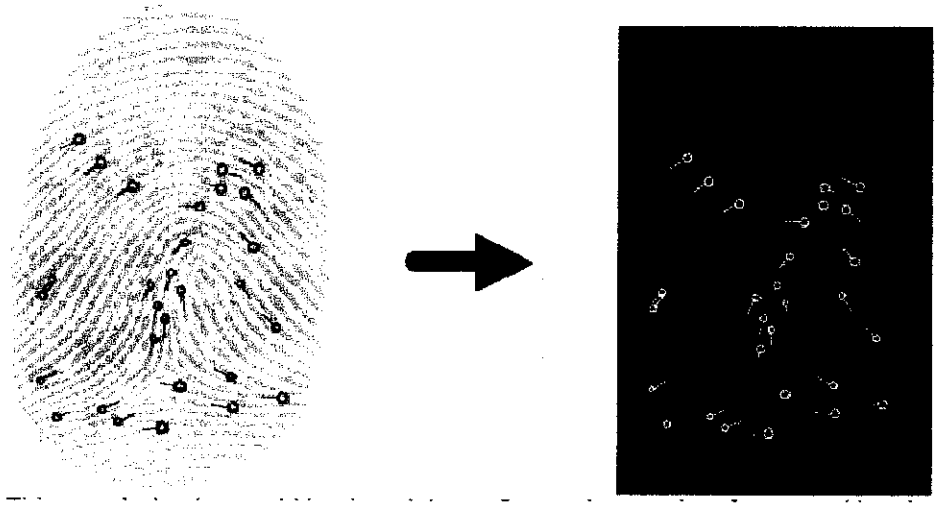


Figure 3 : The minutiae of the fingerprint [4]

Figure 4 shows the minutiae of the fingerprint. This set of minutiae could be the minimum fingerprint template of recognition. In order to increase the performance, the texture vectors should been add. Each minutia is related to the vectors, which describe the frequencies of the ridge, in few directions of the minutiae. This vector is used when the number of the minutiae is low, which insures the better matching process [4].

Fingerprint identification is divided into two systems. The systems are Automatic Fingerprint Identification System (AFIS) and Automated Fingerprint Authentication System (AFAS). The Automatic Fingerprint identification System (AFIS) is used and developed for matching process by replacing the manual matching and used to specify one person's identity. AFIS uses automatic scanning devices that convert the image of a fingerprint into digital minutiae that contains data showing ridges at their

points of termination (ridge endings) and the branching of ridges into two ridges (bifurcations) [1].

The Automatic Fingerprint Authentication System (AFAS) is used to verify the authenticity of one person by her fingerprint. The user provides her fingerprint together with her identity information like her ID number. The fingerprint verification system retrieves the fingerprint template according to the ID number and matches the template with the real-time acquired fingerprint from the user. This type of fingerprint will be used in this research project [1].

1.2 Problem statement

In University Technology Petronas (UTP) right now, the employees are using card to access the building and fill their name manually in the book for the working time sheet. This application is not suitable while UTP is one of the best universities in Malaysia and provided the most modern building and labs. This application is also not appropriate because it is hard for the administrator to identify which staffs didn't come to the office and to record the working time sheet for the staffs. Therefore an advance smart system is required to solve this problem. The project is to design a new system using fingerprint identification sensor for the UTP-building access control. The data of the staffs will be stored in database and collected using the fingerprint of certain staffs. This fingerprint identification system will be suitable to record the working time sheet for the staffs and for access the building.

1.3 Objectives and scope of works

The objectives of this project are as follows:

1. To understand the biometrics and the fingerprint itself.
2. To determine the most suitable sensor and the software which has the capability to register, store and match the fingerprint with the database.
3. To improve and design the system for accessing UTP-building using fingerprint sensor.
4. To collect a data from at least 5 users and used it for the identification.

The scopes of this work are shown below:

1. **Data collection for the research problem**

   To find any appropriate data and information from the Internet, books, journals and magazine regarding the research problem.

2. **Identification of equipment and software.**

   To specify the equipment and software to be used such as fingerprint sensor and Software Development Kits (SDK).

3. **Equipment setup**

   To set up the equipment and understand how it works.

4. **Testing the system**

   To test and implement the system that has been designed with 5 samples of fingerprint images.

CHAPTER 2
LITERATURE REVIEW/ THEORY

2.1 The sensor

The electronic sensor could be manufactured in different ways. Sensors could be capacitive, optical, and thermal or pressure sensitive. In fact, all of those sensors are able to issue image where the ridge is darker than the valley.

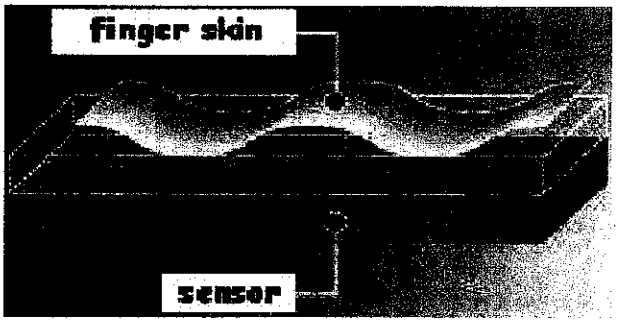


Figure 4 : The sensor [4]

Nowadays, the most common sensors on the market are the capacitive and the optical sensors. The optical sensor is the best regarding image quality, but its cost is high and it could be fooled various fake finger [4].

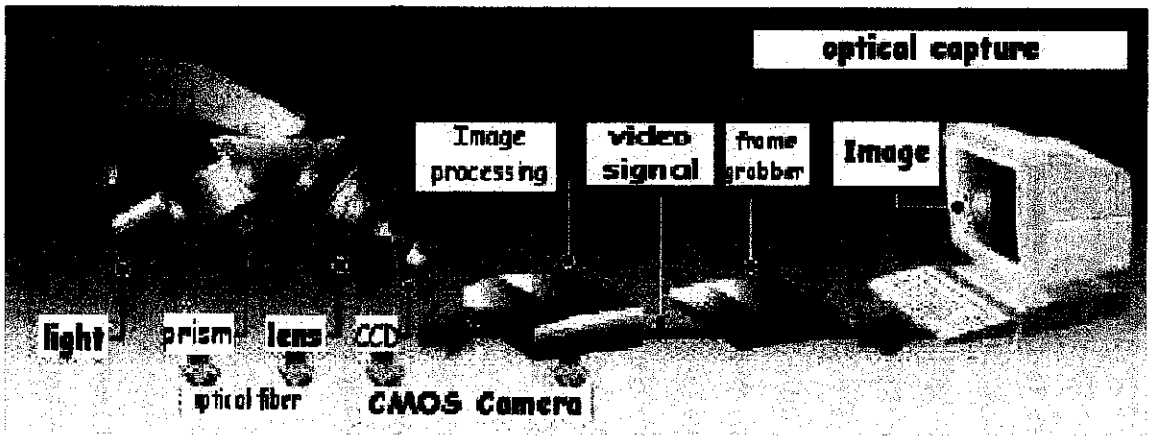


Figure 5 : The optical sensor [4]

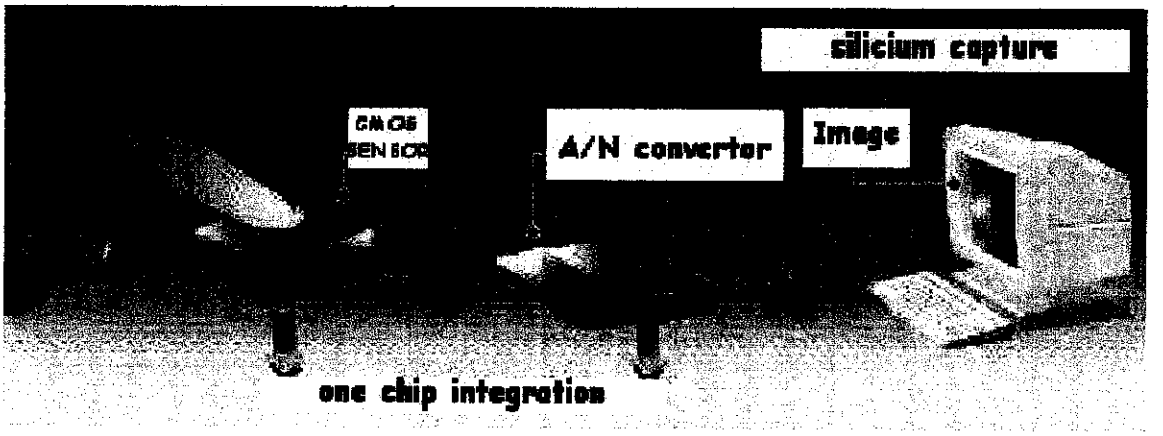


Figure 6 : The capacitive sensor [4]

The choice of the sensor depends on the range of the application. For large-scale application, the capacitive sensors are the best.

2.2 Graphical User Interface (Microsoft Visual Basic Environment)

A GUI is a graphical, (rather than purely textual) user interface to a computer. The term came into existence because the first interactive user interfaces to computer were not graphical; they were text-and-keyboard oriented and usually consisted of commands that we had to remember and computer responses that were infamously brief. The command interface of the DOS operating system is an example of the typical user-computer interface before GUIs arrived. An intermediate step in user interfaces between the command line interface and the GUI was the non-graphical menu-based interface, which allows interaction by using a mouse rather than by having to type in keyboard commands [6].

The Microsoft Visual Basic is known to be among the most popular choice to create windows GUI. In Visual Basic, new windows created are called forms. Elements (such as text boxes and buttons) that are placed inside a form are called controls. The Visual basic allows event-driven programming, where the user's actions cause events and each event in turn triggers a procedure that is associated with it [6].

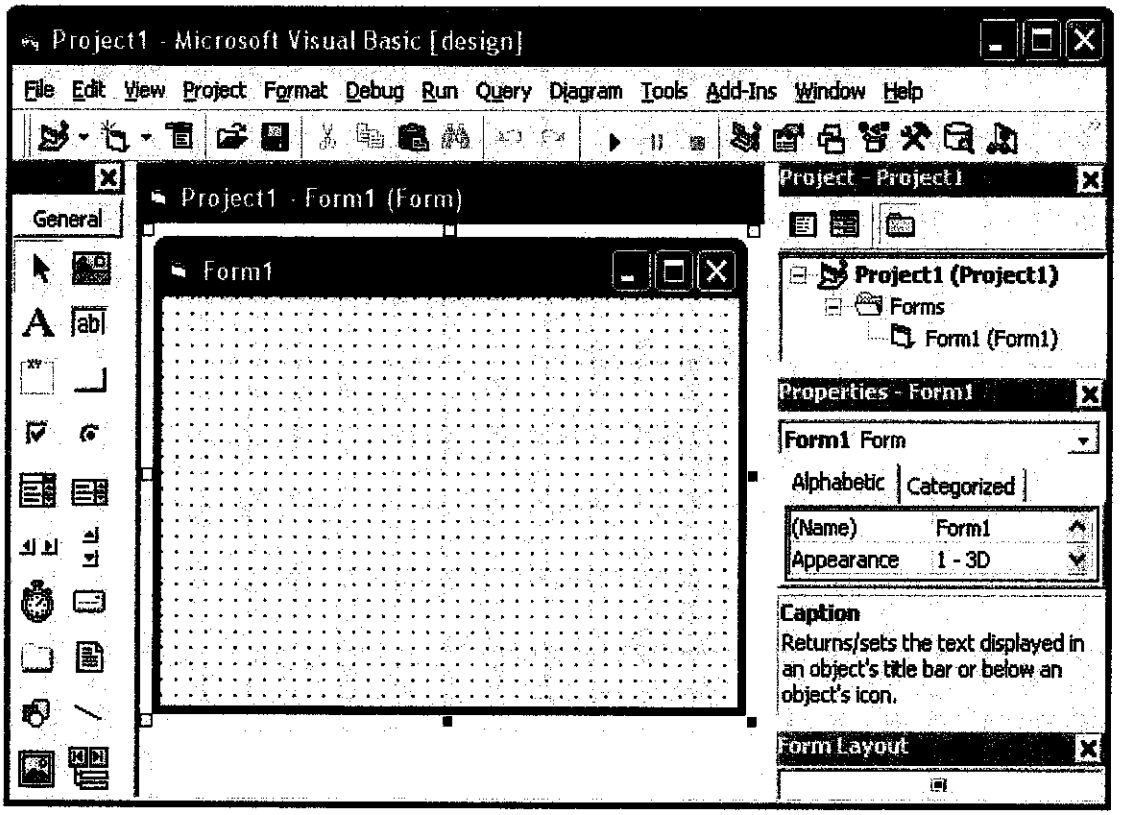


Figure 7 : Visual Basic Editor [6]

The properties of creating an object model in Visual Basic are:

- Object have properties and methods
- Forms and controls are example of objects
- Properties describe an object.
- Methods are actions associated with an object

In order to write a proper Visual Basic project, there are several important elements to learn and understand. The two vital steps are:

1. Planning

- Design the user interface
- Plan the properties
- Plan the Basic code – procedures are associated with the events, actions written in pseudocode.

2. Programming

- Define the user interface – define objects
- Set the properties
- Write the Basic code

Table 1 : Visual Basic Naming Conventions [6]

Object	Prefix
Form	frm
Command Button	Cmd
Text Box	Txt
Label	Lbl
Option Button	Opt
Checkbox	Chk
Frame	Fra
Image	Img
List Box	Lst
Shape	shp

2.3 Database Information

The usage of database has become the norm when it comes to presenting information in most system and application. When the record keeping needs outgrow a filling cabinet or bindle of index cards, a computerized system can help to manage the growing quantity of data, enable more effective usage of data. Spreadsheet program can manage small, simple database. Desktop programs such as Microsoft Access and File Maker can handle databases too big or too complicated for a spreadsheet. Larger

databases, databases accessed by many people at once and database that feed client/server and web applications need real database servers [7].

The relational model, invented by IBM researcher Ted Codd in 1970, was not turned into a commercial product until almost 1980. Since then database systems based on the relational model, called relational database management systems or RDBMSs, have come to dominate the database software market. Today few people know about any other kind of database management system [7].

The benefits of a good RDBMS and a well-designed relational database:

- Data integrity and consistency maintained and/or enforced by the RDBMS
- Redundant data eliminated or kept to a practical minimum
- Data retrieved by unique keys
- Relationships expressed through matching keys
- Physical organization of data managed
- Optimizations of storage and database operations execution times

Examples of well-known industrial-strength relational RDBMSs include:

- Oracle
- Microsoft SQL Server
- IBM DB2
- Informix

Well-known PC-based (desktop) relational RDBMSs include:

- Microsoft Access
- Microsoft FoxPro

2.3.1 Microsoft Access

Microsoft Access is a relational database management system (RDBMS). At the most basic level, a RDBMS is a program that facilitates the storage and retrieval of structured information on a computer's hard drive [7].

The Microsoft Access package contains the following elements:

- A relational database system that supports two industry standard query languages: Structured Query Language (SQL) and Query By Example (QBE).
- A full-featured procedural programming language – essentially a subset of Visual Basic
- A simplified procedural macro language unique to Access
- A rapid application development environment complete with visual form and report development tools.
- A sprinkling of objected-oriented extensions
- Various wizards and builders to make development easier

When designing database in Microsoft Access, it is important to understand the nature of database designing. The three different view of computing in Microsoft Access, better known as 'personalities' are:

- The relational database personality; viewing the application as sets of data
- The procedural programming personality; viewing the application as commands to be executed sequentially
- The object-oriented personality; viewing the application as objects which encapsulate state and behavior information

Since there are often several vastly different ways to implement a particular feature in Access, recognizing the different personalities and exploiting the best features and avoiding the pitfalls of each are important skills for Access developers. The

advantage of these multiple personalities is that it is possible to use Access to learn about an enormous range of information systems concept without having to interact with a large number of 'single-personality' tools [7].

2.4 PIC16F84A Microcontroller

The PIC16F84 belongs to the family of low-cost, high performance, CMOS, fully static, 8-bits microcontroller. All PICmicro™ microcontrollers employ an advanced RISC architecture. PIC16F84 have enhanced core features, eight-level deep stack and multiple internal and external interrupt sources. The separates instruction and data buses of the Harvard architecture allow a 14-bit wide instruction word with separate 8-bit wide data bus. The two stages instruction pipeline allows all instructions to execute in a single cycle, except for program branches (which require two cycles). A total of 35 instructions (reduced instruction set) are available. Additionally, a large register set is used to achieve a very high performance level [8].

PIC16F84 microcontrollers typically achieve 2:1 code compression and up to 4:1 speed improvement (at 20 MHz) over other 8-bit microcontroller in their class. The PIC16F84 has up to 68 bytes of RAM, 64 bytes of Data EEPROM memory and 13 I/O pins. A timer/counter is also available [8].

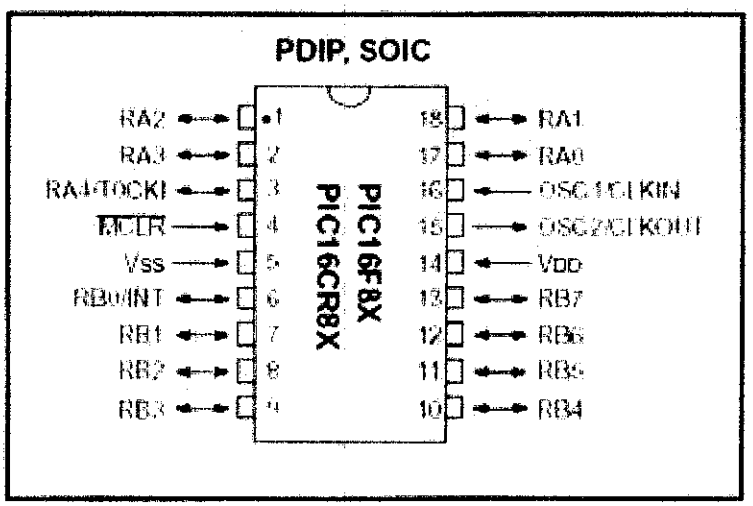


Figure 8 : PIC16F84 pin out [8]

CHAPTER 3
RESEARCH METHODOLOGY

3.1 The flow of the project

For the first step, the research on the topic about the fingerprint must be conduct and then understand the information. The sensor also must be searched and needed to identify which sensor will be used. The sensor that will be use is U-are-U sensor and BioKit SDK. The software needs to be understood and learned how to use it. For this software, it's use visual basic programming. The coding of the programming was needed to understand and modified it.

Then, the good database is settled up to store the all the information required. The database was created using the Microsoft Access. All the information that need to put in the database must be collected and arranged in properly so that it's easy to understand. The tables that stored all the information and used in the database also must be created. All related database files were stored in a single global file in .mdb format. The model used for creating the database is relational database model. This model consists of three components which are structural part, manipulative part, and a set of integrity rules. Structural part defines how the database is to be constructed while Manipulative part defines the type of operations that are allowed. To ensure the data is accurate, the model needs a set of integrity rule.

The last part is to analyze and test the system by conducting an experiment within 5 users. All their information such an ID number, name and fingerprint's string were stored in the database.

Next, the registration process and the matching process are modified from the sample software so that it is suitable for the project. Then the system is executed to test the functionality of it. The flow chart of the project is shown below

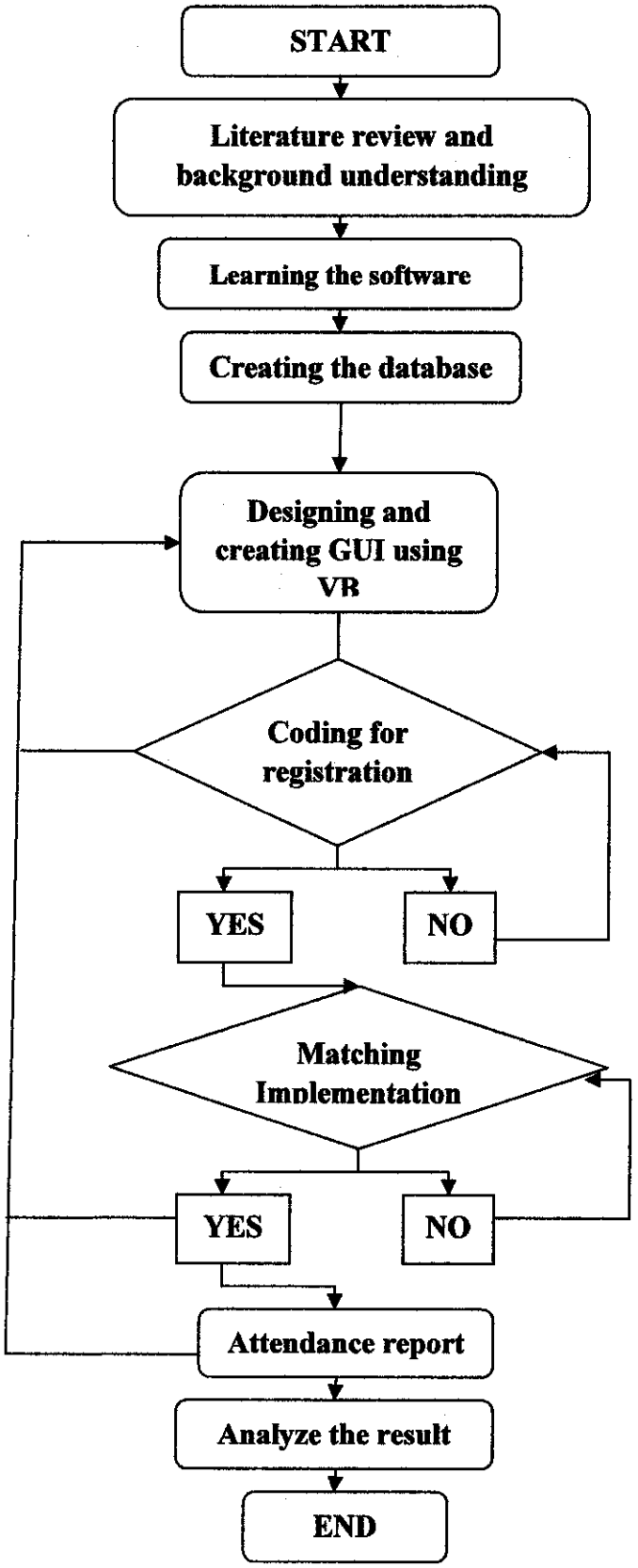


Figure 9 : The flowchart of the project

3.2 The sensor and its software

For this project, a U-are-U sensor is used from Kinetics Bio Base Solutions (M) Sdn. Bhd. The sensor was purchased with its own software, BioKIT SDK (Software Development Kit). So, for this project onward, the author will deal with the sensor and its software. The U-are-U 4000 is a USB driven fingerprint sensor. The user simply places his/her finger on the glowing sensor window, and the device quickly and automatically captures the fingerprint image. On-board electronics calibrate the device and encrypt the image data before sending it over to USB interface. U-are-U 4000 sensor utilize optical fingerprint scanning technology for superior image quality and product reliability. This U-are-U 4000 Sensor has an unmatched ability to recognize even the most difficult fingerprints. The features of the software are it is in small form factor, excellent image quality, encrypted image data, latent print rejection, counterfeit image rejection, rotation invariant, works well with dry, moist, or rough fingerprints, compatible with all U-are-U applications and the drivers support Windows 98, Me, NT 4.0, 2000, XP.

The BioKit SDK components have three stages of process. The first process is the registration. The registration component extracts the fingerprint raw data from the sensor device before generates a register fingerprint templates. It will obtain a several fingerprint capturing and adapting to the quality of the fingerprint that is being imaged. This process consists of extracting features from each print and combining them into one robust template. Finally, it returns a hexadecimal string, which consists of the fingerprint details. The coding of the registration process is shown in Appendix A.

The second process is the matching. The matching component compares registered template that generate from the registration component with the one provided by the user via the sensor device. If the matching of the fingerprint is authenticated, a success status will be returned. The coding for the matching is shown in the Appendix B.

The last process is the hardware information. Basically, this component does nothing more than retrieving the related hardware information from the sensor device. Information can be accessed directly after the process being initialized as an object.

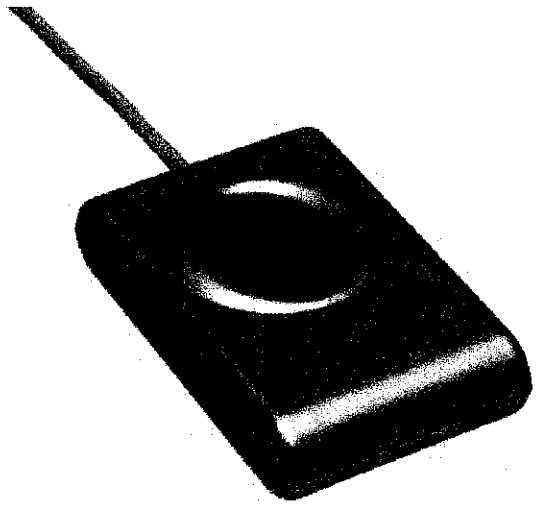


Figure 10 : The U-are-U sensor

CHAPTER 4
RESULT AND DISCUSSION

4.1 The Designing Database

In order to store all the information of the users or staffs, the complete and perfect database must be created. This is to make sure all the gathered info can be retrieved back and can't be stolen or loss. To create the database, the Microsoft Access has been used as the based for the database. First thing to do is to create tables for storing the information. Three tables were required for the information. Figure 11 below illustrated the table for keeping the information of the users or the staffs. The table consists of the name of the user, the Staff ID number, left fingerprint string, right fingerprint string and the building location.

Staff : Table

	StaffNo	StaffName	StaffIDNumber	LeftFingerPrint	RightFingerPrint	BuildingNo
+	1	Faizul Akmal bin Hamidi	2564	622599651902F	622599651902F	2
+	2	Nor Sulyanie binti Sulong	2768	622599651902F	622599651902F	3
+	3	Intan Shakira binti Farouk	2566	622599651902F	622599651902F	17
+	4	Siti Sakinah binti Sari'at	2565	622599651902F	622599651902F	21
+	5	Mohd Hasanul bin Mashod	2567	622599651902F	622599651902F	22
+	6	Mohd Aminullah bin Razak	2562	622599651902F	622599651902F	23
*	(AutoNumber)					0

Record: 2 of 6

Figure 11 : Table staff

From the Figure 12, it shows the second table which was used for keeping the time taken and the user that accessing the building. The table consists of the staff information and the time in and time out for the users.

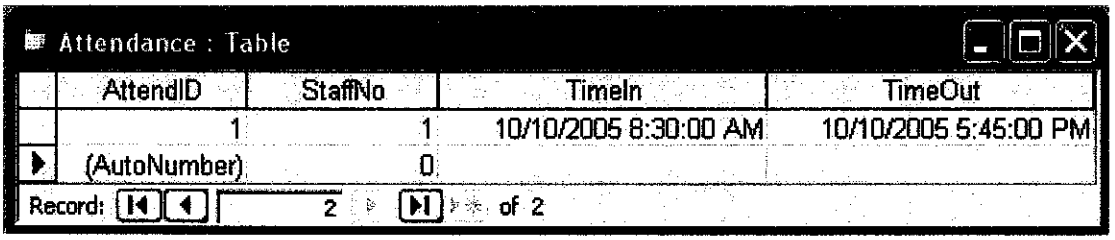


Figure 12 : Table Attendance

Figure 13 below shows the third tables, which was used to specify the building where the staffs or users are located. All the tables in the database are link with each other so that it will be easy to get the information

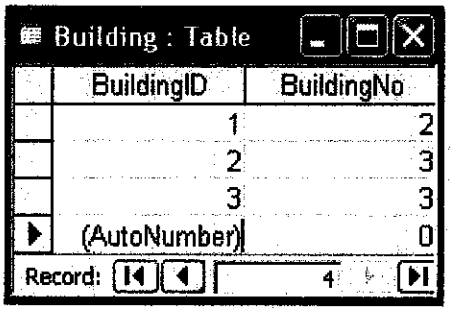


Figure 13 : Table Building

4.2 The Registration Process

In BioKit SDK, it usually transfers the fingerprint image as a string as shown in Figure 14, but it's not stored and appeared in the database. The user needs to copy the string and paste it at the matching procedure to get the verification of the fingerprint. In order to keep and save this string, the coding of the registration process needs to developed and modified so that all the output string will directly stored to the database. The modified coding for the registration process is shown in the Appendix C.

622599651902FE14E11635558F20E6AB1498241D3659A8E546535E497
60195E1C5947C4B778BE7A2961463BB558AB17CAE1C2A2EA571FE
EF010FEC8F76D194A916A085115739425F971FA989E1CE28C15A084
44F6D7AF2550AC68C5A17EBA3BD1855D91A5BAFEA139C2170349
BAD7E9307A4928AE9DDD127A995AC0E756F4BFFF506A914C94FE
D5FAD4878337EE285806708FFD7764BB9BACF5F30EB11E5C087259
D8577089216E66BF3612FCBCC008333B10D15A8F4DDF5F7FBDDD4
0C99F4EB33F6E1F11D858EFA6E43C7D6657E0833CB4FA4A6092F83
75F0D0A9D4F5674FD3D92E3EB8A4AF2B2ED4053A5190ADC8FC36
B340390CF681DAF592B67BCA6FFC87CE7974E8B42516DF482183CE
027AFB770835E39EF945197D7C3F90433C9D254BB280054348C2EE6
973D3A119E9D107C2E84F5D2F8153EF3573D3A119E9D107C23ED30
3C0527EE11BAFEFE10C2B0F11D8001FE391118FE2A2FD6040180595
6E9728189B9FDF27F4F4FD60401805956E97574280185FFAC56EFD60
401805956E973BE7967DE83EF0D8FD60401805956E97AD111044FE65
778CFD60401805956E97B55059F12A44DEE7FD60401805956E973E7E
4ACD2AE65AADEAE87DA0599095C0783A820C783958E597CE8311B
ECDFC830835E39EF945197DE1FB0D7098E1D1C17E599F0DA15C57
DDF41FA264C0A401CA1D2E64838F5CC53AEC5DDF376E1FA7D096
F302F3D710EDAE520E6BA9D8DDF92183C1992A324A7F4BD3AB9C
D625C4045E0DC35B59C3C2EB1B444FB6503E1A0D8D41C4DC04A85
8DB6F1C8340598D02163C2880E21DA391CA87EC5E236237D5644F1
D6C39F0A515F0F5B5FDC04A957821521EF0158A1B9EE037B431053B
CCAA8B5E

Figure 14 : The Hexadecimal string

In the registration process, the user can choose to register whether his/her right finger or left finger. The user also can register both of the fingers. It is recommended for the user to register both of his/her finger. The reason is that if something happened such as the sensor can't detect and read the right fingerprint for example, the user has another option which to use his/her left fingerprint.

In order to complete the registration process, the user is required to press his/her fingerprint on the sensor four times as shown in Figure 15. This is to make sure that the software can take the average string's values for the fingerprint. The reason is that, when the matching process is done, the software can compared and matched the fingerprint with the approximately string from the database. The sensor also can detect the false if the user is press another finger in the registration process. The registration is not done will appear to show that the registration process is not complete.

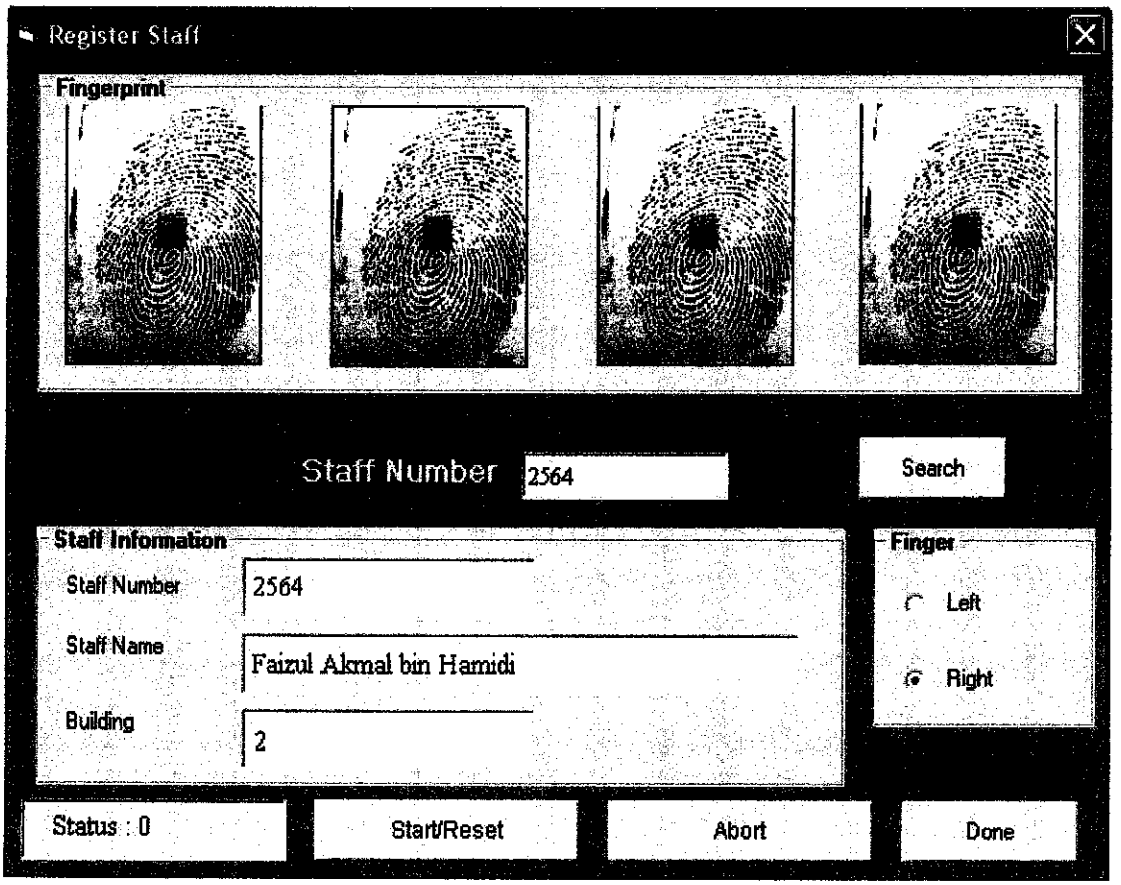


Figure 15 : The Registration Process

4.3 The Matching Process

In order to verify the fingerprint and access the building, the matching process is needed for identifying and matching the fingerprint with the database. The user needs to press his/her fingerprint on the sensor. The sensor will capture the fingerprint and compared it with the string from the database for identifying the fingerprint. If the owner of the fingerprint has registered before, then the fingerprint will be identified. The LED 1 which is green from the circuit will appeared and the door will unlock so that the user can access the building. If else, the door will remain closed and the LED 2 which is red will on. The modified coding is shown in the Appendix D.

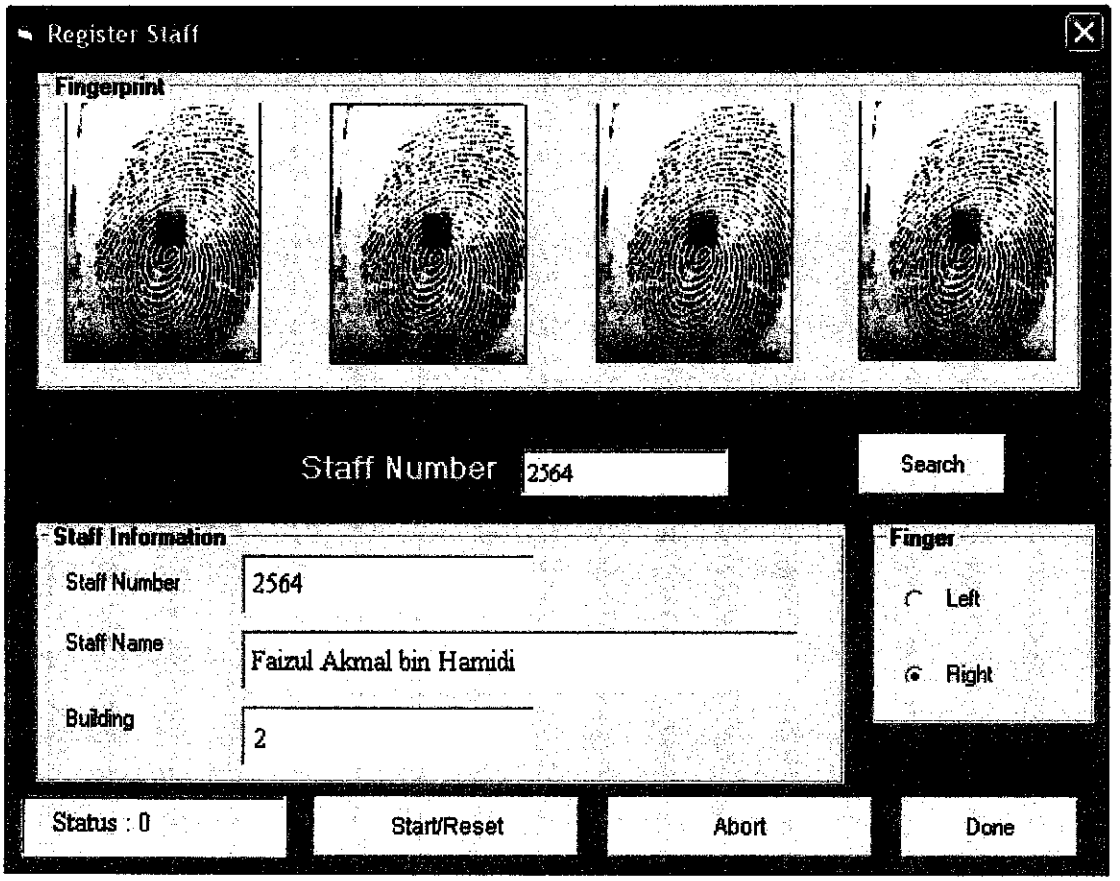


Figure 15 : The Registration Process

4.3 The Matching Process

In order to verify the fingerprint and access the building, the matching process is needed for identifying and matching the fingerprint with the database. The user needs to press his/her fingerprint on the sensor. The sensor will capture the fingerprint and compared it with the string from the database for identifying the fingerprint. If the owner of the fingerprint has registered before, then the fingerprint will be identified. The LED 1 which is green from the circuit will appeared and the door will unlock so that the user can access the building. If else, the door will remain closed and the LED 2 which is red will on. The modified coding is shown in the Appendix D.

4.4 The GUI Layout Design

4.4 The GUI Layout Design

For making the software more interesting and user friendly, the GUI layout must be designed so that it will look more interesting and easy for used. The designing of the GUI layout must be done carefully and tidily to attract the user to use the software and to know more about it. Figure 16, shows the main GUI Layout Interface for this project. The interface will appear first when the users click the program. The GUI layout shows the name of the university, Universiti Teknologi Petronas (UTP), the title of the program which is the smart UTP-building access control and the logo of the UTP. In the GUI layout also, the user can see four main buttons which are match, admin, report and exit. For the admin button, it only can be used by the administration for registration, addition or replaced a new employee or to update the information. The match button is used for the matching process while the report button, it is also can be used by the administration to print the report for the working time sheet of the employees. The exit button is used to exit the program.

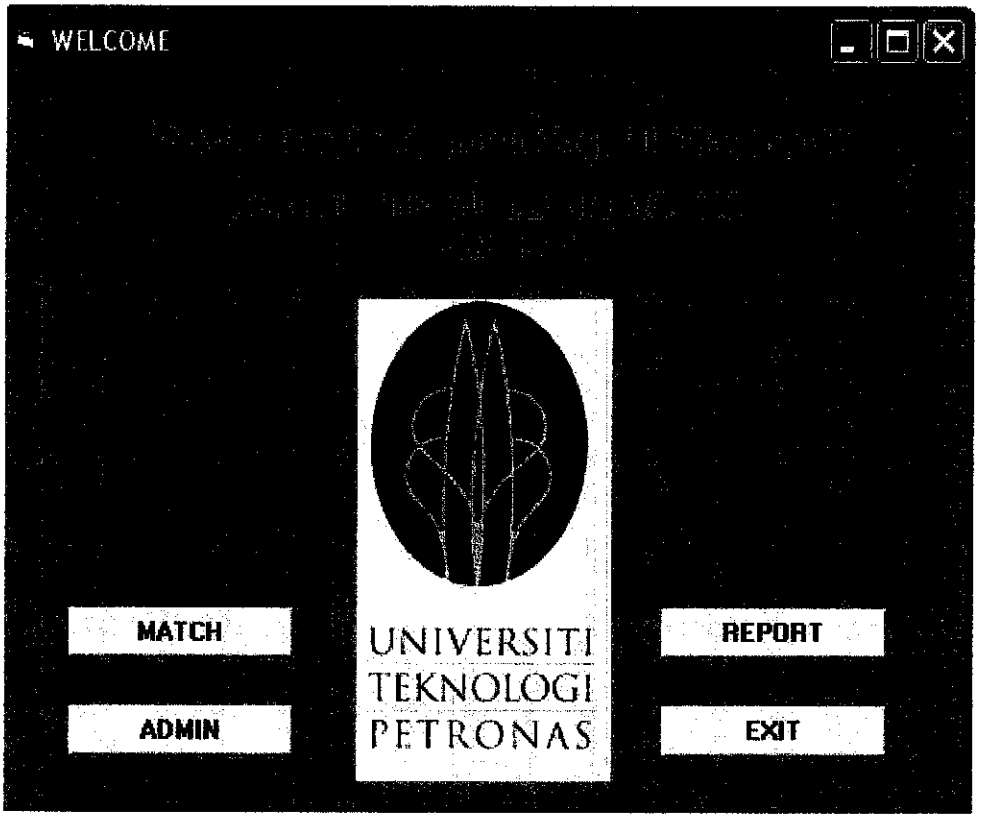


Figure 16 : The main GUI Layout Interface

Figure 17 shows the GUI layout for registration process. The search button is used to find the user's name. The administrator will key in the ID number of the employees and then click the search button to find the employee's name. If the employee's name is exist in the database, the employee's name, ID number and the building location will appeared in the staff's information column. For the column finger, the user can choose to register his/her left or right finger. After the information of the staff has been granted, the start/reset button can be clicked and the user needs to press his/her fingerprint on the sensor. The image of the fingerprint will appear at the fingerprint's column. After the registration process has done, the message box "Register done" will appear. If else, message box "Register is not done" will appear. The abort button is used to cancel the registration process. The done button will exit the application.

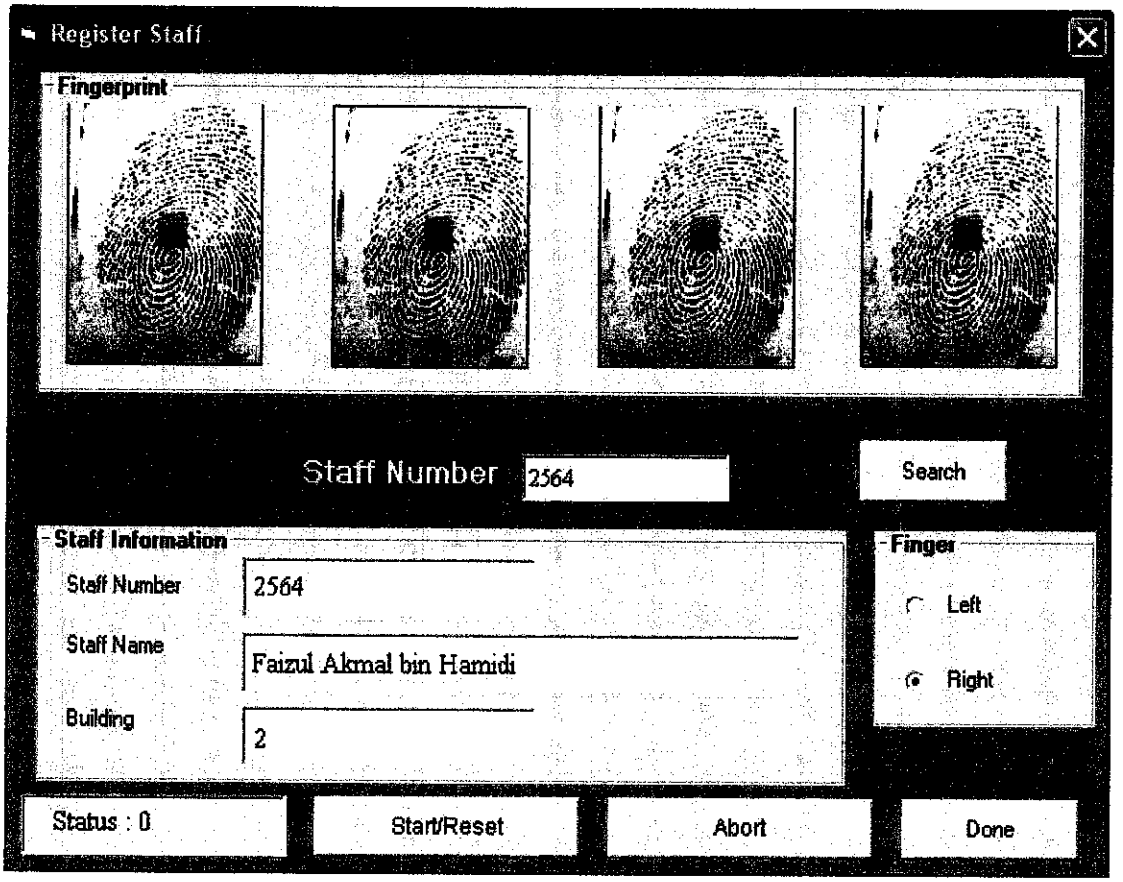


Figure 17 : The registration GUI Layout Interface

Figure 18 below shows the match GUI layout. For this process, the user can choose the building and the date for matching. For this project, the building number was set for each door. Only the date will change automatically. When the user click the match button and press his/her fingerprint on the sensor, the image of his/her fingerprint will appeared. Then the software will search the approximately the same string of the fingerprint from the database. The message box "Match successful" will appear if the matching process is done. If else, the message box "Match is not successful" will appear.

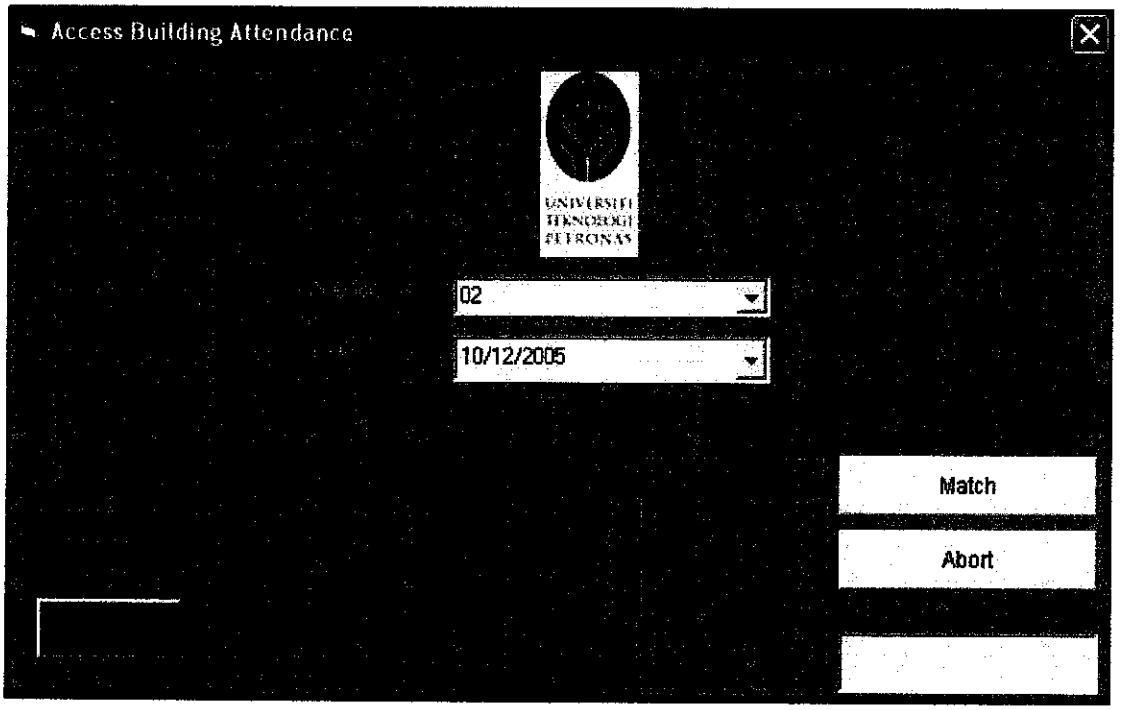


Figure 18 : The match GUI Layout Interface

4.5 The Designing Circuit

For controlling the door, the circuit must be design for the system. The door will be unlocked when the user is identified and the green LED will on to indicates the user is permitted to access the building. If else, the door will remain unlocked and the red LED will on. For the circuit, the PIC 84A is using as the interface between the visual basic programming to the circuit. When the user is defined as legal person, the visual basic will send a signal to the PIC 84A as 1. The output 1 indicates that the fingerprint of the user is matching with the stored database. The PIC 84A will receive

the signal as an input to activate the circuit and unlock the door and switch on the green LED. If else, the 0 output will be send from the visual basic to the PIC 84A and it will switch on the red LED. The Max232 does the connection from the COM port of the computer to the circuit. For the lock system of the door, the solenoid is been used to replace the magnet system. When the solenoids receive current, the solenoid will repeal from each other's and it will unlock the door.

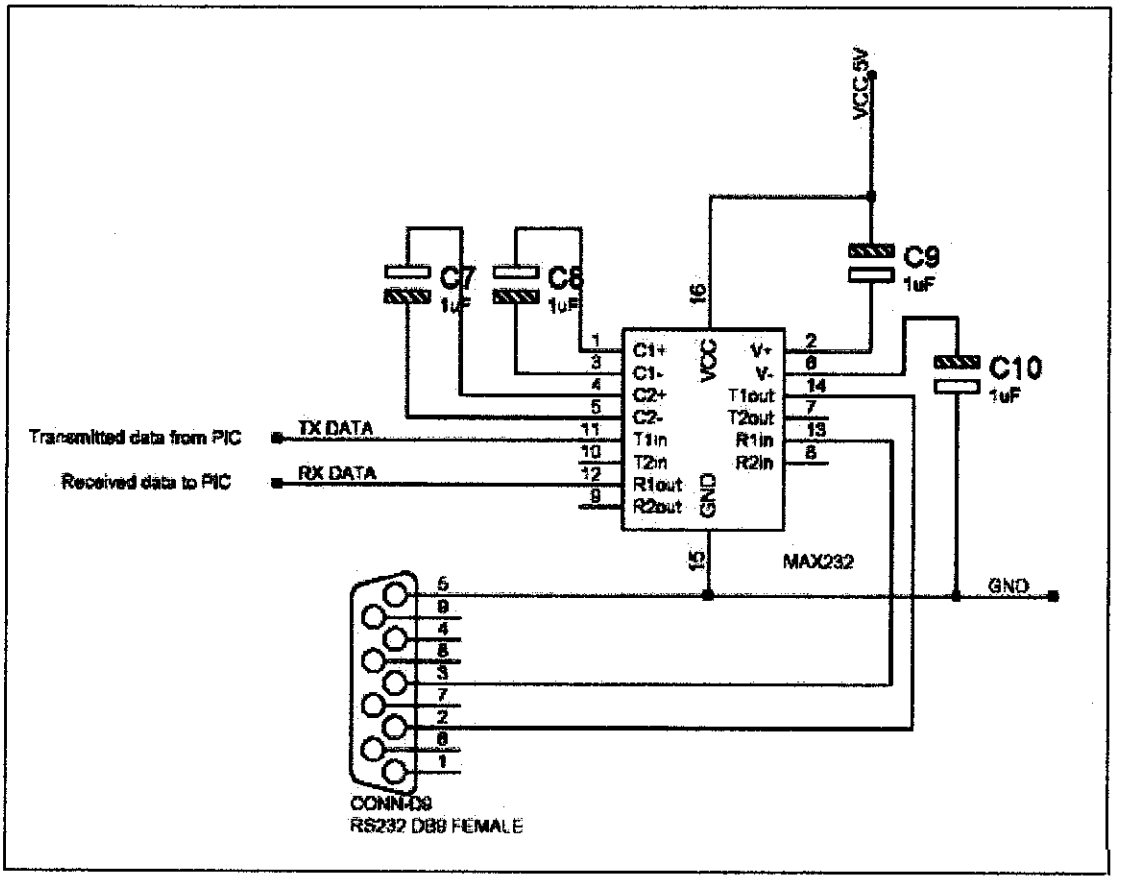


Figure 19 : The MAX232 circuit connection

Figure 19 show the MAX232 circuit connection. The RS232 port female will be connected to the com port of the computer. It will receive the signal from the visual basic whether the matching is successful or not. The signal then will send to the MAX232 in form of ASCII code. Then it will send to the PIC26F84A microcontroller for later process.

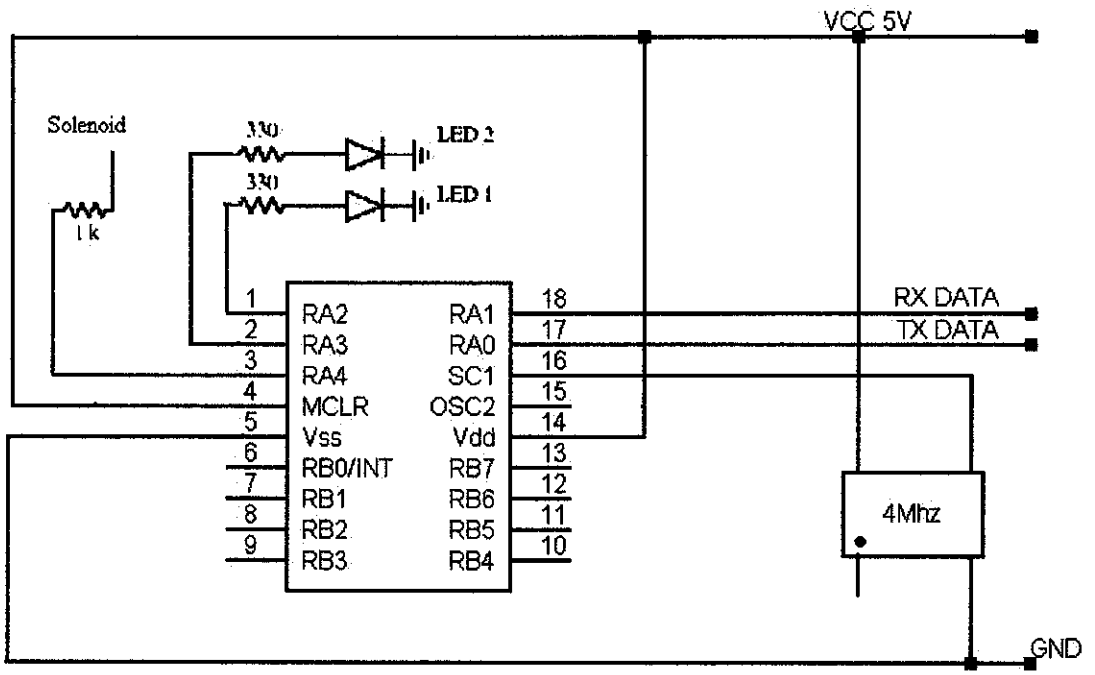


Figure 20 : The PIC16F84A circuit connection

Figure 20 shows the circuit for the PIC16F84A. The PIC16F84A will receive the data from the MAX232. If the matching is successful, the LED 1 and the solenoid will activate. If else, the LED 2 will on.

Figure 21 shows the circuit connection on the breadboard. Some components in the circuit have been added due to some problems encountered. At the end, the circuit works well.

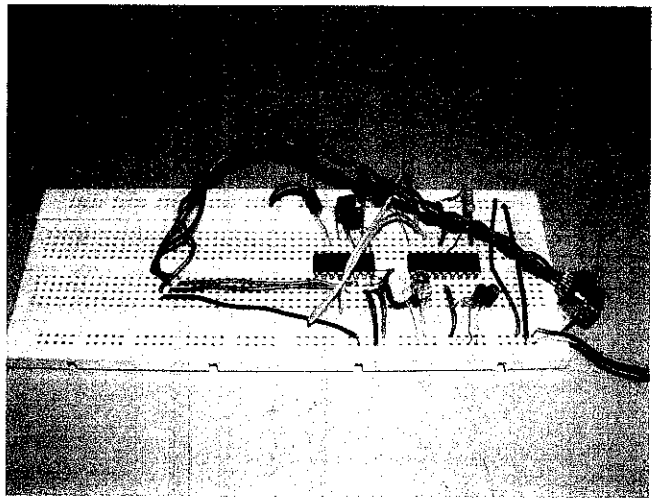


Figure 21 : The circuit connection on the breadboard

4.6 PIC16F84A Microcontroller Programming

The C programming language is chosen to program the PIC16F84. The program will flow in such a way that the PIC16F84 will gather all the eight inputs into the PIC and produce an output that includes transmitting data output to the PC and vise versa. In order to run this program smoothly, several important initializations must be included. They are:

- PIC16F84 header file
- Declare the desired fuses
- Declare the usage of RS-232 which includes the baud rate, receive and transmit pins
- Define the delay or the clock oscillation speed

Before downloading a program into a PIC, the program must be verified and compiled with the C Compiler. The program will be converted into hex file. Then, the EPROM of the PIC must be cleared and blanked. Once this stage is successful, the program can be downloading into the PIC. This was done using the WARP-13 PIC programmer.

4.7 Constraints and Problem Encountered

Naturally, it is almost inevitable to avoid problems and complications when it comes to developing a system. It can be said that the time spent on designing is almost equal to the time spent in the learning and troubleshooting phases. Most of the time used in the project is to learn the visual basic and understand the coding from the BioKit SDK. The first thing done was to understand the coding of the BioKit SDK and modified the coding. The problem is how to save the string of the fingerprint to the database. After trial and error in changing the coding, hopefully the problem was solved. Then it comes to match process, which recalls back the string from the database for identifying the fingerprint. There is other command in the BioKit coding were modified and adjusted so that the function can be utilized.

Meanwhile, in Microsoft Access, some columns in the database with empty or partially filled, the primary key does not function. A primary key does not allow Null values and must always have a unique value. A primary key is used to relate a table to foreign key in other tables. It needs a unique value to fill in the columns and hopefully the program was working well.

The circuit also has to go through a series of troubleshooting so that it will work properly. Upon checking every connection and components, some components were found to be faulty and need to be replaced. Hopefully, after replaced the faulty components, the circuit was worked and improved.

CHAPTER 5
CONCLUSION

5.1 Conclusion

In a nutshell, the project undertaken is a success as it corresponds to the initial objectives despite some problems faced along the way. It is duly noted at this stage that time proven to be a major factor in completing all the tasks and troubleshooting phases. All the related information about the biometrics and the fingerprint has been understood. The sensor that has been used was U.are.U fingerprint sensor and its software was BioKit SDK. All the registration part was done using the visual basic interface and the fingerprint images were captured using the U-are-U sensor. All the data has been collected from 5 users and stored in the database. From here, the matching part is obtained from Microsoft Access which provides the database system that record and stored all the string from the registered fingerprint. Future references are no longer an issue with the presence of the systems.

Looking from the economic perspective, this project is definitely cost effective and looking from the safety aspect, this systems is really very convenience and reliable. There are no worries about losing, stolen or forgotten to bring the matrix card. Only the authorized and registered person only can enter the building and all the time entering and leaving are being stored for futures references. The attendance for working time sheet also can be determined and stored it. The estimated cost for the sensors and software were RM17920.

In the end of this project, a new working prototype of UTP-building access control using a fingerprint identification system was obtained with storing data and information of 5 users. The system can function as a Smart UTP-Building Access Control.

5.2 Recommendations

As to end the project presentation, it is therefore important to suggest some improvement and recommendation for the benefit of the project. In any cases at all, engineers are known to improve and modify to better system.

Therefore, the recommendations suggested are as below:

- The door controller and back up database. This will required the good combination of software and the hardware for controlling the door and stored the database if the power is suddenly off.
- To use the wireless connection sensor. This will be easy for the connection of the system and need more research and study on it

REFERENCES

[1] L. C. Jain, U. Halici, I. Hayashi, 1999, "Intelligent Biometric Techniques in Fingerprint and Face recognition", CRC Press LLC, Boca Raton London New York Washington, D.C

[2] Henry C. Lee and R.E Gaensslen, 1997, "Advances in Fingerprint Technology", CDC Press LLC, Boca Raton London New York Washington D.C

[3] Nalini Ratha and Ruud Bolle, 2000, "Automatic Fingerprint Recognition Systems", Springer-Verlag, New York

[4] Ingenico, 2002, White Paper, Fingerprint Recognition System

[5] Website refers to how stuff works
http://computer.howstuffworks.com/fingerprint-scanner.htm
http://www.mybiobase.com/uru.htm

[6] Diane Zack, 2001, "Visual Basic 6.0, Enhanced Edition", Thomson Learning, Inc., Boston.
Wallace Wang, 1998, "Visual Basic 6 for Dummies, IDG Books Worldwide.
David I. Schneider, 1995, "An Introduction to Programming Using Visual Basic 6.0", Fourth Edition, Prentice Hall.
Diane Zak, 2001, "Microsoft Visual Basic for Application", Thompson Learning, Inc., Boston.
Understanding and Using Visual Basic, Jared Holyman
http://www.renton.com

[7] Virginia Anderson, 2003, "The Complete Reference Microsoft Office Access 2003", McGraw-Hill.
Evangelos Petroutsous, 2000, "Database Programming with Visual Basic 6, SYBEX Inc., USA

[8] John B. Peatman, 1998, "Design with PIC Microcontroller", Prentice Hall, Upper Saddle River, New Jersey.
Carl Bergquist, 2001, "Guide to PICMICRO Microcontroller, Prompt Publications, USA
M. James, 2001, " Microcontroller Cookbook: PIC & 8051", Great Britain, Newnes.

Maxim-IC Home Page, Max232 Datasheet
http://www.maxim-ic.com
How to Check RS-232 Port to Verify Operation
http://www.bb-europe.com/tech_articles

APPENDICES

APPENDIX A

THE CODING FOR REGISTRATION PROCESS

The registration coding

```
Option Explicit
Dim WithEvents RegSample As BioKit.Reg        'SDK Instruction

Private Sub cmdAbort_Click()
  RegSample.Abort                    'SDK Instruction
  cmdStart.Enabled = True
  cmdAbort.Enabled = False
End Sub

Private Sub cmdStart_Click()
  Dim i As Integer
  txtfingerprint.Text = ""
  For i = 0 To 3
    Image1(i).Picture = LoadPicture()
  Next i
  RegSample.Execute                  'SDK Instruction
  cmdStart.Enabled = False
  cmdAbort.Enabled = True
End Sub

Private Sub Form_Load()
Set RegSample = New BioKit.Reg              'SDK Instruction
Dim res As StatusEnum                  'SDK Instruction

RegSample.Initialize                  'SDK Instruction
RegSample.SampleWidth = Image1(0).Width        'SDK Instruction
RegSample.SampleHeight = Image1(0).Height      'SDK Instruction
res = RegSample.ActivateSensor(Me.hWnd)        'SDK Instruction
End Sub

Private Sub RegSample_RegDone(HexFingerPrint As String, Status As BioKit.StatusEnum)   'SDK
Instruction
  If Status = StatusOK Then              'SDK Instruction
    txtfingerprint.Text = HexFingerPrint     'SDK Instruction
    MsgBox "Sample successfully registered...", vbInformation
  Else
    MsgBox "Sample is not successfully registered...", vbCritical
  End If
  cmdStart.Enabled = True
  cmdAbort.Enabled = False
End Sub

Private Sub RegSample_RegLeaving()
  lblStatus.Caption = "Fingerprint Leaving"
End Sub

Private Sub RegSample_RegOnProcessError(SampleQuality As BioKit.QualityEnum) 'SDK Instruction
  lblStatus.Caption = "Status : " & SampleQuality 'SDK Instruction
End Sub

Private Sub RegSample_RegSample(SamplePicture As stdole.Picture, CurFingerIndex As Integer) 'SDK
Instruction
  Image1(CurFingerIndex).Picture = SamplePicture  'SDK Instruction
End Sub

Private Sub RegSample_RegTouching ()
  lblStatus.Caption = "Fingerprint Touching"
End Sub
```

APPENDIX B

THE CODING FOR MATCHING PROCESS

```
Option Explicit
Dim WithEvents MatchSample As BioKit.Match   'SDK Instruction

Private Sub cmdAbort_Click()
   MatchSample.Abort                    'SDK Instruction
   'eleminate biokit's instance before can create another new instance
   Set MatchSample = Nothing
   CmdMatch.Enabled = True
   cmdAbort.Enabled = False
End Sub

Private Sub CmdMatch_Click()
   'create new instance of biokit
   Set MatchSample = New BioKit.Match
   Dim res As StatusEnum
   MatchSample.Initialize                'SDK Instruction
   MatchSample.FAR = 0.005               'SDK Instruction
   MatchSample.SampleWidth = imgFingerprint.Width    'SDK Instruction
   MatchSample.SampleHeight = imgFingerprint.Height   'SDK Instruction
   res = MatchSample.ActivateSensor(Me.hWnd)
   imgFingerprint.Picture = LoadPicture()
   MatchSample.Execute (txtFingerprint.Text)     'SDK Instruction
   CmdMatch.Enabled = False
   cmdAbort.Enabled = True
End Sub

Private Sub Form_Load()
CmdMatch.Enabled = True
cmdAbort.Enabled = False
End Sub

Private Sub MatchSample_MatchDone(Status As BioKit.StatusEnum)   'SDK Instruction
   If Status = StatusOK Then                    'SDK Instruction
      MsgBox "Authentication Successful.", vbInformation
   Else
      MsgBox "Authentication fail.", vbCritical
   End If

  Call cmdAbort_Click

End Sub

Private Sub MatchSample_MatchLeaving()
   lblStatus.Caption = "Fingerprint Leaving"
End Sub

Private Sub MatchSample_MatchSample(SamplePicture As stdole.Picture, SampleQuality As
BioKit.QualityEnum) 'SDK Instruction
   imgFingerprint.Picture = SamplePicture      'SDK Instruction
   lblStatus.Caption = "Status : " & SampleQuality 'SDK Instruction
End Sub

Private Sub MatchSample_MatchTouching()
   lblStatus.Caption = "Fingerprint Touching"
End Sub

Private Sub txtFingerprint_Change()

End Sub
```

APPENDIX C

THE MODIFIED CODING FOR THE REGISTRATION PROCESS

```
Option Explicit
Dim WithEvents RegSample As BioKit.Reg        'SDK Instruction
Dim file_name As String
Dim SQLstmt As String
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim txtfingerprint As String

Private Sub cmdAbort_Click()
   RegSample.Abort                    'SDK Instruction
   'eleminate biokit's instance before can create another new instance
   Set RegSample = Nothing

   cmdStart.Enabled = True
   cmdAbort.Enabled = False
End Sub

Private Sub cmdDone_Click()
   Screen.MousePointer = 0
   Unload Me
End Sub

Private Sub cmdSearch_Click()
AdoFinger.Refresh
Do Until frmRegister.AdoFinger.Recordset.EOF
If frmRegister.AdoFinger.Recordset.Fields("StaffIDNumber").Value = txtStaffNum.Text Then
'lblStat.Caption = "Done"
MsgBox "ID is successfully entered", vbInformation
Exit Sub

Else

frmRegister.AdoFinger.Recordset.MoveNext
'lblStat.Caption = "Not found"
"MsgBox "Invalid Search Key Entered", vbCritical, "Search Error"
"clearAndFocus
End If

Loop

End Sub

Private Sub cmdStart_Click()
   Dim i As Integer
   'create new instance of biokit
   Set RegSample = New BioKit.Reg
   Dim res As StatusEnum                    'SDK Instruction

   RegSample.Initialize                    'SDK Instruction
   RegSample.SampleWidth = Image1(0).Width        'SDK Instruction
   RegSample.SampleHeight = Image1(0).Height      'SDK Instruction
   res = RegSample.ActivateSensor(Me.hWnd)        'SDK Instruction

   txtfingerprint = ""
   For i = 0 To 3
      Image1(i).Picture = LoadPicture()
   Next i
   RegSample.Execute                    'SDK Instruction
   cmdStart.Enabled = False
   cmdAbort.Enabled = True
```

```
End Sub


Private Sub DataGrid1_Click()
With rs
   txtStaffName.Text = !StaffName
   txtStaffNumb.Text = !StaffIDNumber
   txtBuilding.Text = !BuildingNo
   End With
End Sub

Private Sub Form_Load()
file_name = App.Path & "\AccessDoor.mdb"

 Set conn = New ADODB.Connection

 conn.CursorLocation = adUseClient

conn.ConnectionString = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=" & file_name & ";" & _
      "Persist Security Info=False"
conn.Open
 'Once this connection is open you can use it throughout your application
   SQLstmt = "SELECT * FROM Staff"
'  Get the records.
  Set rs = New ADODB.Recordset
  rs.Open SQLstmt, conn, adOpenStatic, adLockOptimistic, adCmdText

  Set DataGrid1.DataSource = rs
   With rs
'    .MoveFirst
   End With
cmdStart.Enabled = True
cmdAbort.Enabled = False
   txtStaffName.Text = ""
   txtStaffNumb.Text = ""
   'txtStaffNum.Text = ""
   txtBuilding.Text = ""
End Sub

Private Sub Option1_Click()

End Sub

Private Sub RegSample_RegDone(HexFingerPrint As String, Status As BioKit.StatusEnum)   'SDK Instruction
   If Status = StatusOK Then                    'SDK Instruction

   If OptRight.Value = True Then
      If AdoFinger.Recordset!RightFingerprint <> "" Then
       If MsgBox("Right finger already registered. Do you want to replace it?", vbOKCancel, "Save") = vbOK
Then
        AdoFinger.Recordset!RightFingerprint = HexFingerPrint
         AdoFinger.Recordset.Update
         MsgBox "Finger successfully registered", vbInformation
        End If
       Else
         AdoFinger.Recordset!RightFingerprint = HexFingerPrint
         AdoFinger.Recordset.Update
         MsgBox "Finger successfully registered", vbInformation
       End If
    Else
     If AdoFinger.Recordset!LeftFingerPrint <> "" Then
     If MsgBox("Left finger already registered. Do you want to replace it?", vbOKCancel, "Save") = vbOK Then
         AdoFinger.Recordset!LeftFingerPrint = HexFingerPrint
         AdoFinger.Recordset.Update
        MsgBox "Fingerprint successfully registered", vbInformation
       End If
```

```
      Else
        AdoFinger.Recordset!LeftFingerPrint = HexFingerPrint
        AdoFinger.Recordset.Update
        MsgBox "Fingerprint successfully registered", vbInformation
      End If
    End If

    Else
      MsgBox "Fingerprint is not successfully registered", vbCritical
    End If
    txtStaffName.Text = ""
    txtStaffNumb.Text = ""
    txtStaffNum.Text = ""
    txtBuilding.Text = ""

    Call cmdAbort_Click
End Sub

Private Sub RegSample_RegLeaving()
    lblStatus.Caption = "Fingerprint Leaving"
End Sub

Private Sub RegSample_RegOnProcessError(SampleQuality As BioKit.QualityEnum) 'SDK Instruction
    lblStatus.Caption = "Status : " & SampleQuality 'SDK Instruction
End Sub

Private Sub RegSample_RegSample(SamplePicture As stdole.Picture, CurFingerIndex As Integer) 'SDK
Instruction
    Image1(CurFingerIndex).Picture = SamplePicture 'SDK Instruction
End Sub

Private Sub RegSample_RegTouching()
    lblStatus.Caption = "Fingerprint Touching"
End Sub
```

APPENDIX D

THE MODIFIED CODING FOR THE MATCHING PROCESS

```
Option Explicit
Dim WithEvents MatchSample As BioKit.Match    'SDK Instruction
Dim file_name As String
Dim SQLstmt As String
Dim conn As ADODB.Connection
Dim rs As ADODB.Recordset
Dim FingerPrint As String

Private Sub cmdAbort_Click()
    MatchSample.Abort                    'SDK Instruction
   'eleminate biokit's instance before can create another new instance
   Set MatchSample = Nothing
   CmdMatch.Enabled = True
   cmdAbort.Enabled = False
End Sub

Private Sub cmdDone_Click()
   Unload Me
   frmMain.Show
End Sub

Private Sub cmdMatch_Click()
   rs.MoveFirst
   FingerPrint = rs!RightFingerprint
    'create new instance of biokit
   Set MatchSample = New BioKit.Match
   Dim res As StatusEnum

   MatchSample.Initialize                'SDK Instruction
   MatchSample.FAR = 0.005                'SDK Instruction
   MatchSample.SampleWidth = imgFingerprint.Width    'SDK Instruction
   MatchSample.SampleHeight = imgFingerprint.Height   'SDK Instruction
   res = MatchSample.ActivateSensor(Me.hWnd)

   imgFingerprint.Picture = LoadPicture()

   MatchSample.Execute (FingerPrint)          'SDK Instruction

   CmdMatch.Enabled = False
   cmdAbort.Enabled = True
End Sub

Private Sub initDB()

file_name = App.Path & "\AccessDoor.mdb"

 Set conn = New ADODB.Connection

 conn.CursorLocation = adUseClient

 conn.ConnectionString = _
      "Provider=Microsoft.Jet.OLEDB.4.0;" & _
      "Data Source=" & file_name & ";" & _
      "Persist Security Info=False"
conn.Open
'Once this connection is open you can use it throughout your application
   SQLstmt = "SELECT RightFingerPrint FROM Staff" ' you should add where to limit the selected students
'Get the records.
 Set rs = New ADODB.Recordset
  rs.Open SQLstmt, conn, adOpenStatic, adLockOptimistic, adCmdText
```

```
'Set DataGrid1.DataSource = rs

End Sub

Private Sub Form_Load()

   Call initDB

CmdMatch.Enabled = True
cmdAbort.Enabled = False
'open port
MSComm1.CommPort = 1
MSComm1.Settings = "9600,n,8,1"
MSComm1.PortOpen = True
End Sub

Private Sub MatchSample_MatchDone(Status As BioKit.StatusEnum)   'SDK Instruction
    Dim found As Boolean
       found = False

   If Status = StatusOK Then                                    'SDK Instruction
      MsgBox "Authentication Successful.", vbInformation
      found = True
   Else

       If Not rs.EOF Then
       Do While (Not rs.EOF And Not found)

        If MatchSample.Compare(rs!RightFingerprint) Then
          found = True
        End If

        rs.MoveNext
      Loop
      End If
      If found Then
       MsgBox "Authentication Successful.", vbInformation
       'MSComm1.Output = Chr(53)
      Else
      MsgBox "Authentication fail.", vbCritical
      'MSComm1.Output = Chr(54)
      End If

    rs.MoveFirst

    cmdAbort_Click
    cmdMatch_Click
   End If

  Call cmdAbort_Click

End Sub

Private Sub MatchSample_MatchLeaving()
   lblStatus.Caption = "Fingerprint Leaving"
End Sub

Private Sub MatchSample_MatchSample(SamplePicture As stdole.Picture, SampleQuality As
BioKit.QualityEnum) 'SDK Instruction
   imgFingerprint.Picture = SamplePicture        'SDK Instruction
   lblStatus.Caption = "Status : " & SampleQuality 'SDK Instruction
End Sub

Private Sub MatchSample_MatchTouching()

  lblStatus.Caption = "Fingerprint Touching"
End Sub
```

APPENDIX E

PIC16F84A PROGRAMMING

```
Dim i As Byte  'declare a variable

TRISB = %00000100  'set PORTB Bit 2 As input(receive pin)
PORTB.5 = 1
PORTB.3 = 0
PORTB.4 = 0
loop:
If i = 53 Then  'door open
PORTB.4 = 1          'set bit 5 high
Endif
If i = 53 Then
PORTB.5 = 1
Endif
If i = 53 Then
PORTB.3 = 0
Endif
If i = 54 Then  'door close
PORTB.5 = 0          'set bit 5 low
Endif
If i = 54 Then
PORTB.4 = 0
Endif
If i = 54 Then
PORTB.3 = 1
Endif

Serin PORTB.2, 9600, i  'set baud rate as serial port
Goto loop  'loop forever
```

APPENDIX F

THE BIOKIT SDK

Bio Base

BioKit
Software Development Kit

BioKit Software Development Kit (SDK) enables developers to add the power of fingerprint authentication security into their Windows applications. For developers who are using Visual Basic to develop their programs, it will be a convenient and easy way to interact with our COM DLL SDK since it provide the sample codes along with the complete documentation. The toolkit also includes header files that define the Data types for Visual C++.

Who will be using it?

Application Developers – To those who are developing an application that needed to be protected using a fingerprints security mechanism. For VB developers, integrating the application with the Biometrics technology would be much easier.

Sensor Device

The device component provides both physical device and plug-and-play management functions. It supports security features such as the use of challenge-response protocol to avoid playback attack when communicating with the terminal. Secure fingerprint images which obtain from the sensor are protected by proprietary encryption technology.

SDK Components

Registration

The registration component extracts the fingerprint raw data from the sensor device(s) before generates a register fingerprint template. It will obtain a several fingerprints capturing and adapting to the quality of the fingerprint that is being imaged. This process consists of extracting features from each print and combining them into one robust template. Finally, it returns a hexadecimal string which consists of the fingerprint details.

Matching

The matching component compares a registered template that generate from the registration component with the one provided by the user via the sensor device(s). If the matching of the fingerprint is authenticated, a success status will be returned.

Hardware Information

Basically, this component does nothing more than retrieving the related hardware information from the sensor device(s). Information can be accessed directly after the class being initialized as an object.

APPENDIX G
SDK DESCRIPTIONS

Contents

The *BioKit* SDK (Software Development Kit) programmer's Guide shows programmers or users how to use the SDK instruction to integrate the functionality into their applications.

1.1 Requisite Knowledge

Programmers or users are assumed to be familiar with the following subjects before using this SDK kit:-

- Having experience in COM (Component Object Model) or similar knowledge of COM-based technologies, such as distributed COM or COM+.
- Proficiency in any Object Oriented Language like Microsoft Visual Basic, Microsoft Visual C++ or Borland C++ Builder.

1.2 Comments and Feedback Welcome

Your comments and feedback are important to us!
The information in this guide has been tested and reviewed thoroughly. But if users find an error during testing or having others suggestion on future SDK releases, please send us your point of view to one of the following ways :-

- Mail to us at:
Bio Base Solutions (M) Sdn Bhd. (527157-V)
No. 7-3, Jalan SP 2/3,
Taman Serdang Perdana
Seksyen 2, Seri Kembangan,
44300 Selangor Darul Ehsan,
Malaysia.

Tel : (+603)-8941 8968
URL : http://www.mybiobase.com

- Fax to us at: (+603)-89418921
- Email to us at: *sales@mybiobase.com*

2.1 To install the *BioKit* SDK

Follow the below step by step instructions and begin the installation of *BioKit* SDK.-

1. Insert the *BioKit* SDK CD in the CD-ROM drive and double-click the file, 'setup.exe'through Windows Explorer or My Computer.
2. When coming to the *BioKit* SDK's welcome setup dialog box, click next to start the installation.
3. Reviewing the license agreement and proceed to next if user agrees to the terms and condition of the development kit.
4. Click next again to install *BioKit* SDK into the default destination folder, 'c:\Program Files\BioKit'. If user wants to install the SDK into another location, click Browse and specify a different location before proceed further.
5. Just as the installation is completed, click Finish and then restart the computer when prompt.

The installer program will create the *BioKit* SDK folder according to the location that specified by user in step 4. Inside the whole SDK, sample applications are included for showing how to use the *BioKit* SDK are provided in both Visual Basic and C++.

2.2 *BioKit* SDK Features

- Works on any programming languages that can interface with COM objects such as Visual Basic, C++ and etc.
- Contains the sample applications that come together with the SDK for user reviewing.
- Secure the fingerprint with Standard Encryption Algorithm.
- Return the users fingerprint in Bitmap format.

2.3 *BioKit* System Requirements

- U are U System Integrator Platinum Edition must be installed.
- Requirement of RAM can be varied, depends on the users application. Basically, 32 MB minimum physical RAM (64 MB physical RAM recommended).
- Support any Pentium-class processor.
- Can be implement under the platform of Windows 98, ME, NT4 with service pack 4 or later, 2000, XP or later operating systems.
- CD-ROM Drive.
- USB Ports.

In this particular chapter, user will be introduced with the fingerprint recognition operation and explains how to use all the SDK instructions that will be implemented in user software application. With this SDK, user however, is shielded from the complexity of integrating with biometrics devices. User only needs to decide which process to perform: registration or verification. Next, user needs to write event handlers that generated by these two processes to control them or provide a user feedback. As a result, writing applications with *BioKit* SDK is much easier, simpler and faster.
During this section, we will show user how to initiate the registration and verification processes using the *BioKit* SDK instructions together with the sample code (current version written in Visual Basic) as an examples.

3.1 Registering a Fingerprint Template

With *BioKit* SDK, a fingerprint template registration process can be done like the following scenario:-

i. Create an instance for the *BioKit.Reg* object.
ii. Next, connect the object to its event interface.
iii. Finally, call the Execute method of *BioKit.Reg* object to start the registration the process.

The following sample code shows the particular process:-

```
Dim WithEvents RegSample As BioKit.Reg

.....
.....
.....

Set RegSample = New BioKit.Reg

RegSample.Initialize
res = RegSample.ActivateSensor(Me.hWnd)

RegSample.Execute
```

Figure 1.1 Invoke the Registration Process

In figure 1.1, an instance of *BioKit.Reg* is initiated and connected to its event interface. As soon as the *Execute* method is called, the registration process is started.

3.2 Handling Events from the Registration Process

User cannot control the entire registration process within the *BioKit* SDK – such as raw sample-to-sample conversion, etc. – User can only handle the events that generated by

object. Yet, this will provide a space where user application can receive a various form of feedback during the registration process.

This section provides event handler code samples for the following events:

- *RegSample*, triggered when a sample is acquired from the sensor device(s).
- *RegOnProcessError*, triggered in order to return the sample quality status.
- *RegDone* event of the *BioKit.Reg* component, triggered when a registration is completed.

3.3 Displaying the Sample Image

When a sample is acquired by the sensor, the *SampleReady* event is triggered.
The following sample code displays the image of the acquired sample:-

```
Private Sub RegSample_RegSample(SamplePicture As stdole.Picture, _
                                CurFingerIndex As Integer)
  Image1 (CurFingerIndex).Picture = SamplePicture
End Sub
```

Figure 1.2 *RegSample* Event Handler

The sample code in figure 1.2 receives the sample acquired by the sensor as a bitmap picture format. *CurFingerIndex* indicate the number of times for that particular finger being acquired during registration process.

3.4 Returning the Sample Quality

With *RegOnProcessError* event, user application able to receive a feedback returning from the event regarding to the sample quality status code.
The below sample code is a handler for the *RegOnProcessError* event and displays the quality of the sample that defined by the *BioKit.QualityEnum* enumerator object:-

```
Private Sub RegSample_RegOnProcessError(SampleQuality As BioKit.QualityEnum)
  lblStatus.Caption = "Status : " & SampleQuality
End Sub
```

Figure 1.3 *RegOnProcessError* Event Handler

In figure 1.3, the event handler receives the quality status of sample thru the *SampleQuality* parameter. This event handler will be triggered every times whenever the sensor device(s) acquired the raw sample.

Note: Please review the SDK reference for more details about the SampleQuality status code.

3.5 Finalizing the Registration process

As soon as the registration is completed, the *RegDone* event of the *BioKit.Reg* component is triggered and the best registration template will be passed as a parameter to the event handler:-

```
Private Sub RegSample_RegDone(HexFingerPrint As String, _
                              Status As BioKit.StatusEnum)
  If Status = StatusOK Then
    txtfingerprint.Text = HexFingerPrint
    MsgBox "Sample successfully registered...", vbInformation
  Else
    MsgBox "Sample is not successfully registered...", vbCritical
  End If
... ... ... ... ... ... ... ... ... ... ... ...
End Sub
```

Figure 1.4 *RegDone* Event Handler

In figure 1.4, the *RegDone* event handler will return a Hexadecimal string which consists of the fingerprint details. User can then save the particular hex string into a file or database.

Note: Please review the SDK reference for more details about the status code.

3.6 Verifying a Fingerprint Template

Just like the registration process, user can initiate the verification process thru the *BioKit.Match* object. It's assumed that user have acquired the registration template from the database or a file and passed in as a parameter into the *Execute* method:

```
Dim WithEvents MatchSample As BioKit.Match

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...
... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

Set MatchSample = New BioKit.Match

MatchSample.Initialize
MatchSample.SampleWidth = imgFingerprint.Width
MatchSample.SampleHeight = imgFingerprint.Height
res = MatchSample.ActivateSensor(Me.hWnd)

MatchSample.Execute (txtFingerprint.Text)
```

Figure 1.5 Initiating the verification process

In the sample code, an instance of *BioKit.Match* is created and connected to its event interface. Then, the *Execute* method is called together with a passing parameter consists of the registration template into it. And eventually, the verification process will be started.

3.7 Handling Events from the Verification Process

Similar to the registration process, events handler generated by *BioKit.Match* is to provide a various forms of feedback to the user.

The event handler *MatchSample*, triggered when a raw sample is acquired from the sensor device(s) together with the sample quality status.

```
Private Sub MatchSample_MatchSample(SamplePicture As stdole.Picture, _
                                    SampleQuality As BioKit.QualityEnum)
  imgFingerprint.Picture = SamplePicture
  lblStatus.Caption = "Status : " & SampleQuality
End Sub
```

Note: Please review the SDK reference for more details about the SampleQuality status code.

3.8 Finalizing the Verification process

When verification is completed, the *MatchDone* event of the *BioKit.Match* object will be triggered eventually. The following sample code shows the returning status when the verification process is completed:-

```
Private Sub MatchSample_MatchDone(Status As BioKit.StatusEnum)
  If Status = StatusOK Then
    MsgBox "Authentication Successful.", vbInformation
  Else
    MsgBox "Authentication fail.", vbCritical
  End If
... ... ... ... ... ... ... ... ... ... ... ... ...
End Sub
```

Figure 1.6 Sample Event Handler for the *MatchDone*

In figure 1.6, the event handler return with a status parameter which generated just after the verification process is completed.

Note: Please refer to the SDK reference for more details about the returning status code.

In this chapter, we will define the structures of *BioKit* SDK (Software Development Kit) associated with functions, events, properties and as well as error codes for the user or programmer reference.

A COM (Component Object Model) component that contains a classes of functions which perform the device(s) initialization, fingerprint registration and matching. It allows in-house software developer to implement a biometrically-secure application easily. This SDK consists of three different classes that having the functionality like the following:-

i. *BioKit.Info*
Allow to perform the hardware initialization as well as abstract the hardware information from the sensor device(s).

ii. *BioKit.Reg*
Enable the fingerprint registration by capturing the fingerprint from the sensor and return a fingerprint data in hexadecimal format.

iii. *BioKit.Match*
Allow to match the fingerprint data generated by *BioKit.Reg* component with the fingerprint capturing from the sensor device(s).

4.1 Structures for *BioKit.Info*

This section will define the description of all the functions, events, properties and error codes which reside within the *BioKit.Info* class function.

4.1.1 Method(s)

Initialize() As StatusEnum

- Initialize the sensor device(s) that connected to the USB ports and return the status codes of device(s)

4.1.2 Properties

VendorName() As String

- Return the string description of the device vendor (read-only).

Index() As Integer

- Return or Set the integer value which the current sensor device(s) connected to the USB ports.

NumOfDevices() As Integer

- Return the number of sensor device(s) that currently connected to the USB ports.

ProductName() As String
- Return the product name which depends on the type of sensor connected to the PC (read-only).

HWRevision() As String
- Return the firmware revision number of the device (read-only).

SerialNo() As String
- Return the serial number for the device (read-only).

Identifier() As String
- Return the device identifier, which varies when other USB devices are plugged into the PC (read-only).

sType() As String
- Return code type for the device (read-only).

ImageWidth() As Integer
- Returns the width (in pixels) of the fingerprint image acquired by the sensor (read-only).

ImageHeight() As Integer
- Return the height (in-pixels) of the fingerprint image acquired by the sensor (read-only).

Xdpi() As Integer
- Return the resolution (dots per inch) on the X axis of the fingerprint image (read-only).

Ydpi() As Integer
- Returns the resolution (dots per inch) on the Y axis of the fingerprint image (read-only).

BitsPerPixel() As Integer
- Returns the number of bits per pixel in the fingerprint image (read-only).

4.1.3 Event Handlers
None

4.1.4 Data Types
StatusEnum
StatusOK
StatusSystem
StatusDeviceBroken
StatusAlreadyCreated

StatusRegisterFail
StatusInvalidArg
StatusBadSignature
StatusUnknown

4.2 Structures for *BioKit.Reg*

This section will define the description of all the functions, events, properties and error codes which reside within the *BioKit.Reg* class function.

4.2.1 Method(s)

Initialize() As StatusEnum

- Initialize the sensor device(s) that connected to the USB ports and return the status codes of device(s).

ActivateSensor(ByVal UserhWnd As Long) As StatusEnum

- Activate the sensor device(s) that currently attach to the USB port thru the *index* properties.

Execute()

- Start the registration process after the *Initialize* and *ActivateSensor* methods are being called.

Abort()

- Abort the whole registration process immediately.

4.2.2 Properties

SampleWidth(ByVal iData As Long)

- Return or set the width of the fingerprint image as it was acquired from the sensor (in pixels) (read-only).

SampleHeight(ByVal iData As Long)

- Return or set the height of the fingerprint image as it was acquired from the sensor (in pixels) (read-only).

Index(ByVal iData as Integer)

- Return or set the integer value which the current sensor device(s) connected to the USB ports.

4.2.3 Event Handlers

RegOnProcessError(SampleQuality As QualityEnum)

- Return a sample quality status codes which is acquired during the registration process.

RegSample(SamplePicture As IPictureDisp, CurFingerIndex As Integer)
- Return a sample with bitmap picture format and the number of times for the particular finger which is being acquired during registration process.

RegDone(HexFingerPrint As String, Status As StatusEnum)
- Return a hexdecimal string format as well as status codes when the registration operation is completed.

RegTouching()
- This event handler will be raised whenever a user finger approaching the sensor.

RegLeaving()
- This event handler will be raised whenever user finger leave the sensor.

4.2.4 Data Types

StatusEnum
StatusOK
StatusSystem
StatusDeviceBroken
StatusAlreadyCreated
StatusRegisterFail
StatusInvalidArg
StatusBadSignature
StatusUnknown

QualityEnum
SQGood
SQNone
SQTooLight
SQTooDark
SQTooNoisy
SQLowContrast
SQNotEnghFtr
SQNoCentralRegion
SQUnknown

PriorityEnum
PRStandard
PRHigh
PRLow

4.3 Structures for *BioKit.Match*

This section will define the description of all the functions, events, properties and error codes which reside within the *BioKit.Match* class function.

4.3.1 Method(s)

Initialize() As StatusEnum

- Initialize the sensor device(s) that connected to the USB ports and return the status codes of device(s).

ActivateSensor(ByVal UserhWnd As Long) As StatusEnum

- Activate the sensor device(s) that currently attach to the USB port thru the *index* properties.

Execute(HexFingerPrint1 As String)

- Start the verification process by passing in the hexadecimal fingerprint template as a parameter after the *Initialize* and *ActivateSensor* methods are being called.

Compare(HexFingerPrint1 As String) As Boolean

- Basically, perform the same function like *Execute* method. The only differences are this method can be worked independently without raising a *MatchDone* event and return a Boolean (true or false) value. It can be only executed after the *Initialize* and *ActivateSensor* methods are being called.

Abort()

- Abort the whole verification process immediately.

4.3.2 Properties

SampleWidth(ByVal iData As Long)

- Return or set the width of the fingerprint image as it was acquired from the sensor (in pixels) (read-only).

SampleHeight(ByVal iData As Long)

- Return or set the height of the fingerprint image as it was acquired from the sensor (in pixels) (read-only).

Index(ByVal iData As Integer)

- Return or Set the integer value which the current sensor device(s) connected to the USB ports.

FAR(ByVal SngData As Single)

- Return or Set the FAR (False Acceptance Rate) which is a probability that an unauthorized personnel attempt to be accepted. The value is valid within the range from 1.0 to 0.000001. The smaller the value, the higher the security level. Default value is 0.005.

4.3.3 Event Handlers

Event MatchSample(SamplePicture As IPictureDisp, SampleQuality As QualityEnum)

- Return a sample with bitmap picture format and the sample quality status codes which is acquired during verification process.

Event MatchDone(Status As StatusEnum)

- Return a status codes when the matching operation is completed. If the return status codes is 'StatusOK' then the matching is successfully verify. Or else others error status codes will be returned.

Event MatchTouching()

- This event handler will be raised whenever a user finger approaching the sensor.

Event MatchLeaving()

- This event handler will be raised whenever user finger leave the sensor.

4.3.4 Data Types

StatusEnum

StatusOK
StatusSystem
StatusDeviceBroken
StatusAlreadyCreated
StatusRegisterFail
StatusInvalidArg
StatusBadSignature
StatusUnknown

QualityEnum

SQGood
SQNone
SQTooLight
SQTooDark
SQTooNoisy
SQLowContrast
SQNotEnghFtr
SQNoCentralRegion
SQUnknown

PriorityEnum

PRStandard
PRHigh
PRLow

4.4 Data Types Description

4.4.1 StatusEnum Data Types

StatusOK	0	When the operation is successfully performed.
StatusSystem	1	When the system occurs an errors during processing the sample.
StatusDeviceBroken	2	When the sensor device(s) is in malfunction state.
StatusAlreadyCreated	3	When the object is not empty but already contains data.
StatusRegisterFail	4	When the registration template cannot be created.
StatusInvalidArg	5	When input or output parameter is invalid.
StatusBadSignature	6	When the signature is not verified.
StatusUnknown	7	When unknown error is detected by the SDK.

4.4.2 QualityEnum Data Types

SQGood	0	The sample is in good condition.
SQNone	1	The sample does not have quality information.
SQTooLight	2	The sample is too light.
SQTooDark	3	The sample is too dark.
SQTooNoisy	4	The sample is too noisy.
SQLowContrast	5	The sample is in low contrast.
SQNotEnghFtr	6	There are not enough features on your finger.
SQNoCentralRegion	7	The sample is not located on the centre of the sensor.
SQUnknown	8	Unknown error is returned.

4.4.2 PriorityEnum Data Types

PRStandard	0	The calling application will response when active.
PRHigh	1	The calling application will response even being minimize.
PRLow	2	If two calling applications active together, it will allow the other window handler with higher priority to be executed first.