

ROCKY4ND

Developer's Guide

Version 1.1

FEITIAN Technologies Co., Ltd.

Copyright © 1999-2005

FEITIAN Technologies Co., Ltd.

Software Developer's Agreement

All Products of FEITIAN Technologies Co., Ltd. (FEITIAN) including, but not limited to, evaluation copies, diskettes, CD-ROMs, hardware and documentation, and all future orders, are subject to the terms of this Agreement. If you do not agree with the terms herein, please return the evaluation package to us, postage and insurance prepaid, within seven days of their receipt, and we will reimburse you the cost of the Product, less freight and reasonable handling charges.

1 **Allowable Use** – You may merge and link the Software with other programs for the sole purpose of protecting those programs in accordance with the usage described in the Developer's Guide. You may make archival copies of the Software.

2 **Prohibited Use** – The Software or hardware dongle or any other part of the Product may not be copied, reengineered, disassembled, decompiled, revised, enhanced or otherwise modified, except as specifically allowed in item 1. You may not reverse engineer the Software or any part of the product or attempt to discover the Software's source code. You may not use the magnetic or optical media included with the Product for the purposes of transferring or storing data that was not either an original part of the Product, or a FEITIAN provided enhancement or upgrade to the Product. You may not Transmit Any part of THE SOFTWARE in your public servers.

3 **Warranty** – FEITIAN warrants that the dongles and Software storage media are substantially free from significant defects of workmanship or materials for a time period of twelve (12) months from the date of delivery of the Product to you.

4 **Breach of Warranty** – In the event of breach of this warranty, FEITIAN's sole obligation is to replace or repair, at the discretion of FEITIAN, any Product free of charge. Any replaced Product becomes the property of FEITIAN.

Warranty claims must be made in writing to FEITIAN during the warranty period and within fourteen (14) days after the observation of the defect. All warranty claims must be accompanied by evidence of the defect that is deemed satisfactory by FEITIAN. Any Products that you return to FEITIAN, or a FEITIAN authorized distributor, must be sent with freight and insurance prepaid.

EXCEPT AS STATED ABOVE, THERE IS NO OTHER WARRANTY OR REPRESENTATION OF THE PRODUCT, EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

1 **Limitation of FEITIAN's Liability** – FEITIAN's entire liability to you or any other party for any cause whatsoever, whether in contract or in tort, including negligence, shall not exceed the price you paid for the unit of the Product that caused the damages or are the subject of, or indirectly related to the cause of action. In no event shall FEITIAN be liable for any damages caused by your failure to meet your obligations, nor for any loss of data, profit or savings, or any other consequential and incidental damages, even if FEITIAN has been advised of the possibility of damages, or for any claim by you based on any third-party claim.

2 **Termination** – This Agreement shall terminate if you fail to comply with the terms herein. Items 2, 3, 4 and 5 shall survive any termination of this Agreement.

Contact Information

World Wide Web:

www.FTsafes.com

FEITIAN Technologies Co., Ltd.

Tel: +86-10-62304466 Fax: +86-10-62304477 Email: world.sales@Ftsafe.com Add: Bldg 7A, 5th Floor, Xueyuan Road 40, Haidian District, Beijing 100083, P.R China

Please Email any comments, suggestions or questions regarding this document to us at:
world.sales@Ftsafe.com

Version	Date
1.0	2005.9
1.1	2005.10

CE Attestation of Conformity



ROCKEY is in conformity with the protection requirements of CE Directives 89/336/EEC Amending Directive 92/31/EEC. ROCKEY satisfies the limits and verifying methods: EN55022/CISPR 22 Class B, EN55024: 1998.

FCC Standard



This device is in conformance with Part 15 of the FCC Rules and Regulation for Information Technology Equipment.

Operation of this product is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.



USB

This equipment is USB based.

Quick Start

- 1 All ROCKEY dongles mentioned in this document are ROCKEY4ND dongles.
- 2 We provide a Developer's Kit (DK) to software developers for their evaluation. The DK contains everything you need to evaluate the software protection system: a Developer's Guide, a CD-ROM and a ROCKEY4ND dongle(s). The ROCKEY4ND dongles are exactly the same as the commercial version of the product in all respects, save one. The only difference is that the passwords burned into the demonstration dongles are publicly known (P1:C44C, P2:C8F8, P3:0799, P4:C43B). After evaluation if you decide to use the product, you can purchase dongles with secret and unique passwords, so others can not read, edit or modify the content in your dongles.

Run *Setup.exe* under the root directory of the CD-ROM, you may finish your required installation with this installation wizard.

ROCKEY4ND is a driverless dongle that requires no device driver. The operating system has the driver for ROCKEY4ND embedded inside the system. We provide the driver files in the directory *Driver* for the operating systems that may not have these files.

- 3 To call the ROCKEY4ND dongle, you are not required to install a driver. It is only required to copy the library (Rockey4ND.dll and specific lib for other programming languages) to the same path of your applications.

You may find the ROCKEY tools, such as the ROCKEY Editor and Envelope Encryption program, in the directory *Tools* on the CD-ROM.

RyEnv32_ND.exe is the tool for Envelope Encryption. You do not need to write any code, just choose the Win32 PE files, such as EXE、DLL、ARX files with your mouse, and then you may encrypt them with this program. (Refer to Chapter 6 -- ROCKEY4ND Envelope Encryption)

- 4 Rocky4ND_Editor.exe was designed to help you edit, modify, test and write the contents into the dongle in batch. It is the Utility for you to develop encryption programs.
- 5 Rocky4RU.exe is the tool for updating ROCKEY4ND remotely. Software developers may use this tool to perform a remote update for ROCKEY4ND and change the contents of the unit. We provide several remote update schemes to meet various requirements of your software release.
- 6 DataRecorder.exe provides a powerful dongle management solution. With the assistance of DataRecorder.exe, the developer can record the data inside the dongle, the owner of the dongle, the hardware info, the passwords and other information regarding the unit. A database that records this information will assist developers to control their software release with convenience.
- 7 ROCKEY4ND API is the ideal tool for you to make full use of the security functions of ROCKEY4ND in your applications. You will learn how to use ROCKEY4ND quickly with the reference of provided program samples.

Examples for different programming languages can be found in the directory *Samples*. The folder *Beginner* in *Samples* provides step-by-step examples to assist users in understanding the functions of ROCKEY4ND.

8 Please visit our website: <http://www.FTsafe.com>. This site is updated frequently with all relevant information for the Feitian family of products.

Content

Chapter 1 Brief Introduction	7
1.1 About ROCKEY4ND	7
1.2 ROCKEY4ND Advantages.....	8
1.3 How to Choose the Right Software Protection Solution	8
Chapter 2 ROCKEY4ND Hardware Features	9
2.1 ROCKEY4ND Internal Structure.....	9
2.2 ROCKEY4ND Hardware Interface.....	9
Chapter 3 Installing ROCKEY4ND SDK	10
3.1 Content of the CD-ROM.....	10
3.2 Installing the SDK	10
3.3 Uninstalling the Development Package	13
Chapter 4 Concepts for Developers	14
4.1 Passwords.....	14
4.2 Customer Code	14
4.3 Hardware ID	14
4.4 User Data Zone	14
4.5 Module Zone	14
4.6 Module Attributes	15
4.7 User Algorithm Zone.....	15
4.8 User ID	15
4.9 Random Number	15
4.10 Seed and Return Values	15
Chapter 5 ROCKEY4ND Editor	16
5.1 Brief Introduction.....	16
5.2 Operation	19
5.3 Template Storage	27
5.4 Log File and File Setting.....	27
5.5 Editor Update	28
Chapter 6 ROCKEY4ND Envelope Encryption	29
6.1 Single File Encryption.....	30
6.2 Multi File Encryption.....	33
Chapter 7 ROCKEY4ND API.....	35
7.1 ROCKEY4ND Function Prototype and Definition	35
7.2 ROCKEY4ND API Services	37
7.3 Return Codes	43
7.4 Basic Application Examples	44
7.5 Advanced Application Examples.....	64
Chapter 8 ROCKEY4ND Hardware Algorithms	95
8.1 ROCKEY User Defined Algorithm Introduction	95
8.2 Writing User Defined Algorithms into ROCKEY	99
8.3 User Defined Algorithm Examples.....	100
8.4 Note.....	129
8.5 Tips	129
ROCKEY4ND Technical Specification	131

Chapter 1 Brief Introduction

1.1 About ROCKEY4ND

ROCKEY4ND is an advanced software protection system that attaches to the USB port of a computer. Your software may be duplicated, but it will only run when your ROCKEY4ND “dongle” is attached to the computer. It can also limit the use of your software. Your application will interact with ROCKEY4ND at start-up and during runtime. If the dongle has been removed, or if an application module has been accessed a preset number of times, it can issue an error message and terminate, or take other alternative actions to ensure compliance with your licensing agreement. ROCKEY4ND is versatile and can be applied to other scenarios as required.

Unlike some competing products, ROCKEY4ND is a powerful miniature computer, with a CPU, memory and specialized firmware built-in that allows for a robust interaction with your application. You may write complex algorithms that are securely stored in the dongle, and then call those algorithms from time-to-time in your application. This method for software protection is strongly recommended and is very difficult to crack, and although ROCKEY4ND was designed to implement extremely high levels of security - it is also relatively easy to implement. The ROCKEY4ND API set has been simplified and improved based on experience gained from earlier versions.

The ROCKEY4ND product also provides an Envelope program. The ROCKEY4ND Envelope (*RyEnv32_ND.exe*) will encrypt Windows Portable Executable files (such as .dll, .exe and .arx) and is simple enough to implement in only a few seconds. The ROCKEY4ND Envelope is an ideal solution if you do not possess the source code for your application, or are unfamiliar with implementing an API. A security system that combines both the API set and the Envelope program will offer the greatest level of protection.

There are several components to the ROCKEY4ND software security solution and each of them will be discussed in this document. The following is an overview of the ROCKEY4ND components, along with a reference to where they will be discussed in this document:

The ROCKEY4ND Envelope program (*RyEnv32_ND.exe*) is a fast and convenient means of encrypting .exe, .dll, .arx and other Portable Executable (PE) files. This solution is ideal if you do not have access to source code or you are not familiar with the ROCKEY4ND API set. (See Chapter 6: ROCKEY4ND Envelope Encryption)

The ROCKEY4ND Editor (*Rockey4ND_Editor.exe*) is a graphical tool for performing operations on the dongle. The Editor may be used to read data from and write data to the dongle, perform arithmetic operations in the dongle or test the dongle for malfunctions. (See Chapter 5: ROCKEY4ND Editor)

ROCKEY4ND has an API set that you may use to create flexible and powerful software protection systems. This document provides VC ++ examples and other examples are provided on the CD-ROM under the *Samples\Beginner* directory. (See Chapter 7: ROCKEY4ND API)

Software Protection Mechanism of ROCKEY4ND: The protected software application must call the ROCKEY4ND dongle during run time, since the application is dependant on the hardware. It is impossible to duplicate the chipset of the ROCKEY4ND hardware, and so too it is impossible to duplicate your software, ensuring your software is protected from piracy.

1.2 ROCKEY4ND Advantages

- 1 **Compact Design** -- FEITIAN's USB dongles are among the smallest in the world. The USB dongle is 50mm x 17mm x 7mm.
- 2 **High Speed** -- ROCKEY4ND was designed to process even very complex algorithms with minimal delay for your application. Users will typically notice no degradation in application performance as a result of ROCKEY4ND being implemented.
- 3 **Ease of Use** -- ROCKEY4ND's reduced API set simplifies the programming effort in implementing API calls within your code, and the Envelope program has also been improved for increased security with the release of ROCKEY4ND. Developers lead time in implementing ROCKEY4ND is vastly reduced, saving both time and costs in deploying security into your software.
- 4 **High Security Levels** -- Redesigned ROCKEY4ND offers a much higher level of security over previous version. ROCKEY4ND implements a two level security system to segregate users who require read only access from those who require administrative privileges. ROCKEY4ND has a built in time gate to prevent software tracking and is powerful enough to support developer defined algorithms that brings software protection to a new level of security.
- 5 **High Reliability** -- FEITIAN employs an advanced customers managing system for ROCKEY4ND. We guarantee that the password of every customer is unique and that the hardware ID of every dongle is also unique. The password and hardware ID are burnt into the CPU, it is absolutely impossible to change, even for us—the manufacturer.
- 6 **Broad Support for Operating Systems** -- ROCKEY4ND protected applications may run on: Windows 98SE/ME/2000 /XP/2003; Linux; MAC.
- 7 **Abundant Programming Language Interfaces** -- ROCKEY4ND provides interfaces for these common development tools: PB, DELPHI, VFP, VB, VC, C++ BUILDER and etc.

1.3 How to Choose the Right Software Protection Solution

The protection level applied to software not only depends on the dongle, but also on how the developer uses the dongle. Even if the dongle is the best in the world, a rudimentary implementation of security with your dongle can render the total security solution weak. ROCKEY4ND dongles offer two protection methods: envelope encryption and API encryption.

You may invoke the program *RyEnv32_ND.exe* under the directory *Tools\Envelop* to perform the envelope encryption function. As the name indicates, envelope encryption adds an envelope to the user's designated files to protect them. The envelope will call the dongle. When users execute the program protected by the envelope, the protected program will automatically call the ROCKEY4ND and decide whether to allow the program to continue according to the results of the call. The envelope program directly encrypts the compiled files. The advantage of envelope encryption is that it is very easy and quick to implement and the source code does not need to be modified. The envelope method is the ideal choice if there is no time for learning the API method or if the source code is lost or unavailable. The disadvantage is that an envelope program uses a rule based encryption method, and rule based encryption methods are not as strong as methods that use an encryption key. Also, envelope encryption cannot support script languages that cannot be compiled, such as VBA.

For API encryption developers need to choose the appropriate language interface according to their programming language to access the dongle. API encryption was designed to be flexible; so you can make full use of the encryption functions of ROCKEY4ND. Developers can decide where and how to encrypt their software. API encryption is more secure than envelope encryption and especially so when the internal algorithm function of ROCKEY4ND is utilized. But API encryption must work with the original program and it can take the developer more time to become familiar with the API.

Chapter 2 ROCKEY4ND Hardware Features

2.1 ROCKEY4ND Internal Structure

At the core of ROCKEY4ND is a specialized CPU with a USB interface. It supports the USB 1.0 standard and is compatible with USB 2.0 standard. In addition to the CPU is a non-volatile memory chip that can save your data in the event of a power loss. The ROCKEY4ND functions are divided into User, Module and Algorithm zones. The developer may store important information (such as an application serial number) inside the dongle. You can write to the ROCKEY4ND dongle as many as 100,000 times – there is no appreciable limit on the numbers of reads.

The ROCKEY4ND chip supports special functions for random number generation, seed code generation and user defined algorithm interpretation.

2.2 ROCKEY4ND Hardware Interface

ROCKEY4ND USB supports USB Standard 1.0. At the most 16 USB dongles can attach to a computer with a USB extension HUB. The LED of ROCKEY4ND USB indicates the status of the dongle. (In a normal state after the dongle is attached to the computer the LED will be on all the time. If the LED blinks it indicates that the driver is not installed. Other LED responses indicate hardware failure.)

Note: ROCKEY4ND is a plug and play USB device. To unplug a ROCKEY4ND while writing/reading, the dongle may cause crashes to the operating system in some instances.

Chapter 3 Installing ROCKEY4ND SDK

You will find the program *Setup.exe* under the root directory of the CD-ROM included in the Software Developer's Kit (SDK). The contents of the CD-ROM are not zipped. Experienced developers may simply copy all necessary content to the computer.

3.1 Content of the CD-ROM

The content of the CD-ROM consists of two parts:

Tools under the directory *Tools*. Some documents on how to use these tools are provided in the corresponding folder.

APIs for different programming languages

3.2 Installing the SDK

Below we will discuss how to install and use the development package.

Step 1.

FEITIAN provides a *Setup.exe* installation wizard program on the CD-ROM. You may select the components you need. The installation of the drivers is also integrated in this wizard. Double click the *setup.exe* file from the root directory of the ROCKEY4ND CD-ROM. You will see the first screen of the Setup Wizard pictured below (Figure 3.1). Select language in the first step.



Figure 3.1

Step 2.

Close other application to avoid the need of rebooting the system.

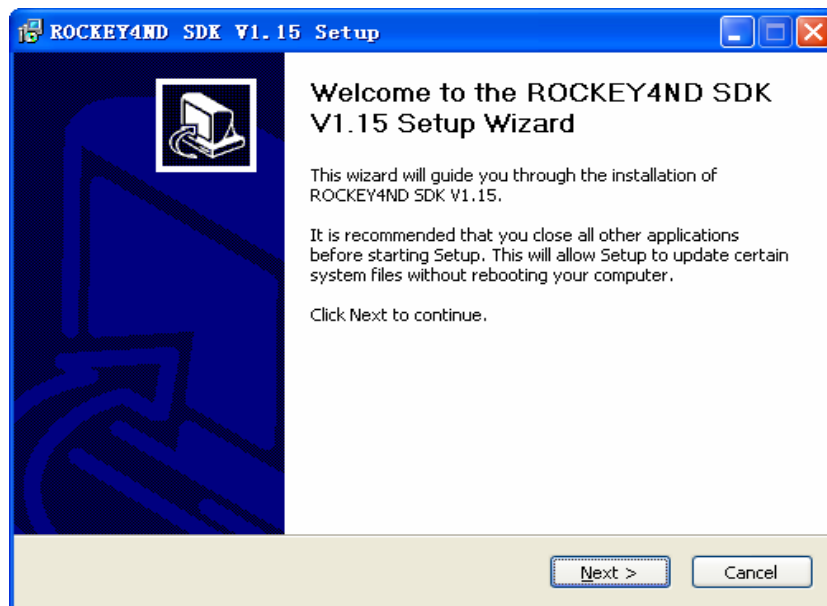


Figure 3.2

Step 3.

View the software release agreement.

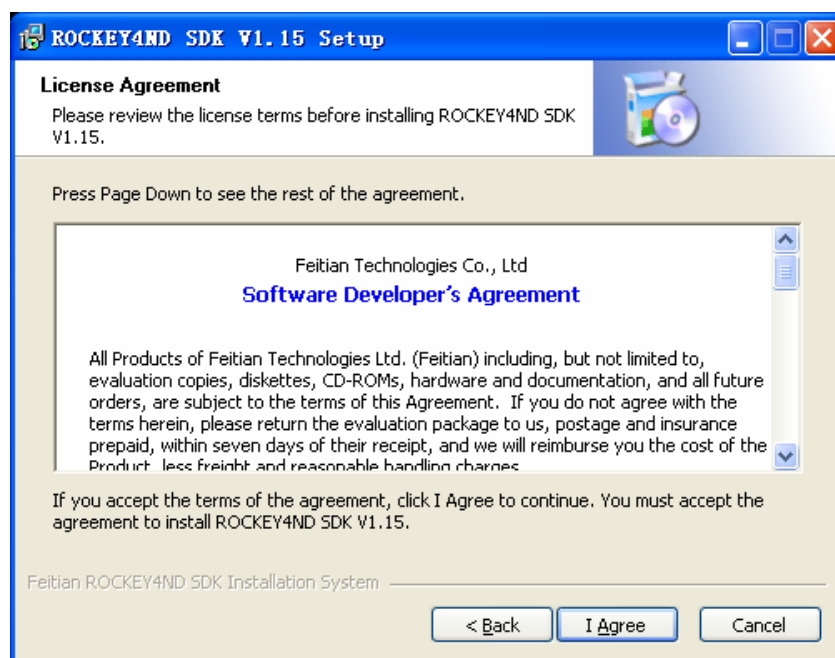


Figure 3.3

Step 4.

Figure 3.4 show the type of installation. Please select the content you want to install.

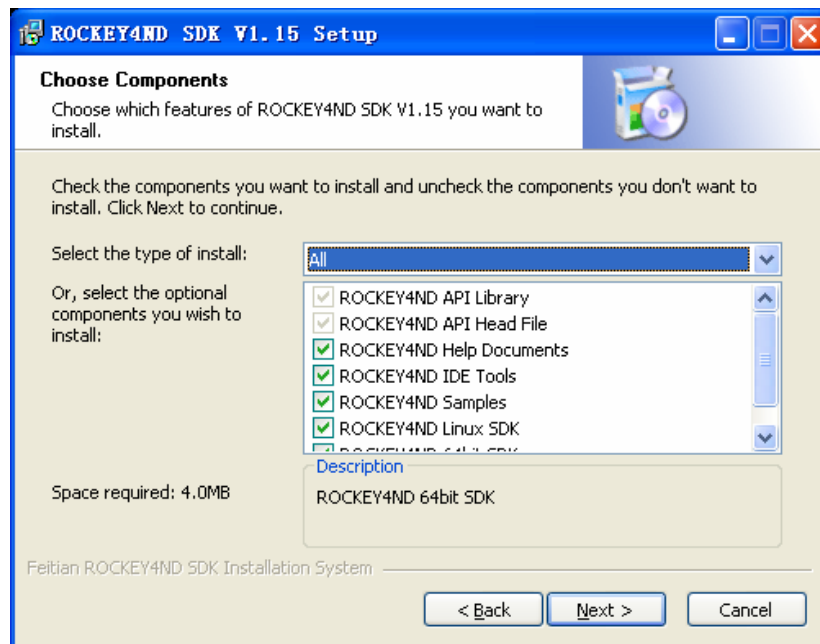


Figure 3.4

Step 5.

Select the path to install the SDK.

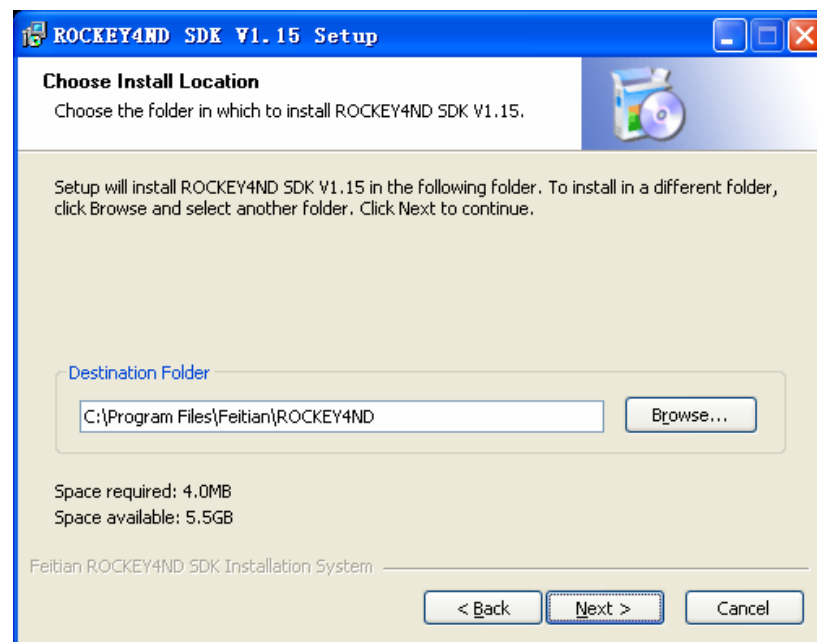


Figure 3.5

Step 6.

Finish installing the SDK.

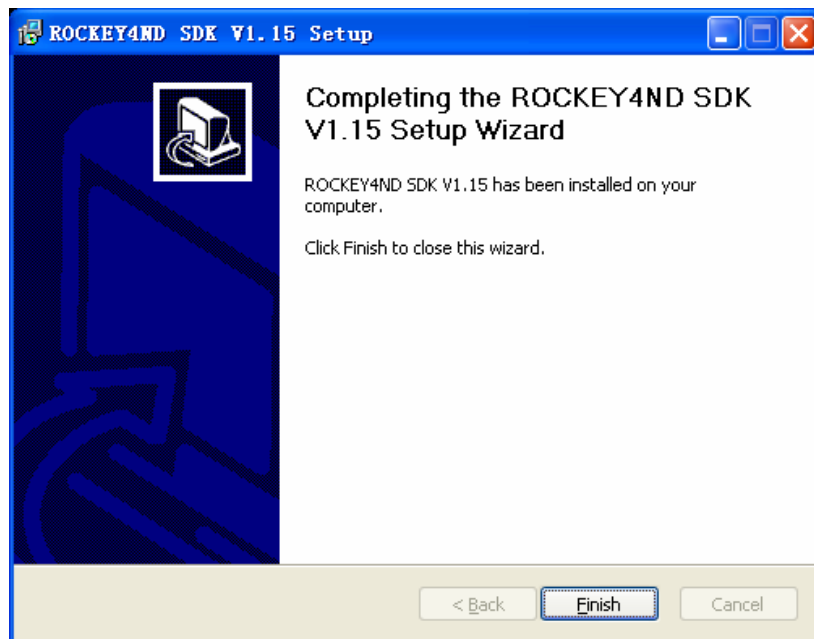


Figure 3.6

3.3 Uninstalling the Development Package

You may use Add or Remove Programs from the Windows Control Panel or select “FEITIAN” and then “Uninstall” from the Windows Start menu to uninstall the installed components.

Chapter 4 Concepts for Developers

This chapter covers the basic concepts and functions of the ROCKEY4ND software protection system. All ROCKEY users should read this chapter carefully to familiarize themselves with ROCKEY.

4.1 Passwords

When developers purchase ROCKEY they will get 4 16-bit passwords. The first two are Basic passwords (first grade passwords); the last two are Advanced passwords (second grade passwords). The 4 passwords for the demo dongles in the SDK are: P1: C44C, P2: C8F8, P3: 0799, P4: C43B. The passwords are “burned” into the hardware so that neither the user nor the manufacturer may change them. The developers must input the 4 passwords correctly to have full access to the dongles. The developer should set any reference to the Advanced password set to zero in the application program that is delivered to the end user – you should never reveal the Advanced passwords to the end user in any form. The Basic passwords allow the end users to access all necessary ROCKEY functions. We will discuss when one should input the Basic passwords, and when both Basic and Advanced passwords are required in the chapters that follow.

4.2 Customer Code

The Customer Code is five to seven characters in length and corresponds to a unique customer password set. You may use the Customer Code for reordering ROCKEY4ND to be sure that all of the units in your inventory are consistent.

4.3 Hardware ID

FEITIAN will burn a globally unique Hardware Identification (HID) number into each ROCKEY4ND dongle. The HID cannot be changed. You may use the HID to positively identify an individual ROCKEY4ND.

The HID is readable with the Basic passwords. It is impossible to write HID even if you have the advanced passwords.

4.4 User Data Zone

The User Data Zone (UDZ) is a memory space that the developer can use to store data needed by the software protection system. Users can read from and write to this space at any time. The total UDZ is 1000 bytes.

The UDZ is divided into 2 parts.

The low part (0-499 bytes): Users with any level of passwords have full permission (read/write). The high part (500-999 bytes): Users with basic passwords (password 1 and password 2) can only read the UDZ. Users with advanced password (password 3 and password 4) have full permission (read/write).

4.5 Module Zone

The Module Zone was designed for multi-module encryption. It may be used to store module specific data for Envelope encryption and/or API calls.

A ROCKEY4ND module is a 16-bit protected memory space. There are 64 “modules” in each ROCKEY4ND dongle, so as many as 64 application modules may be protected with a single ROCKEY4ND dongle. The developer may write data into the ROCKEY4ND modules and then use that data, along with ROCKEY4ND functions, to create powerful and flexible software protection systems. If the content of the module is not “0” you can use the module; if it is “0” you cannot use the module. You may determine if a module is useable by analyzing the attributes of the module. The exact content can only be determined algorithmically.

ROCKEY4ND modules cannot be read and it can only be written with Advanced passwords.

4.6 Module Attributes

There are two attributes associated with each ROCKEY4ND module: “Zero Value” attribute and “Decrement” attribute. A 16-bit protected memory space stores an attribute of the module. The value stored in the “Zero Value” attribute indicates if the value in the associated module is “0” or not “0”. “1” indicates not “0” and that the module is usable; “0” indicates “0” and that the module is not usable. The “Decrement” attribute indicates if the value stored in the associated module can be decreased. “1” indicates it can be decreased. “0” indicates it cannot be decreased.

The “Zero Value” attribute can be read with the Basic passwords and cannot be written with Advanced passwords.

The “Decrement” attribute can be read with the Basic passwords and can be written with the Advanced passwords.

4.7 User Algorithm Zone

The User Algorithm Zone (UAZ) is a user-defined area for instruction storage. The number of instructions that may be stored in the UAZ varies according to the ROCKEY4ND model. ROCKEY4ND supports a maximum of 128 instructions. (Please refer to Chapter 8 ROCKEY4ND Hardware Algorithms.)

The User Algorithm Zone (UAZ) cannot be read and may only be written with Advanced passwords.

4.8 User ID

The User ID is a 32-bit memory allocation that may be used to store an application serial number or other identification information.

It may be read with the Basic passwords and written with the Advanced passwords.

4.9 Random Number

ROCKEY4ND can generate a true random number from its hardware. The random number can be used to prevent tracing or used in hardware algorithms.

4.10 Seed and Return Values

ROCKEY4ND contains a proprietary algorithm that will generate four 16-bit return values from input of a 32-bit seed code and the Basic/Advanced passwords. ROCKEY dongles with the same passwords should return the same values if the seed codes are the same. The return values will be different for ROCKEY dongles with different Basic/Advanced passwords.

Chapter 5 ROCKEY4ND Editor

5.1 Brief Introduction

You may use the ROCKEY4ND Editor to edit data stored in ROCKEY4ND, test its functions or write in batch. The Editor is a convenient tool for learning to use ROCKEY4ND and its edit operations. To run the Editor, select “Editor” under “Tools”. The ROCKEY4ND Editor interface is organized into five parts: Tool Bar & Pull down Menu, Status Bar, Device Selector, Operation Status Log and Operation Main Window. See Figure 5.1.

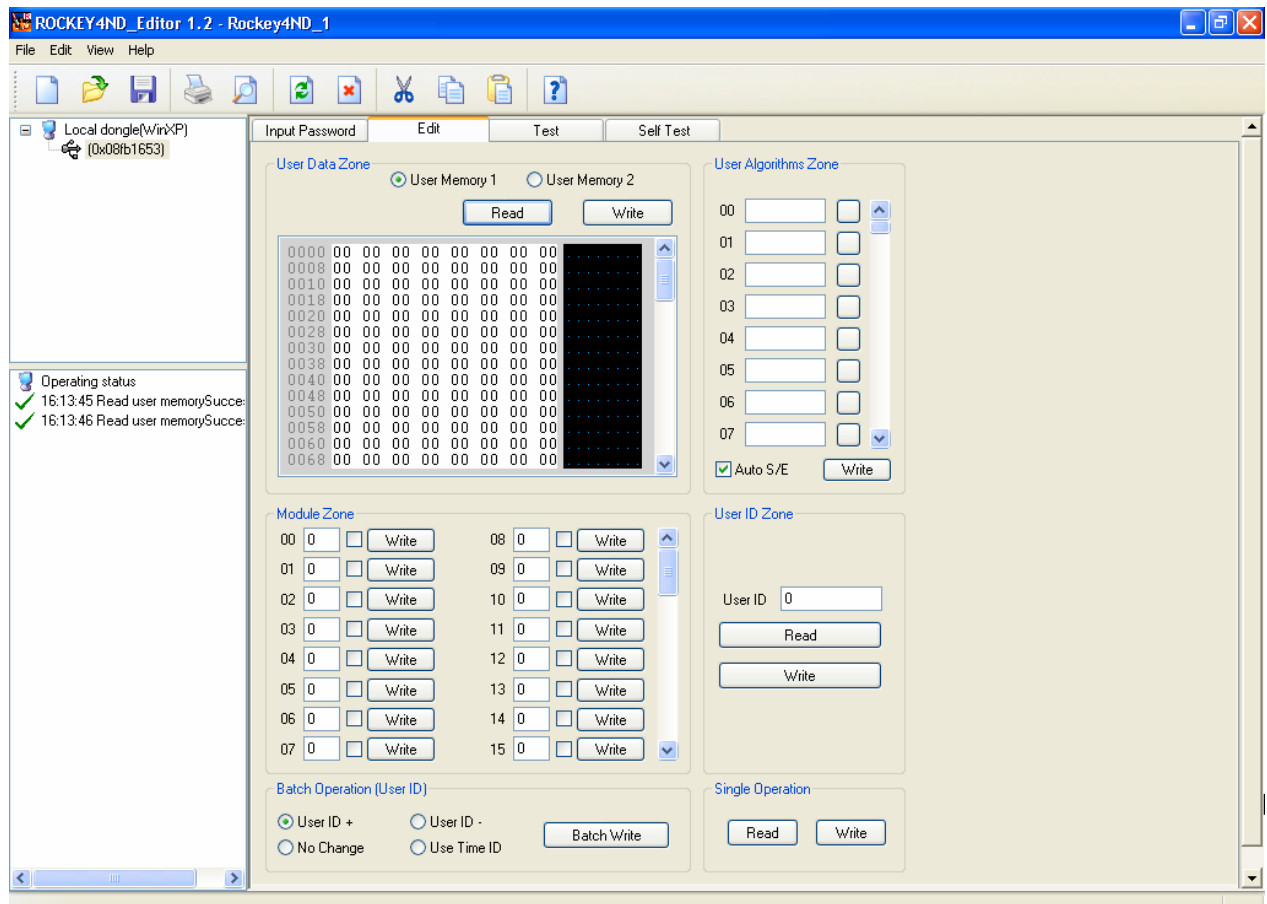


Figure 5.1

1. **Tool Bar & Pull down Menu** - This is the very topmost section of the screen. The typical Windows functions can be invoked from the icons or pull-down menus, such as print, save and refresh. Shortcut keys and icons are also offered. See Figure 5.2 and 5.3.

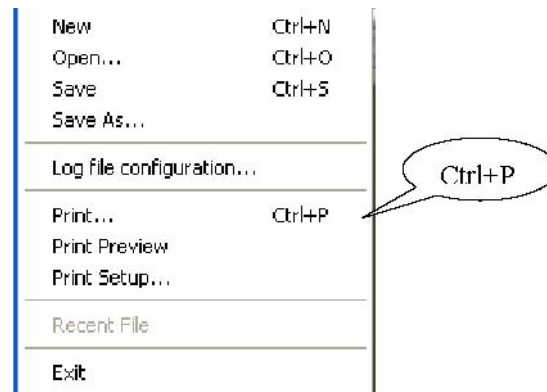


Figure 5.2 Figure 5.3



2. **Status Bar** -The Status Bar is at the bottom of the screen. The Status Bar message is for the dongle selected in the “Device Selector” (See below) portion of the screen. Status messages are: Read, Write and Ready. See Figure 5.4.

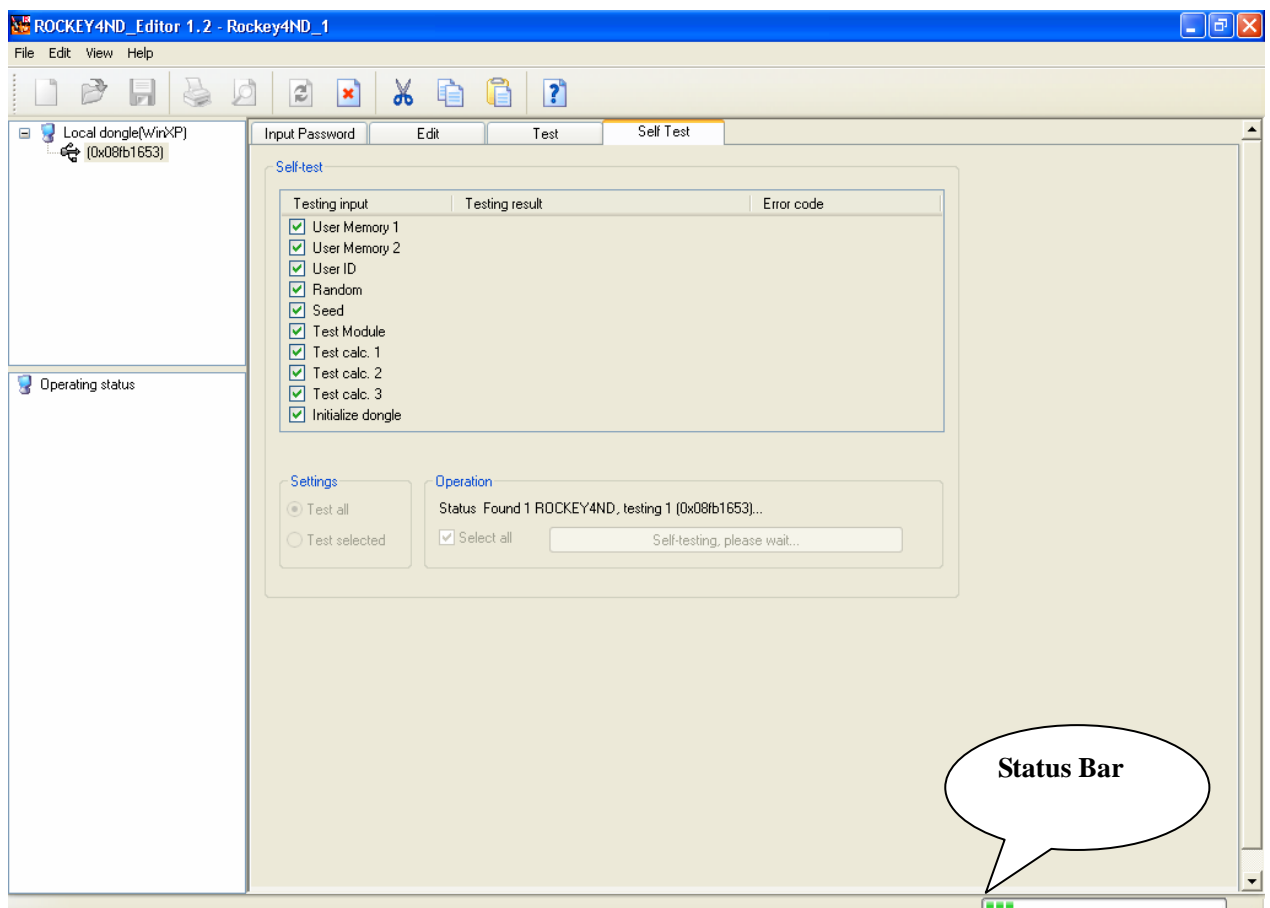


Figure 5.4

3. **Device Selector** - This is the upper left portion of the screen and shows the current OS version and ROCKEY4ND dongles that are attached to the computer. See Figure 5.5.

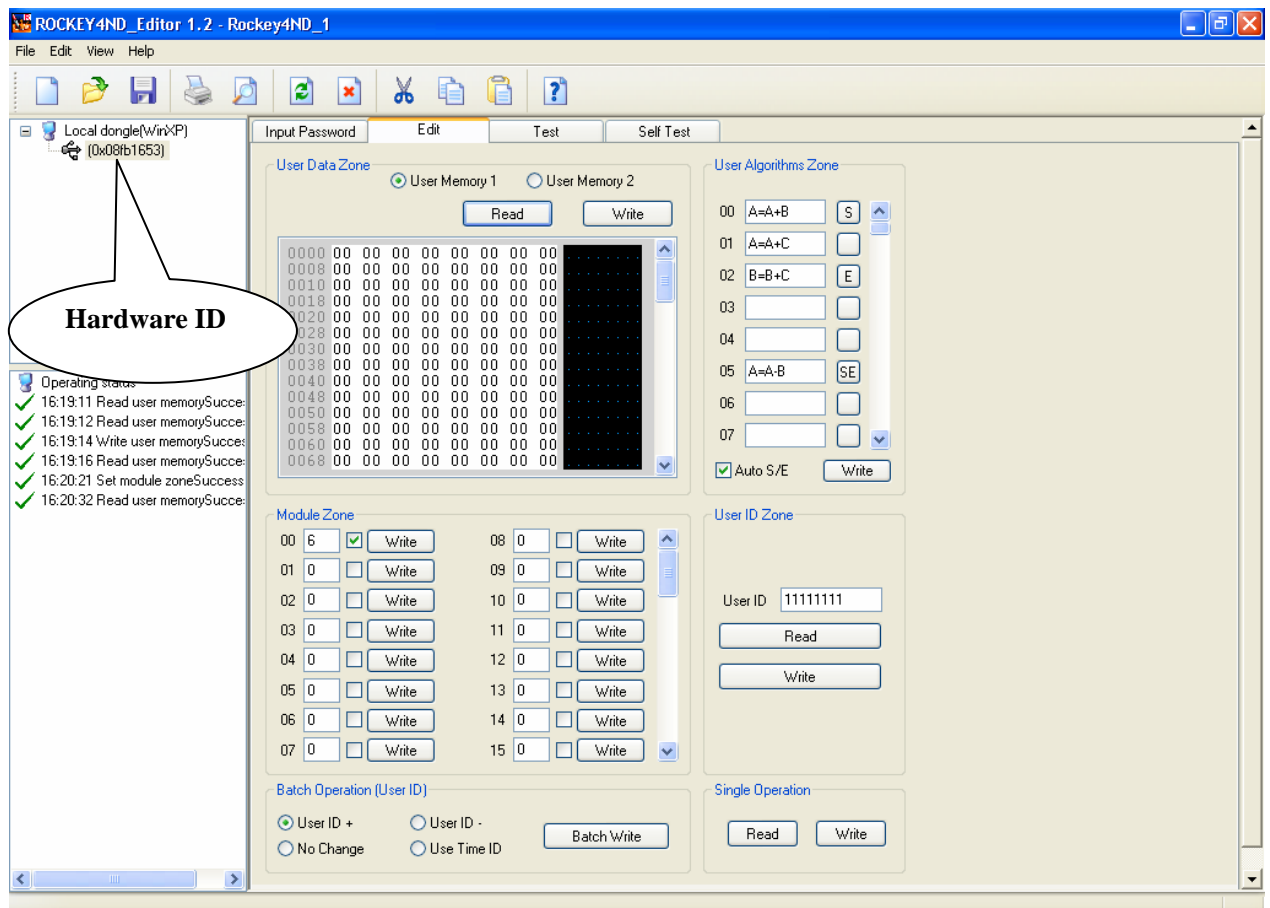


Figure 5.5

4. **Operation Status** - The time, results and error prompt of the previous operations will display here. This section is the lower left portion of the screen. See Figure 5.6.

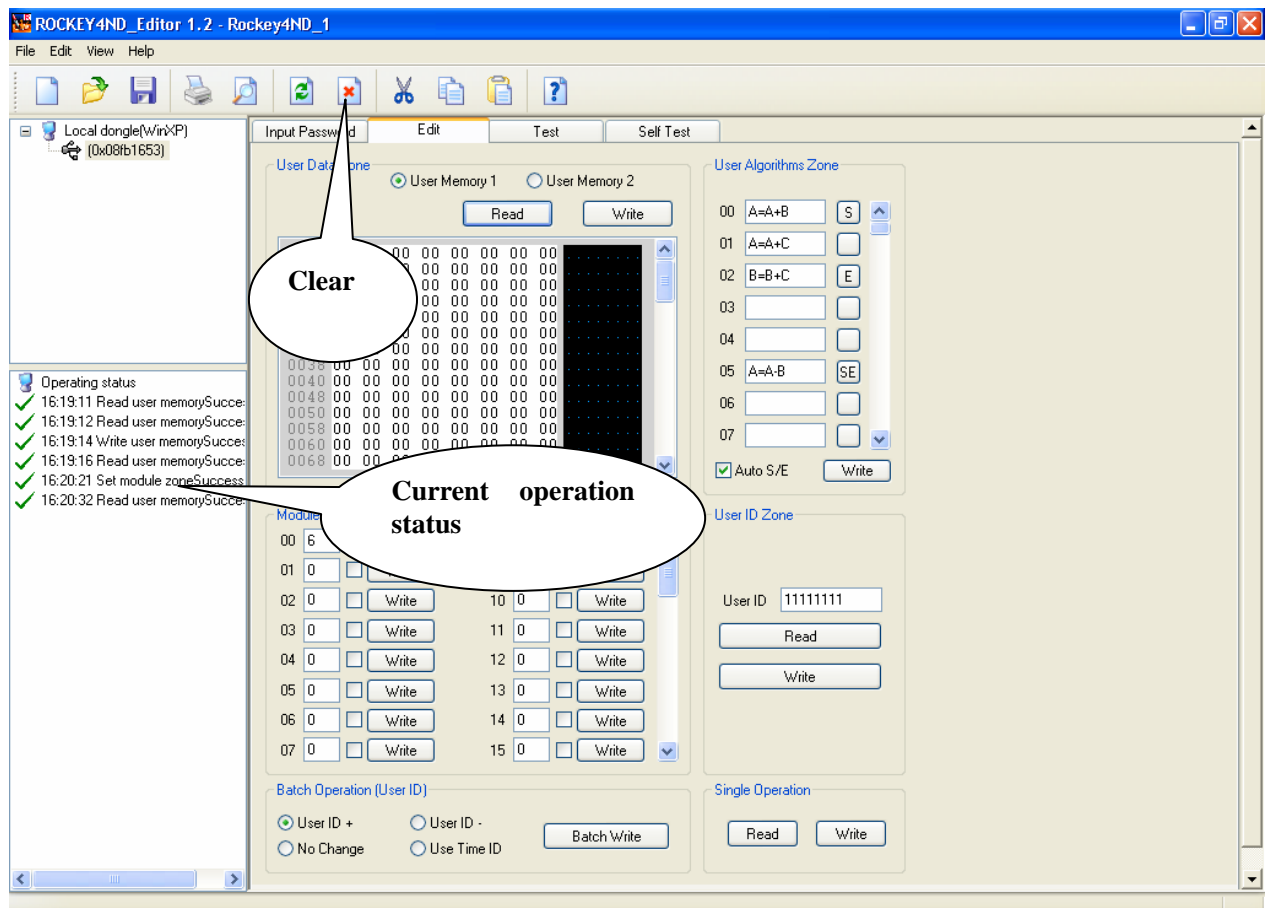


Figure 5.6

5. **Operation Main Window** - The Operation Main Window has five selection tabs: Password, Edit, Test, Self Test and Batch Write. Each tab corresponds to a screen and a function.

Template files (.rki) can be opened by dragging and dropping the template file to the open Editor window. It can also be opened from the file pull down menu in the Editor or by clicking the file from Explorer. You may print preview the template file and print it out. You may use the Editor without a ROCKEY4ND dongle attached to the computer and save the results to a template file. The template file can later be used for a “Write” or “Batch Write” to a ROCKEY4ND dongle(s). The template file may be updated with the Editor while dongles are attached. A progress bar will display all your operation progress and you may stop your operations at any time.

Notes:

All numbers are input and displayed in hexadecimal with the exception of the number of generated seed codes in the test screen.

5.2 Operation

1. Input Password

You may enter the Basic and Advanced passwords as shown in Figure 5.7.

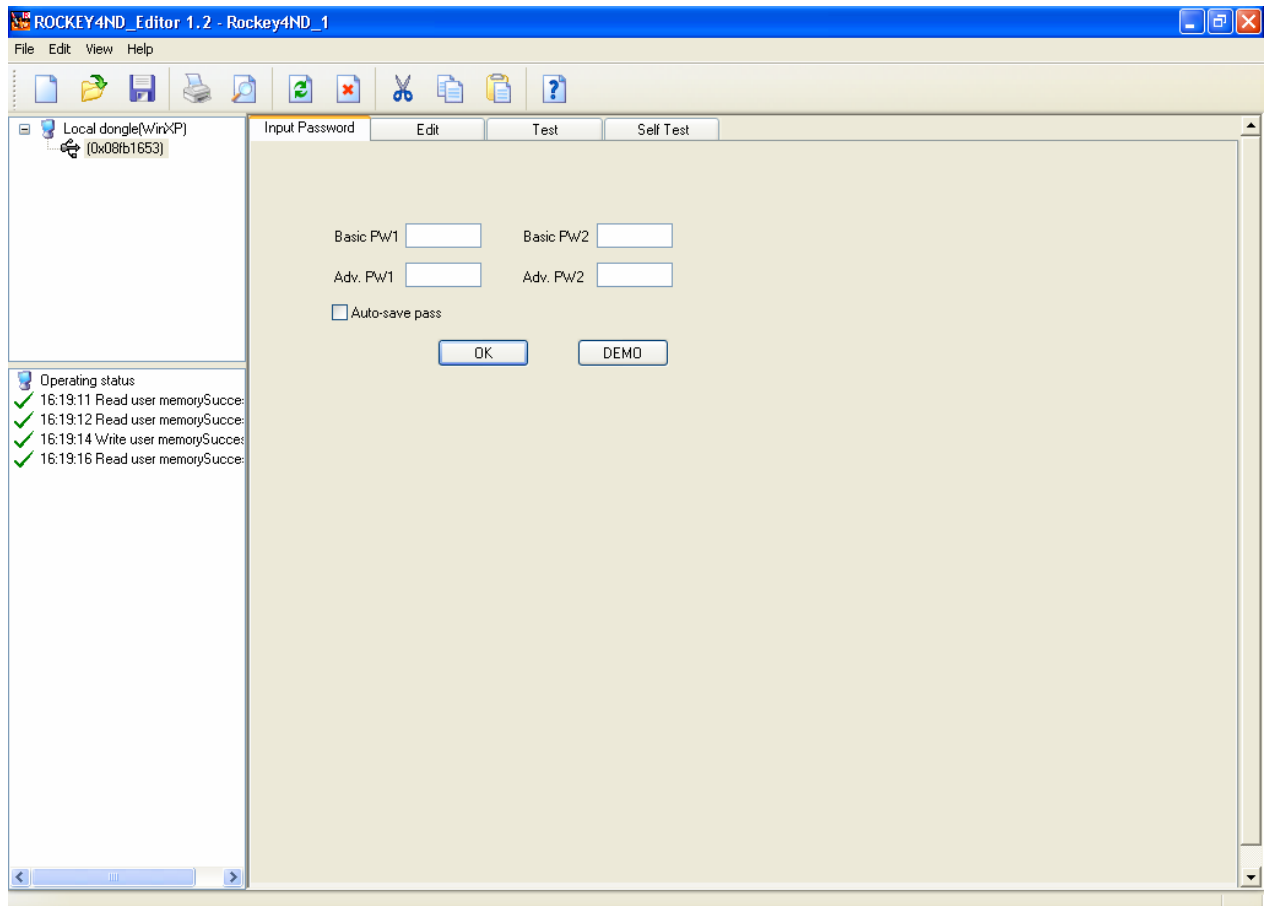


Figure 5.7

Make sure you enter the correct passwords. If the Basic passwords are incorrect Editor cannot find the dongle. If the Basic passwords are correct, and the Advanced passwords are invalid, the Editor should find the dongle and allow Read functions, but it will not allow Write functions.

If you click “DEMO” button, you may perform any operations on DEMO dongles. The 4 passwords for DEMO dongles are: P1: C44C, P2: C8F8, P3: 0799, P4: C43B.

The passwords will be saved automatically when you choose “Auto save password”. This function avoids future password entry errors.

If the entered password information corresponds with the attached ROCKY4ND dongle, selecting the “OK” or “Demo” button will take you to the “Edit” screen. The system will automatically begin to search for attached dongles.

Note:

You may edit, save, open and print template files without inputting the passwords. However, you cannot operate the dongle without at least the Basic passwords. Entering the Basic passwords will allow you to both edit template files and perform Read operations on the corresponding attached dongle.

2. Edit ROCKY

The ROCKEY Hardware ID (HID) is displayed for all found dongles. The HID is globally unique and cannot be changed. See Figure 5.8:

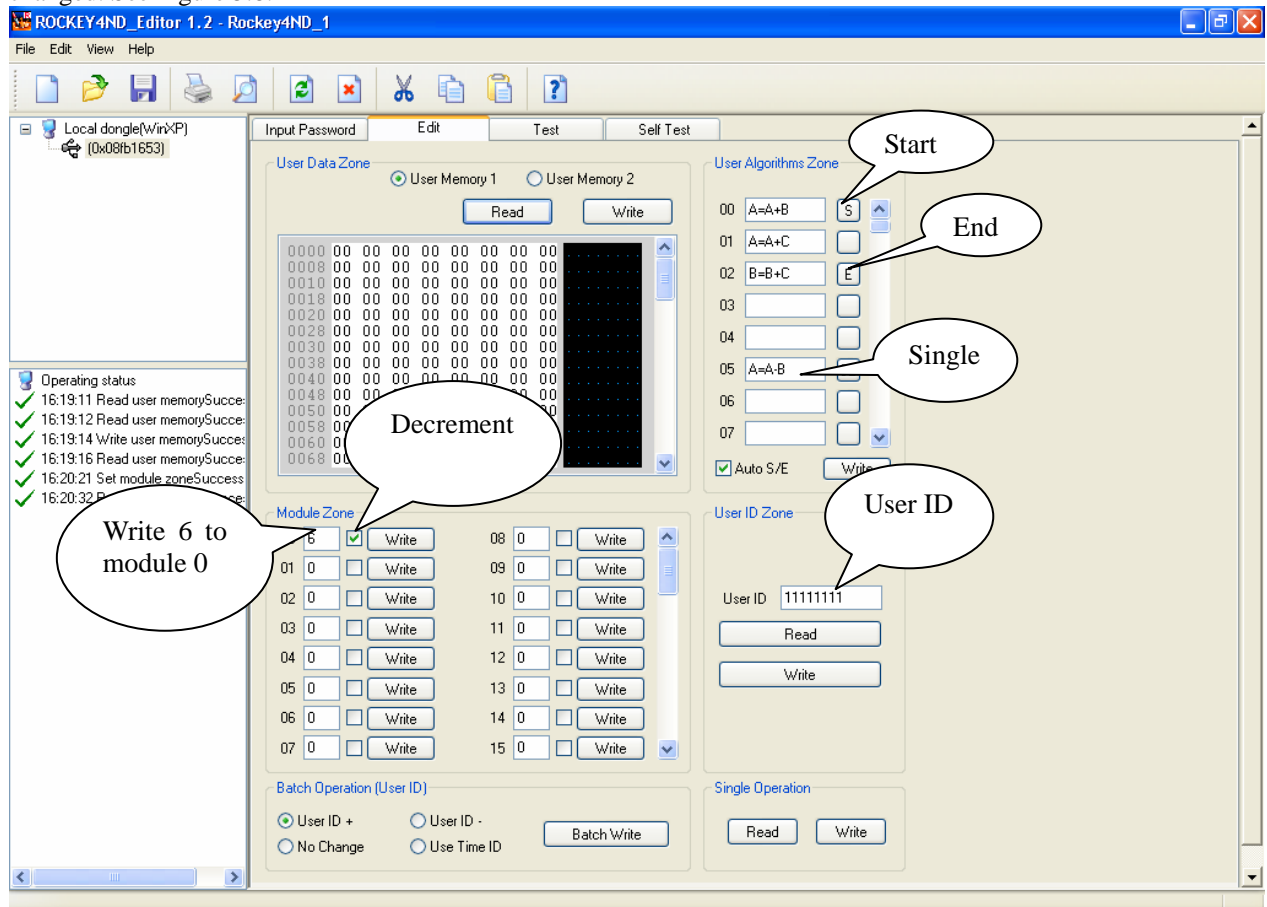


Figure 5.8

Here you may edit the specified ROCKEY. There are four components to the Edit screen: User Data Zone (UDZ), Module Zone, User Algorithm Zone (UAZ) and the User ID Zone (UIZ).

User Data Zone (UDZ) – The UDZ is a user defined memory space. Data may be entered here in hexadecimal or ASCII text in the field provided. Click the “Read” button to read data from the UDZ and “Write” to write to the UDZ. If you click the Read or Write button a progress bar will appear. After the operations are finished the results will be displayed in the Operation Status section. See Figure 5.9.

Module Zone - This part of the screen is used to update the values and decrement attributes of the ROCKEY4ND modules. To add new values to a module simply enter the new value in the field of the module, and click “Write”. The Decrement attribute can likewise be altered. (All 64 ROCKEY4ND modules are displayed here, labeled 0 to F in hexadecimal.)

User Algorithm Zone (UAZ) – User defined algorithms may be written here. The algorithms consist of operands and operators, such as $A=A+B$ (Please refer to Chapter 8 ROCKEY4ND Hardware Algorithms). The function of the small button to the right of each instruction is to set the Start/End attribute. An algorithm will begin with an instruction marked “S” and end with one marked “E”. Single instruction algorithms will have the attribute value of “SE”. A null value attribute will be assigned to any instruction that is not explicitly the start or end of an algorithm. The “Auto S/E” option will automatically assign an attribute of “S” to the first instruction of a string and “E” to the

last instruction

– and a null attribute to all attributes between the start and end of the algorithm or “SE” if there is only one instruction in the string. Click the “Write” button to write the instructions to the UAZ in the ROCKEY4ND dongle. See Figure 5.9.

User ID Zone (UIZ) - User identification information may be read from or written to the UIZ of the ROCKEY4ND dongle in hexadecimal. See Figure 5.9.

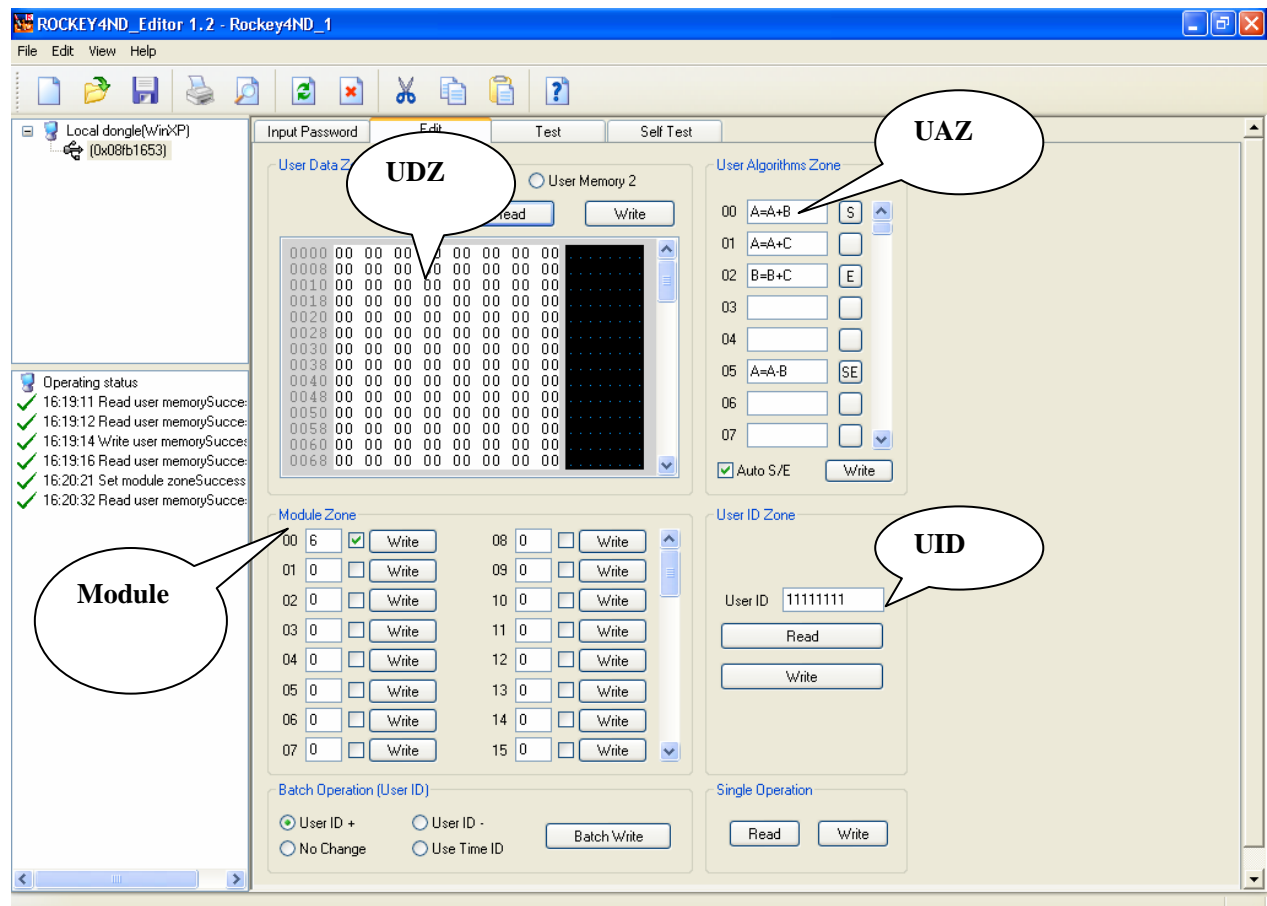


Figure 5.9

3. Test ROCKEY

There are five components to the Test screen: User Data Zone (UDZ), Calculation Zone, User ID Zone (UIZ), Module Attribute Zone and the Seed Calculation Zone. See Figure 5.10.

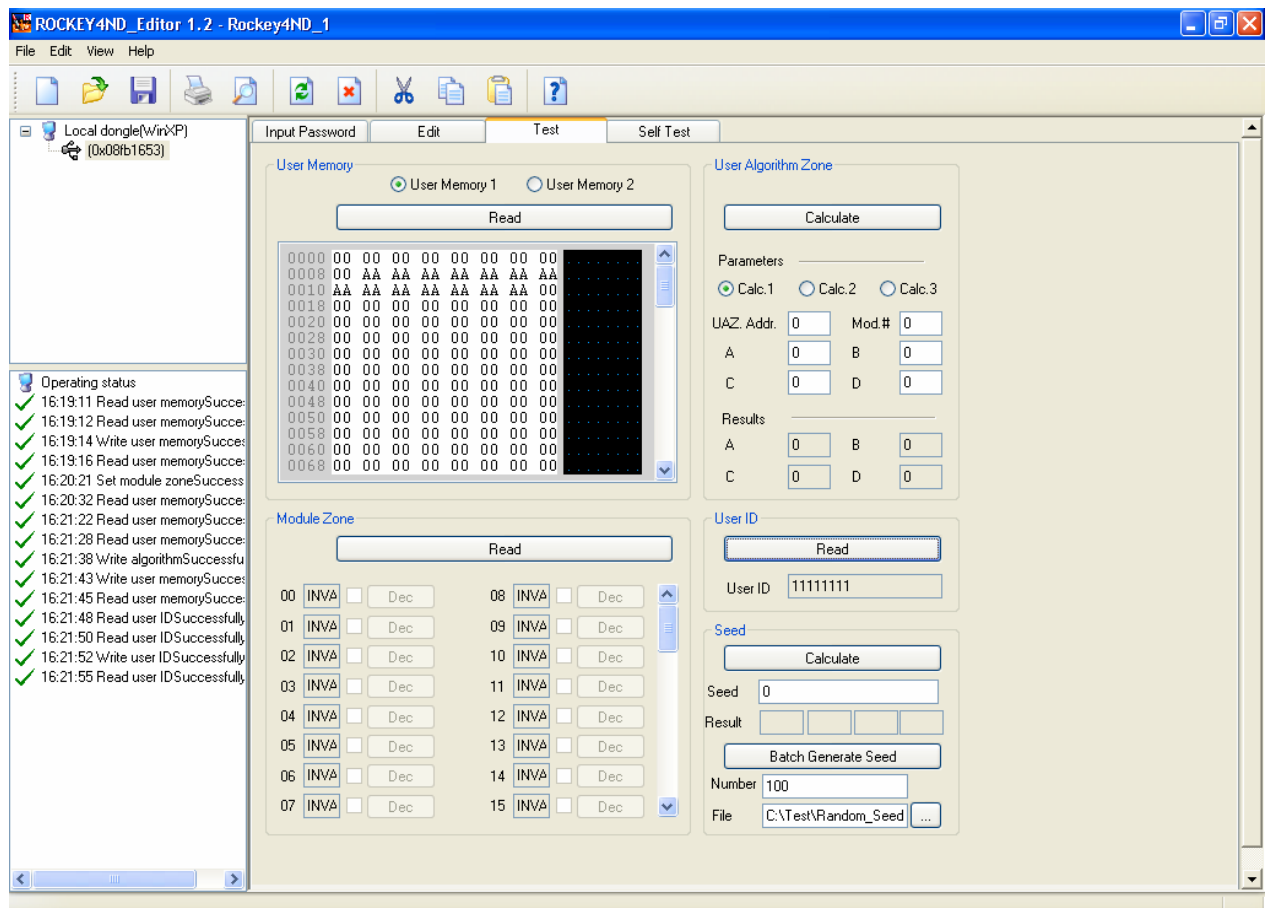


Figure 5.10

User Data Zone –The UDZ is a user defined memory space. Data may be displayed in hexadecimal form, or as ASCII text. Click the “Read” button to read data from the UDZ. You may view hexadecimal data or ASCII text here.

Calculation Zone – Be sure that you are familiar with the calculation functions before using the Calculation Zone. First select the calculation you would like to test (For Calc1 and Calc3 a “Module” entry box will appear. For Calc2 a “Seed Code” box will appear.). Then input the start address of your algorithm stored in the UAZ. The start address is where the instruction is marked with “S” or “SE”. Enter hexadecimal input values to the parameter A, B, C and D. Enter the module number or seed code and click the “Calculate” button. The results of the operation will be written to the parameters listed in the “Results” section of the Calculation Zone.

User ID Zone (UIZ) – Click the “Read” button to read the user defined ID from the UIZ of the ROCKEY4ND dongle. UIZ is 32 bits in length.

Module Attribute Zone – This zone indicates the status of the Zero Value and Decrement attributes of the ROCKEY4ND modules. Click the “Read Module Attributes” button to update this portion of the Test screen. “N/A” means that the Zero Value attribute is “0”. “Valid” means the Zero Value attribute is not “0”. If the “Dec” button is grayed out the module cannot be decremented. If it is to be decremented, clicking the “Dec” button will reduce the value stored in the module by “1”.

Seed Calculation Zone – There are two small sections to the Seed Calculation Zone. The top section will display four calculated seed results for any entered seed code. Enter a decimal number in the “Number Generate” field in the

bottom section, and that same number of random seed codes and corresponding results will be written to a text file defined in the “File” field. The default file is C:\seed.txt. See Figure 5.11.

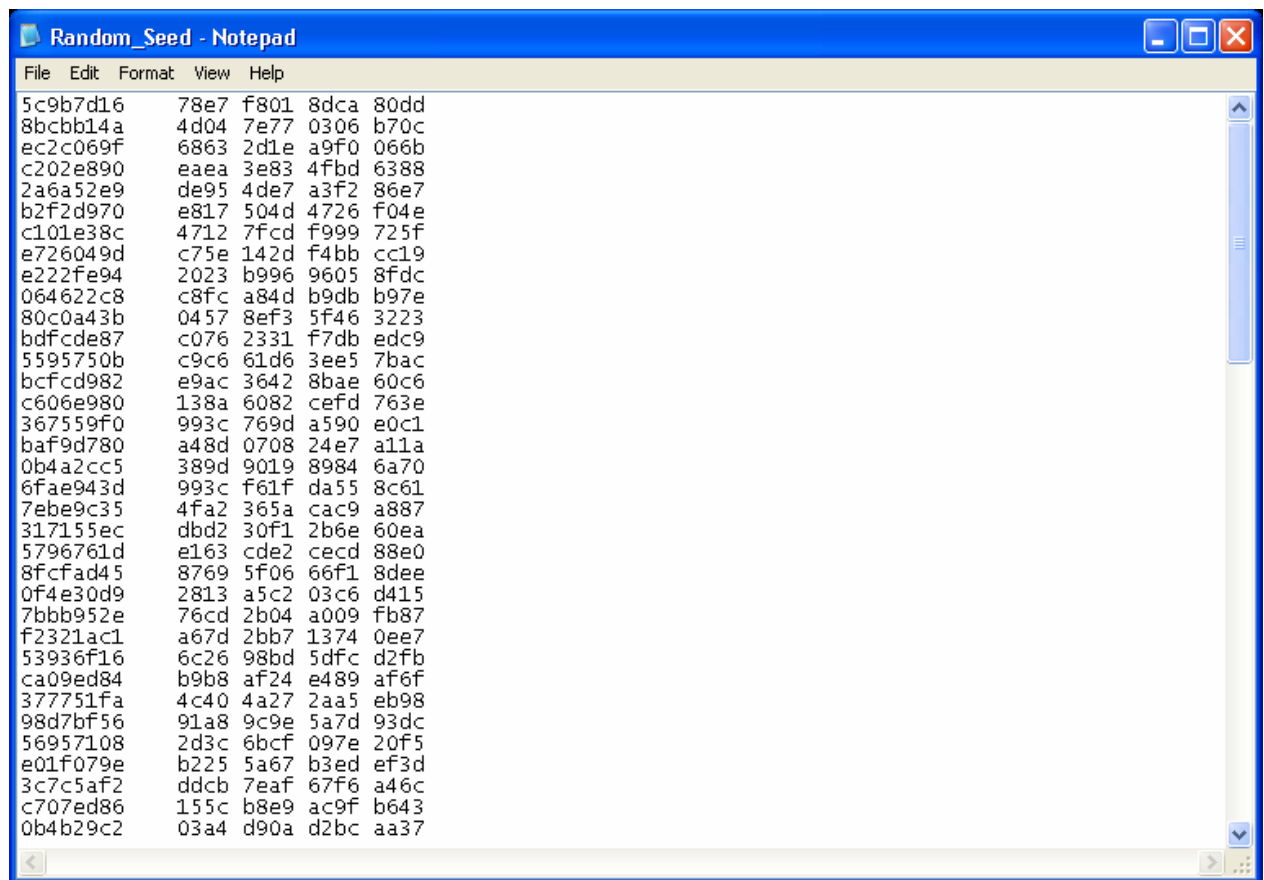


Figure 5.11

4. Self Test

Self test all ROCKEY dongles attached to the computer. See Figure 5.12.

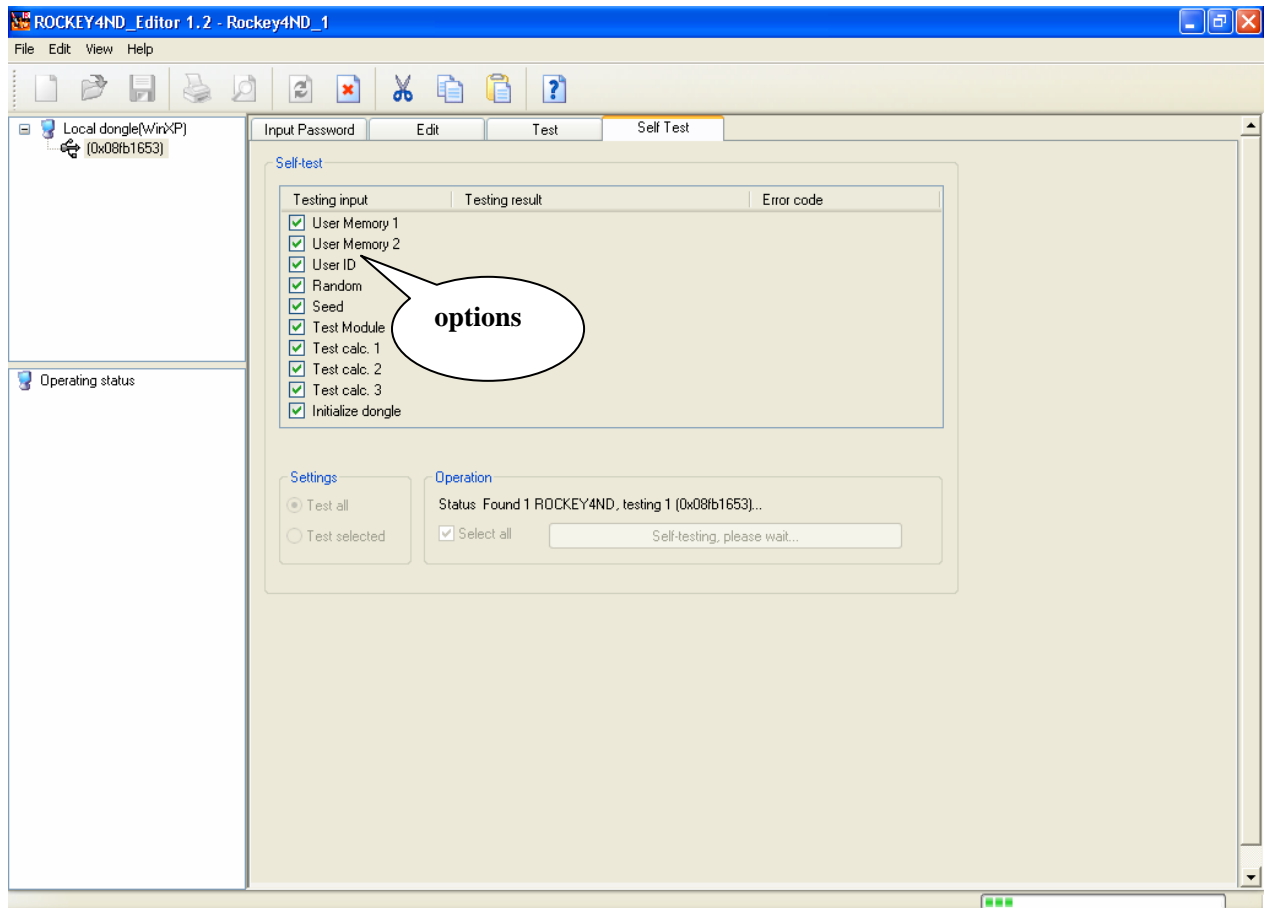


Figure 5.12

The Self Test screen will test all ROCKEY dongles automatically and write the Hardware ID (HID) to a file named “log.txt” in the current directory. (Please refer to Section Log File and File Setting)

Note:

This test is like a “Format” command in that it will delete any data or parameters stored in the dongle. It would be best to run the Self Test upon receipt of the dongle or if there is a significant problem with the dongle.

5. Batch Write

There are six components to the Batch Write screen: User Data Zone (UDZ), Module Zone, User Algorithm Zone (UAZ), User ID Zone (UIZ), Single Operation Zone and the Write All Zone. See Figure 5.13.

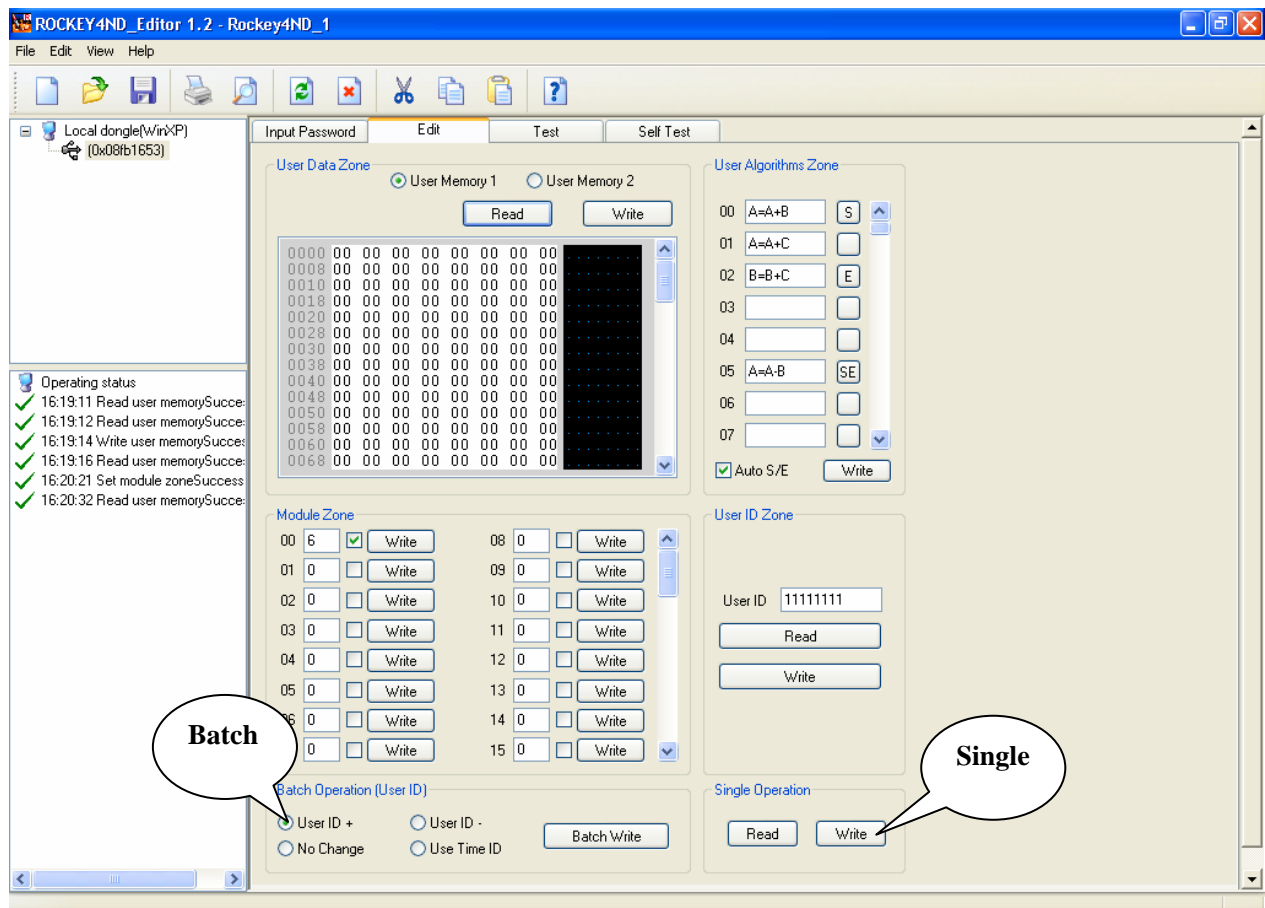


Figure 5.13

User Data Zone (UDZ) – See the previous introduction. **Module Zone** – See the previous introduction. **User Algorithm Zone (UAZ)** – See the previous introduction. **User ID Zone (UIZ)** - See the previous introduction. **Single Operation Zone** – Select the “Read” button to read the contents of the ROCKEY dongle to memory. Select the Write button to write the contents of memory to ROCKEY. **Write All Zone** – This section will change the UIDs of all of the ROCKEY4ND dongles displayed in the Device Selector section. Enter the starting UID in the “User ID” field and then click one of the “User ID” setting buttons (The setting buttons for the UID Zone are described below.) Then click “Write All” to update the attached dongles:

User ID +: This button will write the value entered in the “User ID” field to the top most dongle in the Device Selector section, and increase that value by one for the subsequent dongle. It will continue to increase by “1” and write the new value to each dongle listed in the Device Selector section. For example, if the number “123” were entered into the User ID field and there were three dongles in the Device Selector section - selecting “User ID +” and clicking “Write All” would write “123” to the top most dongle. The next dongle would have a UID of “124” and the last a UID of “125”.

User ID –: This has the same effect as the “User ID +” field, except the UID is decreased instead of increased.

Use Time ID: This button will alter all of the UIDs by a value taken from the system clock.

No Change: The UID entered into the “User ID” field will be written to all of the dongles in the Device Selector section.

5.3 Template Storage

You may save the template on the disk or print out for backup. See Figure 5.14.

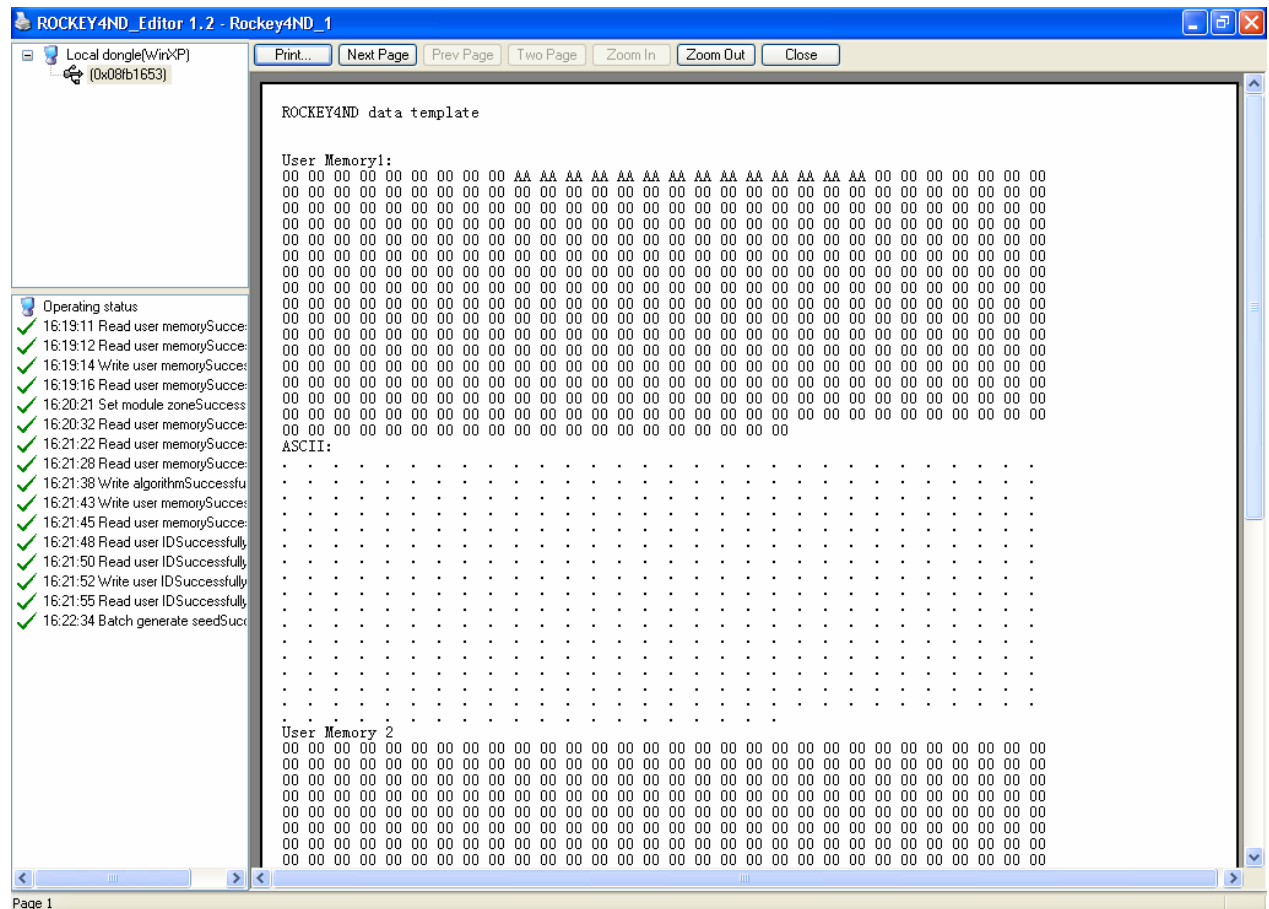


Figure 5.14

5.4 Log File and File Setting

The ROCKEY4ND HID and User ID processed by the Self Test and Batch Write screens can be recorded in a log file. This function is enabled by default. You can disable the Log Function in the “Log file configuration” screen that is under the “File” pull-down menu. The log function may be disabled here and the log file name altered. Please see Figure 5.15 and 5.16.

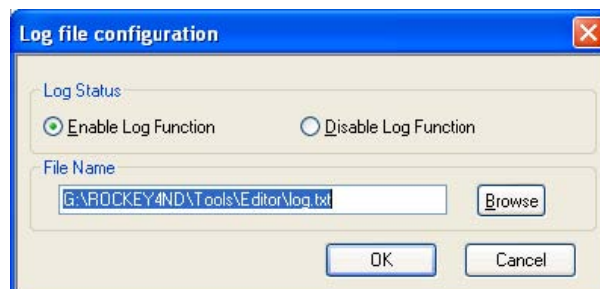


Figure 5.15

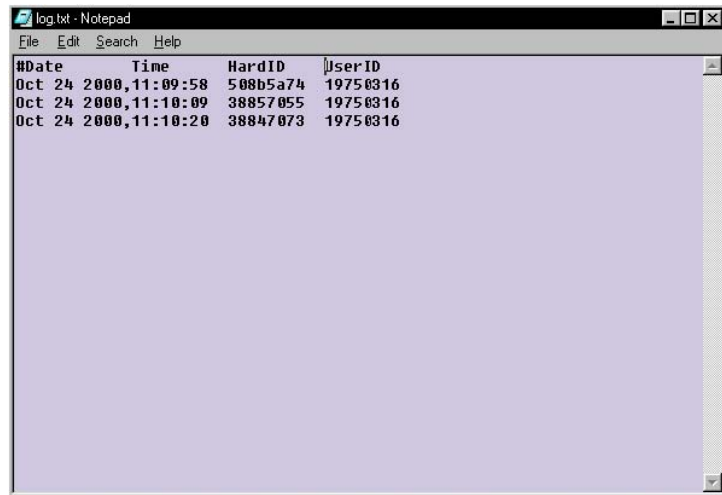


Figure 5.16

5.5 Editor Update

You may click the “Help” button to check the update information of the editor. See Figure 5.17.



Figure 5.17

Note: If our functions cannot meet all your requirements or you need some expanded functions, please contact FEITIAN. We will do our best to perfect it in future versions.

Chapter 6 ROCKEY4ND Envelope Encryption

The ROCKEY4ND Envelope program may be used for direct encryption of Win32 Portable Executable (PE) files (such as .exe, .dll or .arx). Envelope encryption is a good solution if you do not have the source code or the time to use the API functions. Envelope encryption only works with 32-bit applications. For increased software protection it is strongly suggested to use both the Envelope and API implementations.

RyEnv32_ND.exe under the directory “*TOOLS\ENVELOP*” is the ROCKEY4ND Envelope encryption tool. Its main interface is shown below in Figure 6.1.

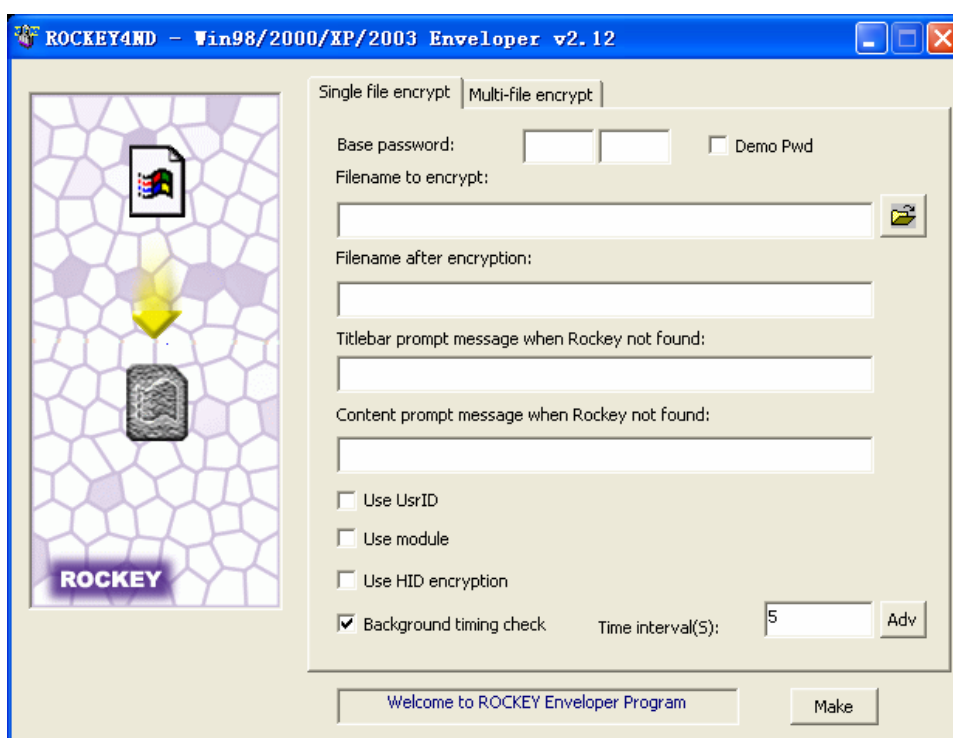


Figure 6.1

The “Single File Encrypt” program will be the most efficient protection method if only several files need to be encrypted. If many PE files need to be protected use the “Multi-File Encrypt” program.

Note: Please back up your files before you encrypt them with the Envelopee program (*RyEnv32_ND.exe*). Envelope encryption allows you to encrypt your files many times and in different ways with one ROCKEY. If you encrypt the files with the evaluation default settings, software encrypted with one ROCKEY can work with another ROCKEY with the same passwords.

If you plan to protect your software with both Envelope encryption and API encryption, please call the API first and then encrypt the software with the Envelope program.

6.1 Single File Encryption

6.1.1 Encrypt with Default Settings

The Basic passwords (the first two passwords) must be entered into the fields shown in Figure 6.2 before you can proceed. The passwords are masked with “*”. Select the file to be encrypted with the “browse” button or enter its name in the “File name to encrypt:” field. Then enter the new encrypted file name and path into the “File name after encrypt” field. You also can select the time interval to check if the ROCKEY4ND is attached. See Figure 6.2.

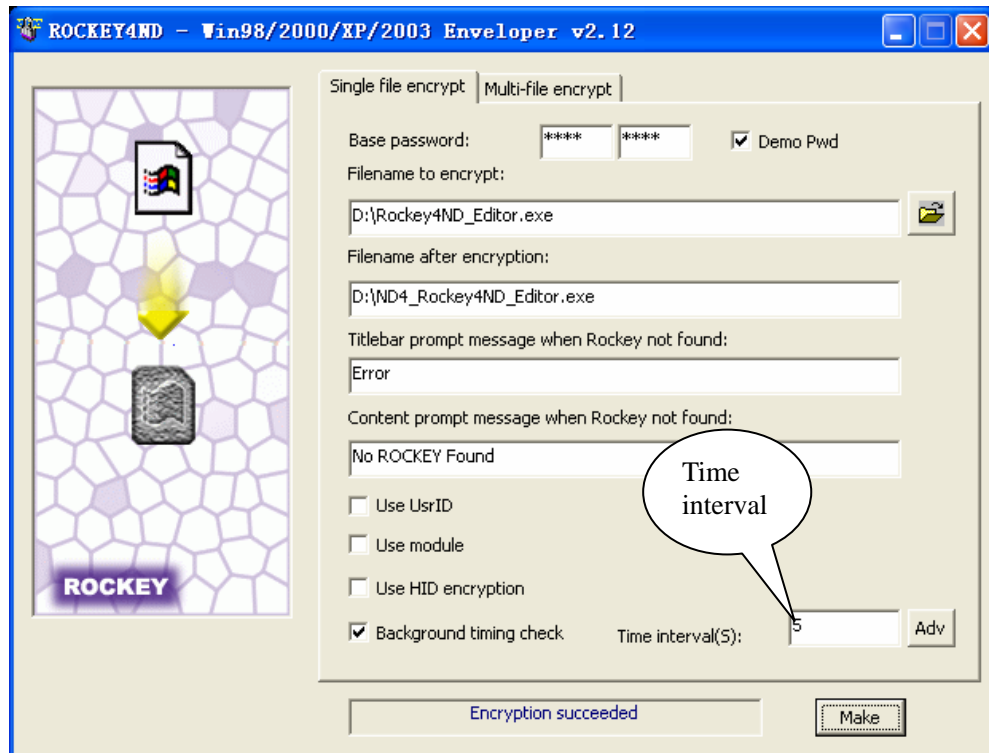


Figure 6.2

Note: The dialog box below will appear if you perform an encryption function without the correct ROCKEY4ND dongle attached to the computer. When the correct ROCKEY is not found, the dialog box below will appear the message “Cannot find ROCKEY4ND”

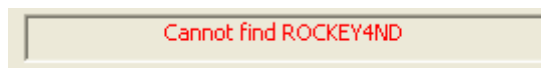


Figure 6.3

Click the “Make” button. A progress bar will appear and then the “File encryption succeeded” message will appear after a successful encryption. See Figure 6.4.

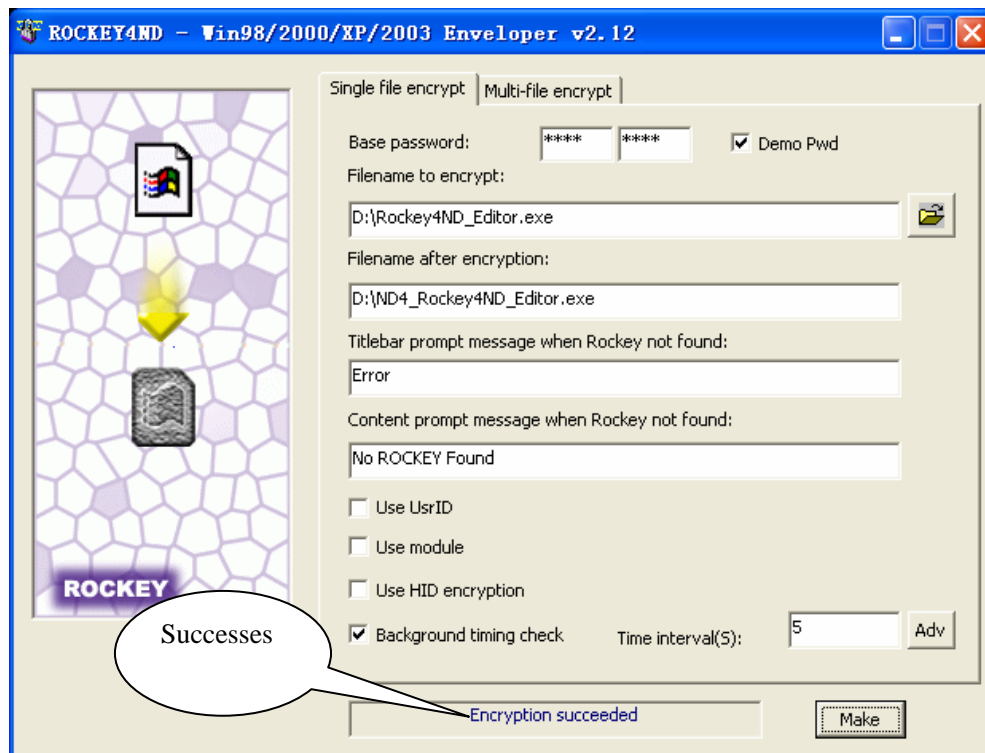


Figure 6.4

6.1.2 Encrypt with Module

If you choose “Use module encryption”, the system will not only check whether the dongle exists and whether the passwords are valid, but also requires that the Zero Value attribute is not equal to “0”. To use module encryption you must write one non-zero value in the corresponding module with the Editor Program and select “Use module encryption” from the Envelope encryption program. Be sure to enter the appropriate module number. See Figure 6.5.

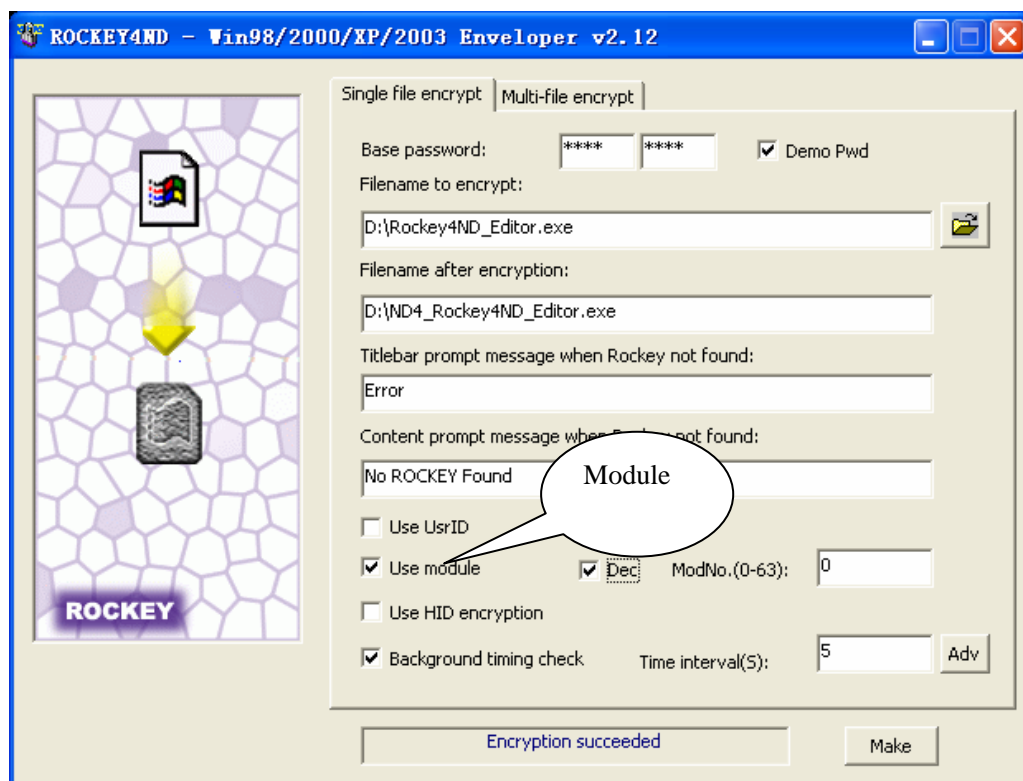


Figure 6.5

Note, please read the document for advanced function (set run-timelimitation with envelopeer) in *\Tools\Envelope*

6.1.3 Encrypt with HID

If you choose “Use HID encryption” the HID of the attached ROCKEY4ND dongle will be copied to the Envelope program when you click the “Make” button. Thereafter, the Envelope program (and thus the application) will only run with that specific HID ROCKEY4ND dongle attached to the computer. See Figure 6.6.

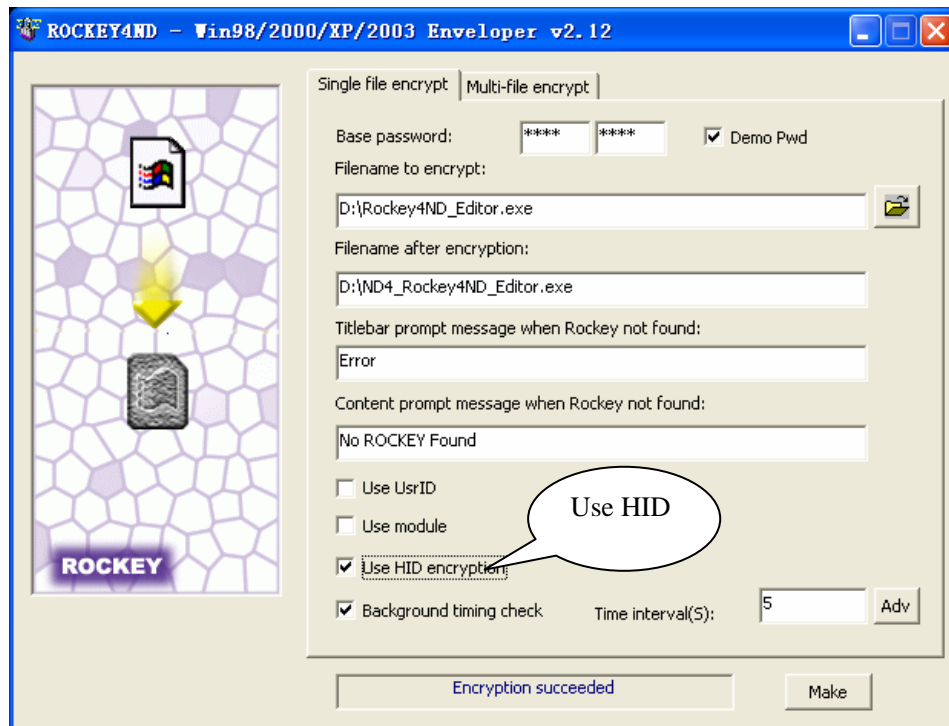


Figure 6.6

Note:

It is recommended that you use module encryption to enhance security. Combining that with HID protection will improve security even further.

Using Envelopee encryption with default settings will allow end users to access two or more of your applications with a single dongle. Most developers prefer that a dongle will allow end users to access only one specific application. Use module encryption to create a one-to-one relationship between a dongle and an application. For example, use module “0” to protect application A. Use module “1” to protect application B. In this way as many as sixteen different applications may be tied to a dongle coded for only that specific application. Developers can achieve similar results by ordering dongles with password sets specified for an application. But that approach is administratively more difficult and not recommended.

6.1.4 Encrypted with UID

You may enter a range of User ID. When the protected application runs, it not only checks if there is a ROCKEY4ND, but also checks if there is a valid UID. Only when a ROCKEY4ND dongle with a valid (pre-defined) UID is presented, the application works. You can use editor to edit the UID.

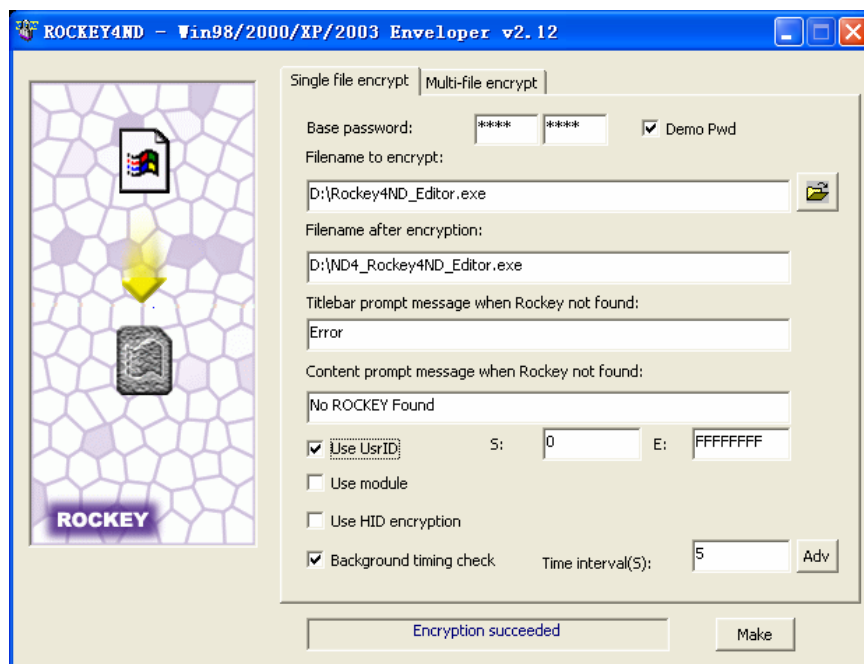


Figure 6.7

6.2 Multi File Encryption

“Multi-File Encrypt” is an ideal solution when developers need to encrypt many programs or several versions of the same program. See Figure 6.8.

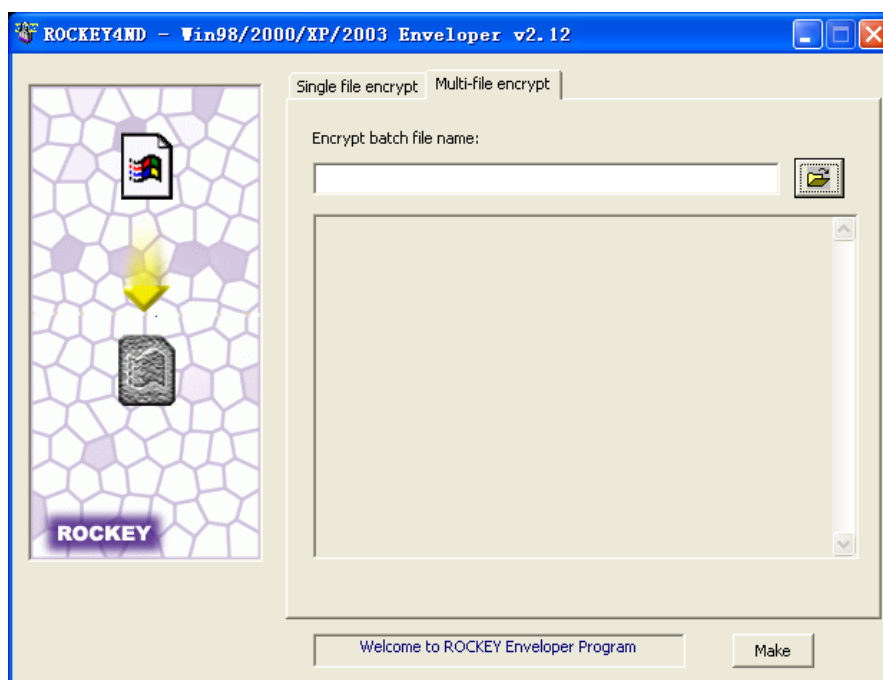


Figure 6.8

First you use a text editor to write a script file with the extension “.rbt”. There is a sample of *Ryenv32_ND.rbt* under “TOOLS\ENVELOP”:

[Common]

Password1 = c44c

```
Password2 =c8f8
Password3 = 0
Password4 = 0
MessageTitle = "Rockey4ND Dongle"
MessageContent = "ROCKEY4ND not found or Driver not installed"
```

```
[EncryptFile]
```

```
InputFile = f:\windows\system32\calc.exe
OutputFile =z:\calc.exe
TimeCheck = 5
```

```
[EncryptFile]
```

```
InputFile = f:\windows\system32\notepad.exe
OutputFile =z:\notepad.exe
UseHardID = YES
ModuleNo = 0
```

Its structure is like a Windows .INI file. There is a [Common] section we call the “general definition” section. Each file that needs to be encrypted has its own [EncryptFile] section. If a parameter is not defined in an [EncryptFile] section, its default value will come from the [Common] section as shown in the example above. Since the passwords of all files to be encrypted and the error information to be displayed are the same, they are all defined in the [Common] section. If a user needs a different prompting message, he/she may simply alter the MessageTitle and MessageContent in the [EncryptFile] section.

All definable parameters are listed below:

Password1	Password 1
Password2	Password 2
Password3	Password 3 (“0”, Advanced passwords not needed)
Password4	Password 4 (“0”, Advanced passwords not needed)
InputFile	Input file name
OutputFile	Output file name
UseHardID	Encryption with hardware ID (Default value is “NO”)
ModuleNo	Module number, checks Zero Value attribute of selected module
MessageTitle	Error message title
MessageContent	Error message body

Chapter 7 ROCKEY4ND API

The ROCKEY4ND Application Programming Interface (API) is the most flexible and powerful means of protecting your software. The security level of your software is determined by how you implement the API. The API set has been simplified and is intended to make the ROCKEY4ND programming effort as effective as possible.

API encryption enables you to call ROCKEY in your application to enhance its security level. You may check the existence of the dongle anywhere in your application and take actions as a result of the check. You may also check the data you stored in the UDZ.

You may use the Editor program to set and write data to the modules, write algorithms to the User Algorithm Zone (UAZ), user information to the User ID zone (UID) or take other actions. All such operations may be performed with the API.

We will take the interface of the C language to demonstrate how to call the API. Similarly, you may call other language interfaces the same way.

7.1 ROCKEY4ND Function Prototype and Definition

```
WORD Rockey
(
WORD function,
WORD* handle,
DWORD* lp1,
DWORD* lp2,
WORD* p1,
WORD* p2,
WORD* p3,
WORD* p4,
BYTE* buffer
);
```

FEITIAN provides developers with a unified function from which they can employ all ROCKEY4ND operations. This function is defined as a multi-function function.

Below is a call example for C language, and we will discuss future applications in a similar way. retcode = Rockey(function,handle,lp1,lp2,p1,p2,p3,p4,buffer);

The “ROCKEY” function parameters are defined as: Note: All interface parameters must be defined in your program. ROCKEY4ND cannot transfer NULL or 0 pointers. Use of Null or 0 pointers in your program will generate an error.

Parameter Name	Parameter Type	Parameter Meaning
Function	A 16-bit number	API function
Handle	Address of a 16-bit number	ROCKEY4ND session address
lp1	Address of a 32-bit number	long parameter 1
lp2	Address of a 32-bit number	long parameter 2
p1	Address of a 16-bit number	parameter 1
p2	Address of a 16-bit number	parameter 2
p3	Address of a 16-bit number	parameter 3
p4	Address of a 16-bit number	parameter 4
Buffer	Address of a 8-bit number	Buffer

- 1 “function” is a 16-bit number. It stands for the specific function and it is defined below:
- 2 “handle” is the pointer for ROCKEY operation’s handle.
- 3 “lp1” and “lp2” are the pointers for long integer parameters. Their content depends on the functions.
- 4 “p1”, “p2”, “p3” and “p4” are the pointers for short integer parameters. Their content depends on the functions.
- 5 “buffer” is the pointer for character buffer. Its content depends on the functions.

RY_FIND	1	// Find ROCKEY4ND
RY_FIND_NEXT	2	// Find next ROCKEY4ND
RY_OPEN	3	// Open ROCKEY4ND
RY_CLOSE	4	// Close ROCKEY4ND
RY_READ	5	// Read ROCKEY4ND
RY_WRITE	6	// Write ROCKEY4ND
RY_RANDOM	7	// Generate Random Number
RY_SEED	8	// Generate Seed Code
RY_WRITE_USERID	9	// Write User ID
RY_READ_USERID	10	// Read User ID
RY_SET_MOUDLE	11	// Set Module
RY_CHECK_MOUDLE	12	// Check Module
RY_WRITE_ARITHMETIC	13	// Write Arithmetic
RY_CALCULATE1	14	// Calculate 1
RY_CALCULATE2	15	// Calculate 2
RY_CALCULATE3	16	// Calculate 3
RY_DECREASE	17	// Decrease Module Unit

7.2 ROCKEY4ND API Services

Here we discuss the API services in detail. The following functions marked with *[*]* require the two Advanced passwords.

Note: p3 and p4 are Advanced passwords. They are for developers to operate on the dongle. The Advanced passwords should not appear in the software you offer to your customers and you should set the two Advanced passwords “0” when searching for dongles in your application.

1. Find a ROCKEY4ND dongle (RY_FIND)

Objective: To check if a specific ROCKEY4ND is attached to the USB port.

Input parameters:

function = RY_FIND

*p1 = Password 1

*p2 = Password 2

*p3 = Password 3 (optional)

*p4 = Password 4 (optional)

Return value:

A return value = “0”

indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the ROCKEY4ND Hardware ID (HID) to *lp1.

2. Find the Next ROCKEY4ND dongle (RY_FIND_NEXT)

Objective: To check if another specific ROCKEY4ND is attached to the USB port.

Input parameters:

function = RY_FIND_NEXT

*p1 = Password 1

*p2 = Password 2

*p3 = Password 3 (optional)

*p4 = Password 4 (optional)

*lp1 = The hardware ID of the last dongle found by RY_FIND or RY_FIND_NEXT

Return value:

A return value = “0”

indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the ROCKEY4ND Hardware ID (HID) to *lp1.

3. Open the ROCKEY4ND dongle (RY_OPEN)

Objective: To open a ROCKEY4ND dongle with specified passwords or hardware ID.

Input parameters:

function = RY_OPEN
*p1 = Password 1
*p2 = Password 2
*p3 = Password 3 (optional)
*p4 = Password 4 (optional)
*lp1= Hardware ID

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will write the handle address to the *handle parameter

4. Close the ROCKEY4ND dongle (RY_CLOSE)

Objective: To close a ROCKEY4ND dongle with a specific handle.

Input parameters:

function = RY_CLOSE
*handle = ROCKEY4ND's handle

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error.

5. Read the ROCKEY4ND dongle (RY_READ)

Objective: To read the contents of the User Data Zone (UDZ).

Input parameters:

function = RY_READ
*handle = ROCKEY4ND's handle
*p1 = off set of UDZ(zero base)
*p2 = length (unit is byte)
buf = address of buffer

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the contents of the UDZ written to the memory buffer.

6. Write to the ROCKEY4ND dongle (RY_WRITE)

Objective: To write data to the User Data Zone. (UDZ)

Input parameters:

function = RY_WRITE
*handle = ROCKEY4ND's handle
*p1 = off set of UDZ
*p2 = length (unit is byte)
buf = address of buffer

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error.

7. Generate a Random Number (RY_RANDOM)

Objective: To get a random number from the dongle.

Input parameters:

function = RY_RANDOM
*handle = ROCKEY4ND's handle

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the *p1 address populated with the random number.

8. Generate Seed Code Return Values (RY_SEED)

Objective: To get return codes from the input of a seed code.

Input parameters:

function = RY_SEED
*handle = ROCKEY4ND's handle
*lp2 = Seed Code

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the following addresses populated with seed code return values: *p1 = Return Code 1 *p2 = Return Code 2 *p3 = Return Code 3 *p4 = Return Code 4

9. Write the User ID [*] (RY_WRITE_USERID)

Objective: To write the user defined "User ID" to the User ID Zone (UIZ).

Input parameters:

function = RY_WRITE_USERID
*handle = ROCKEY4ND's handle
*lp1 = User ID

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error.

10. Read User ID (RY_READ_USERID)

Objective: To read the user defined "User ID" from the User ID Zone (UIZ).

Input parameters:

function = RY_READ_USERID
*handle = ROCKEY4ND's handle

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the *lp1 address populated with the User ID.

11. Set a ROCKEY4ND Module [*] (RY_SET_MOUDLE)

Objective: To write a value to a specific ROCKEY4ND module and set the Decrement attribute.

Input parameters:

function = RY_SET_MOUDLE
*handle = ROCKEY4ND's handle
*p1 = Module Number
*p2 = Module Unit Value
*p3 = If decreasing is allowed (1 = allowed, 0 = not allowed)

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in module unit # "*p1" storing value "*p2" and the Decrement attribute set to "0" or "1".

12. Check a ROCKEY4ND Module (RY_CHECK_MOUDLE)

Objective: To read the attributes of a specific ROCKEY4ND module.

Input parameters:

function = RY_CHECK_MOUDLE
*handle = ROCKEY4ND's handle

*p1 = Module Number

Return value:

A return value = "0"

indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in "*p2" populated in the value from the Zero Value attribute (1 = module value is not zero), and "*p3" populated with the value from the Decrement attribute (1 = module can be decreased).

13. Write Arithmetic [*] (RY_WRITE_ARITHMETIC)

Objective: To write user-defined mathematical instructions to the User Algorithm Zone (UAZ).

Input parameters:

function = RY_WRITE_ARITHMETIC

*handle = ROCKEY4ND's handle

*p1 = position of first instruction in UAZ buffer = buffer address of the algorithm command string

Return value:

A return value = "0"

indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the UAZ populated with the algorithm command string from the buffer.

14. Calculate 1 (RY_CALCULATE1)

Objective: To return the results of a calculation performed in ROCKEY4ND, using input provided by the developer and the RY_CALCULATE1 function.

Input parameters:

function = RY_CALCULATE1

*handle = ROCKEY4ND's handle

*lp1 = Start point of calculation

*lp2 = Module number

*p1 = Input value 1

*p2 = Input value 2

*p3 = Input value 3

*p4 = Input value 4

Return value:

A return value = "0"

indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the calculation.

15. Calculate 2 (RY_CALCULATE2)

Objective: To return the results of a calculation performed in ROCKEY4ND, using input provided by the developer and the RY_CALCULATE2 function.

Input parameters:

function = RY_CALCULATE2
*handle = ROCKEY4ND's handle
*lp1 = Start point of calculation
*lp2 = Seed Code (32-bit)
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the calculation.

16. Calculate 3 (RY_CALCULATE3)

Objective: To return results of a calculation performed in ROCKEY4ND, using input provided by the developer and the RY_CALCULATE3 function.

Input parameters:

function = RY_CALCULATE3
*handle = ROCKEY4ND's handle
*lp1 = Start point of calculation
*lp2 = Module number
*p1 = Input value 1
*p2 = Input value 2
*p3 = Input value 3
*p4 = Input value 4

Return value:

A return value = "0"
indicates that the function worked correctly. Any other return value indicates an error. A successful operation will result in the addresses p1, p2, p3 and p4 populated with the results of the calculation.

17. Decrease Module Unit (RY_DECREASE)

Objective: To decrease the value in a specified ROCKEY4ND module by "1".

Input parameters:

function = RY_DECREASE

*handle = ROCKEY4ND's handle

*p1 = Module number

Return value:

A return value = "0"

indicates that the function worked correctly. Any other return value indicates an error. A successful operation will reduce the value stored in module *p1 by "1".

7.3 Return Codes

RETURN CODE	Value	DESCRIPTION
ERR_SUCCESS	0	Success
ERR_NO_ROCKEY	3	No ROCKEY4ND dongle
ERR_INVALID_PASSWORD	4	Found ROCKEY4ND dongle, but Basic passwords are incorrect
ERR_INVALID_PASSWORD_OR_ID	5	Wrong password or ROCKEY4ND HID
ERR_SETID	6	Set ROCKEY4ND HID wrong
ERR_INVALID_ADDR_OR_SIZE	7	Read/Write address or length is wrong
ERR_UNKNOWN_COMMAND	8	No such command
ERR_NOTBELEVEL3	9	Internal error
ERR_READ	10	Read error
ERR_WRITE	11	Write error
ERR_RANDOM	12	Random number error
ERR_SEED	13	Seed code error
ERR_CALCULATE	14	Calculate error
ERR_NO_OPEN	15	No open dongle before operating dongle
ERR_OPEN_OVERFLOW	16	Too many open dongles (>16)
ERR_NOMORE	17	No more dongle
ERR_NEED_FIND	18	No Find before FindNext
ERR_DECREASE	19	Decrease error
ERR_AR_BADCOMMAND	20	Arithmetic instruction error
ERR_AR_UNKNOWN_OPCODE	21	Arithmetic operator error
ERR_AR_WRONGBEGIN	22	Const number can't use on first arithmetic instruction
ERR_AR_WRONG_END	23	Const number can't use on last arithmetic instruction

ERR_AR_VALUEOVERFLOW	24	Const number > 63
ERR_TOOMUCHTHREAD	25	Too many (>100) threads in the single process open the dongle
ERR_RECEIVE_NULL	0x100	Receive null
ERR_UNKNOWN_SYSTEM	0x102	Unknown operating system

RETURN CODE		DESCRIPTION
-------------	--	-------------

ERR_UNKNOWN	0xffff	Unknown error
-------------	--------	---------------

7.4 Basic Application Examples

FEITIAN offers several program examples to help beginners quickly familiarize themselves with ROCKEY. These sample programs are intentionally simplified to illustrate various security objectives and should not be construed as sufficient for most real world implementations. These samples are for demonstration purposes only. This document is not intended to illustrate how to take full advantage of the ROCKEY software protection system – that will depend on particularities of the developer, the application and the licensing objectives. Section 7.5 Advanced Application Examples are also a good reference but the developer will need to determine the best protection method given his own constraints and objectives.

Some key points that you need to pay attention to when programming:

- 1 Please copy files *rockey4_ND_32.h* to the appropriate directory.
- 2 P3 and P4 are Advanced passwords enabling the developers to write to the dongles. They should not appear in software delivered to end users.
- 3 Be sure that none of the address parameters in the ROCKET4 functions are Null pointers. For example, even if you do not require the Buffer, but it cannot be null, otherwise the result is unpredictable.

The following sample programs are written in VC 6.0. Let us discuss how to perform the required functions step by step from an original program. Software developers who develop software in other languages please do not skip this section. There are no special developing skills for the C language. Most software developers will understand the concepts illustrated here.

Note: All the original codes of sample programs are stored in the *Samples\Beginner* directory on the CD-ROM.

1. Original program –Step 0

This program is the original program before it is protected with ROCKEY4ND.

```
#include <windows.h>
#include <stdio.h>
```

```
void main()
```

```

{
// Anyone begin from here.

printf("Hello FEITIAN!\n");
}

```

2. Find a ROCKEY –Step 1

We add an operation to find the ROCKEY at the beginning of the program. If the dongle is found the program will continue. If it is not found the program will exit.

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h"    // Include Rocky Header File

void main()

{ // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;
    // Rocky Variable
    DWORD lp1, lp2;
    // Rocky Variable
    BYTE buffer[1024];
    // Rocky Variable

    p1 = 0xc44c; // Rocky Demo Password 1
    p2 = 0xc8f8; // Rocky Demo Password 2
    p3 = 0; // Program needn't Password 3, Set to 0
    p4 = 0; // Program needn't Password 4, Set to 0

    // Try to find specified Rocky
    retcode = Rocky(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    printf("Hello FEITIAN!\n");

}

```

It is a very simple security objective. We only need to call the function “Find a ROCKEY dongle”. You may refer to the introduction of the function “Find a ROCKEY dongle” in the section “ROCKEY4ND API Services”.

For testing purposes you might try to run this program with and without the ROCKEY4ND dongle attached to the computer.

3. Open the ROCKEY –Step 2

We add an operation to open ROCKEY with specified passwords at the beginning of the program. If the dongle is opened the program continues. If not the program exits.

```
#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h"    // Include Rockey Header File

void main()

{ //=====

    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rockey Variable
    DWORD lp1, lp2; // Rockey Variable
    BYTE buffer[1024];    // Rockey Variable

    p1 = 0xc44c; // Rockey Demo Password 1
    p2 = 0xc8f8; // Rockey Demo Password 2
    p3 = 0; // Program needn't Password 3, Set to 0
    p4 = 0; // Program needn't Password 4, Set to 0

    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
}
```

```

printf("Hello FEITIAN!\n");
retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{ printf("Error Code: %d\n", retcode);
return;
}
}

```

4. Initialize ROCKEY with Editor or API. Write “Hello FEITIAN!” to the dongle and read it back from the dongle. See Step 3 and Step 4.

Initialize ROCKEY and write “Hello FEITIAN!” to it – Step 3:

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h"    // Include Rockey Header File

void main()
{ // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rockey Variable
    DWORD lp1, lp2; // Rockey Variable
    BYTE buffer[1024];    // Rockey Variable

    p1 = 0xc44c; // Rockey Demo Password 1
    p2 = 0xc8f8; // Rockey Demo Password 2
    p3 = 0; // Program needn't Password 3, Set to 0
    p4 = 0; // Program needn't Password 4, Set to 0

    // Try to find Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {

```

```

printf("Error Code: %d\n",
retcode);
return;
}
p1 = 0; // Pos
p2 = 14; // Length

strcpy((char*)buffer, "Hello FEITIAN! "); retcode = Rockey(RY_WRITE, &handle, &lp1, &lp2, &p1, &p2, &p3,
&p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}
printf("Write: %s\n", buffer);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}

}

```

In Step 3 we have written “Hello FEITIAN!” to the ROCKEY dongle.

In Step 4 we will read the contents of the dongle.

Read dongle contents – Step 4:

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

void main()
{
    // =====

```



```

WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

// Try to find specified Rockeyretcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

p1 = 0; // Pos
p2 = 14; // Length
buffer[14] = 0;
retcode = Rockey(RY_READ, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}

// =====
printf("%s\n", buffer);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
}

```

```

        return;
    }

}

```

5. Generate a true random number with ROCKEY – Step 5

Generate a random number when the program starts and write this random number to the dongle. The program should check if the random number is correct during run-time. If a sharing device is installed to this computer, and someone else runs this program also from another computer, another random number will be generated and written to the dongle. Thus the program on the first computer will be terminated since the random number is not correct.

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h"// Include Rocky Header File


void main()

{ //=====

    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rocky Variable
    DWORD lp1, lp2; // Rocky Variable
    BYTE buffer[1024]; // Rocky Variable


    p1 = 0xc44c; // Rocky Demo Password 1
    p2 = 0xc8f8; // Rocky Demo Password 2
    p3 = 0; // Program needn't Password 3, Set to 0
    p4 = 0; // Program needn't Password 4, Set to 0


    // Try to find specified Rocky
    retcode = Rocky(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }


    retcode = Rocky(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {

```

```

    printf("Error Code: %d\n", retcode);
    return;
}

retcode = Rockey(RY_RANDOM, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Random:%04X\n", p1);


sprintf(buffer, "%04X", p1);
p1 = 0; // Pos
p2 = 4; // Length
retcode = Rockey(RY_WRITE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Write: %s\n", buffer);


p1 = 0; // Pos
p2 = 4; // Length
buffer[4] = 0;
retcode = Rockey(RY_READ, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
printf("Read: %s\n", buffer);
if(buffer)
printf("Hello FEITIAN!\n");
else
exit(0);

```

```

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}
}

```

6. Read the seed code return values with the Editor or API. The seed code calculation is performed inside the dongle and the algorithm is confidential. You may verify the return codes or use the return codes in an encryption routine. See Step 6 and Step 7.

Read the return codes of fixed seed code (0x12345678), Step 6:

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

void main()
{
    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rockey Variable
    DWORD lp1, lp2; // Rockey Variable
    BYTE buffer[1024]; // Rockey Variable

    p1 = 0xc44c; // Rockey Demo Password 1
    p2 = 0xc8f8; // Rockey Demo Password 2
    p3 = 0; // Program needn't Password 3, Set to 0
    p4 = 0; // Program needn't Password 4, Set to 0

    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {

```

```

printf("Error Code: %d\n", retcode);
return;
}

//seed Rockey
lp2 = 0x12345678;
retcode = Rockey(RY_SEED, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}

printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

// Close Rockey
retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}
printf("\n");
getch();
}

```

Verify the return codes of the seed code to see if the program should be terminated, see Step 7:

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

void main()
{
WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

```

```

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
printf("ROCKEY not found!\n");
return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}

//seed Rockey
lp2 = 0x12345678;
retcode = Rockey(RY_SEED, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}

if (p1==0xD03A && p2==0x94D6 && p3==0x96A9 && p4==0x7F54)
printf("Hello FEITIAN!\n");
else
{
printf("Hello error!\n");
return;
}

// Close Rockey
retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}}

```

7. Write the User ID to the dongle with the Editor or API. User ID may be a software version or product type and it may also be used in some encryption schemes. See Step 8 and Step 9.

Note: Advanced passwords are needed for Step 8. Initialize ROCKEY and write User ID to the dongle. See Step 8:

```
#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h"           // Include Rockey Header File

void main()
{
// =====

WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0x0799; // Rockey Demo Password 3
p4 = 0xc43b; // Rockey Demo Password 4

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
printf("ROCKEY not found!\n");
return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}

lp1 = 0x88888888;
retcode = Rockey(RY_WRITE_USERID, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

```

if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}
printf("Write User ID: %08X\n", lp1);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}

}

```

Verify the User ID. If the User ID is not 0x88888888 output “Hello DEMO!”. See Step 9:

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

void main()
{ // =====

WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
printf("ROCKEY not found!\n");
return;
}
}

```



```

retcode = Rockey(RY_OPEN,
&handle,&lp1, &lp2, &p1, &p2,
&p3,&p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}

lp1 = 0;
retcode=Rockey(RY_READ_USERID,&
handle,&lp1,&lp2, &p1, &p2, &p3, &p4,
buffer);

if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}
if (lp1==0x88888888)
printf("Hello FEITIAN!\n");
else
{
printf("Hello DEMO!\n");
return;
}
retcode = Rockey(RY_CLOSE, &handle,
&lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) {
printf("Error Code: %d\n", retcode);
return;
}
}

```

8. Set module value and attributes with Editor or API then check if the module is allowed to be used. Determine whether to activate the associated application module. The module value may also be used by the program. Check if the module is allowed to be decreased to limit the number of software executions. See Step 10, Step 11 and Step 12.

Note: Advanced passwords are needed for Step 10.

Initialize ROCKEY and set module value. For example we set module 0 to be valid and its value cannot be decreased.
See Step 10:

```
#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File


void main()

{ //=====

    WORD retcode;
    WORD handle, p1, p2, p3, p4; // Rockey Variable
    DWORD lp1, lp2; // Rockey Variable
    BYTE buffer[1024]; // Rockey Variable


    p1 = 0xc44c; // Rockey Demo Password 1
    p2 = 0xc8f8; // Rockey Demo Password 2
    p3 = 0x0799; // Rockey Demo Password 3
    p4 = 0xc43b; // Rockey Demo Password 4


    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }


    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }


    p1 = 0; p2 = 3; p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        printf("Error Code: %d\n", retcode);
```

```

    return;
}
printf("Set Module 0: Pass = %04X Decrease no
allow\n",P2);

retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    printf("Error Code: %d\n", retcode);
    return;
}
}

```

If module 0 is valid in the program, output “Hello FEITIAN!”. Otherwise terminate or exit the program. See Step 11:

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

void main()

{ // =====
WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

// Try to find specified Rockey
retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Not found
{
    printf("ROCKEY not found!\n");
    return;
}

retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{

```

```
printf("Error Code: %d\n", retcode);
return;
}
```

```
p1 = 0;
retcode = Rockey(RY_CHECK_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}
```

```
if (p2) printf("Hello FEITIAN!\n"); else
return;
```

```
retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}
}
```

In Step 10 we set p2=3(allowed software run times) and p3=1(Decrement allowed). That is to say module 0(p1=0) sets the maximum software run time to 3. See Step 12:

```
#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File
```

```
void main()
{
// =====
WORD retcode;
WORD handle, p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
```

```

    p2 = 0xc8f8; // Rockey Demo Password 2
    p3 = 0; // Program needn't Password 3, Set to 0
    p4 = 0; // Program needn't Password 4, Set to 0
    // Try to find specified Rockey
    retcode = Rockey(RY_FIND, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Not found
    {
        printf("ROCKEY not found!\n");
        return;
    }

    retcode = Rockey(RY_OPEN, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    p1 = 0;
    retcode = Rockey(RY_CHECK_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {

printf("Error Code: %d\n", retcode); return; }

    if (p2!=1)
    {
        printf("Update Please!\n"); return;
    }

    if(p3==1)
    {
        p1=0;
        retcode = Rockey(RY_DECREASE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if(retcode)
        {
            printf("Error Code: %d\n", retcode);
            return;
        }
    }
}

```

```
printf("Hello FEITIAN!\n");
```

```
retcode = Rockey(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) {
printf("Error Code: %d\n", retcode);
return;
}
}
```

9. Multi ROCKEY dongles with the same passwords may work on the same computer no matter whether the dongle types are the same or not. The program can distinguish the dongles because every dongle has a unique hardware ID. See Step 13:

```
#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

void main()
{
int i, rnum;
WORD retcode;
WORD handle[16], p1, p2, p3, p4; // Rockey Variable
DWORD lp1, lp2; // Rockey Variable
BYTE buffer[1024]; // Rockey Variable

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

// Try to find all Rockey
for (i=0;i<16;i++)
{
if (0 == i)
{
retcode = Rockey(RY_FIND, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE) break;
}
else
{
// Notice : lp1 = Last found hardID
retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE)
break;
}
```

```

}
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}

printf("Found Rockey: %08X ", lp1);
retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}
rynum = i;
// Do our work
for (i=0;i<rynum;i++)
{
// Read Rockey user memory
p1 = 0; // Pos
p2 = 12; // Length
buffer[12] = 0;
retcode = Rockey(RY_READ, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}
printf("%s\n", buffer); // Output

lp1=0;
retcode = Rockey(RY_READ_USERID, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);
return;
}
printf("Read User ID: %08X\n", lp1);

p1=0;
retcode = Rockey(RY_CHECK_MOUDLE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
if (retcode) // Error
{
printf("Error Code: %d\n", retcode);

```

```

return;
}

printf("Check Moudle 0: ");
if (p2) printf("Allow ");
elseprintf("No Allow  ");
if (p3) printf("Allow Decrease\n");
elseprintf("Not Allow Decrease\n");
}

// Close all opened Rockey
for (i=0;i<rynum;i++)
{

retcode = Rockey(RY_CLOSE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
printf("Error Code: %d\n", retcode);
return;
}
}

}

```

A maximum of 16 dongles may be attached to the same computer at the same time. The program can access any dongle you specify.

In the above program we defined a handle array to save the opened ROCKEY handle to prepare for the next operation on the specified dongle. When we find the dongle we open it and we close all opened ROCKEY handles before exiting the program. Developers are better off operating in this manner, but for a large program it is OK to open/close the dongle just once at the beginning/end of the program. Frequent open and close operations will reduce performance. We open the dongle in share mode so that another programs may also simultaneously operate with the dongle.

Note: We called function RY_OPEN and RY_CLOSE in the above program. We must open ROCKEY before all operations with the exceptions of RY_FIND and RY_FIND_NEXT. This is similar to the operation on the disk files. You should close the dongle immediately after finishing dongle related operations.

7.5 Advanced Application Examples

This section is dedicated to providing additional illustrative examples of methods you may employ to protect your software with ROCKEY4ND. These examples are intentionally simplified and not intended to be a complete solutions for software protection. The method appropriate for your application will depend on constraints set by your

licensing agreement and other factors. (If you are familiar with the API call already, you may skip to Chapter 8 ROCKEY4ND Hardware Algorithms.)

1. User Data Zone advanced application.

In Step 14 we will write “Hello FEITIAN!” to User Data Zone (UDZ). In general we would write “Hello FEITIAN!” to the UDZ as one character string, but security may be enhanced by writing it in two parts and then later combining the character strings.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)

{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];  BYTE buf[1024];

    int i, j;
    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0;
    p4 = 0;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    } printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);  if (retcode) {
```

```

ShowERR(retcode);
return;
}

i = 1;
while (retcode == 0)
{
retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE)
break;
if (retcode)
{
ShowERR(retcode);
return;
}

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}

i++;
printf("Find Rock: %08X\n", lp1); } printf("\n");

for (j=0;j<i;j++)
{
p1 = 0;
p2 = 10;
strcpy((char*)buffer, "Hello ");
retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}

printf("Write: Hello \n");
p1 = 12;
p2 = 12;

strcpy((char*)buffer, "FEITIAN!");
retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

```

```

{
    ShowERR(retcode);
    return;
}

printf("Write: FEITIAN!\n");

p1 = 0;
p2 = 10;
memset(buffer, 0, 64); retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,
buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("Read: %s\n", buffer);

p1 = 12;
p2 = 12;
memset(buf, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buf);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Read: %s\n", buf);
printf("\n");
printf("%s\n", strcat(buffer,buf));
retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
getch();
}

}

```

Step 15: You may write a serial number in the User Data Zone (UDZ) and then verify it during run time as a means of protecting and controlling a program module.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
}
```

```

    }

i = 1;
while (retcode == 0)
{

retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE) break;
if (retcode)
{
ShowERR(retcode);
return;
}

retcode = Rockey(RY_OPEN, &handle[i],
&lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

if (retcode)
{
ShowERR(retcode);
return;
}
i++;
printf("Find Rock: %08X\n", lp1); } printf("\n");

for (j=0;j<i;j++)
{
p1 = 0;
p2 = 12;
strcpy((char*)buffer, "a1b2c3d4e5f6");

retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}
printf("Write:a1b2c3d4e5f6\n");

```

```

p1 = 0;
p2 = 2;
memset(buffer, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}
printf("Read: %s\n", buffer);

```

```

if (!strcmp(buffer,"a1"))
printf("Run Module 1\n");
else break;
p1 = 2;
p2 = 2;
memset(buffer, 0, 64);
retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}
printf("Read: %s\n", buffer);

```

```

if (!strcmp(buffer,"b2"))
printf("Run Module 2\n");
else break;

```

```

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}

```

```

printf("\n");
getch();

```

```

    }
}

```

Step 16: Write a number to the UDZ and decrease it during run time as a means of controlling a software module. We recommend you use the encryption idea in Step 12 combined with Step 16.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0)
        return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    int i, j, num;
    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0;
    p4 = 0;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)

```

```

{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    i++;
    printf("Find Rock: %08X\n", lp1); } printf("\n");

for (j=0;j<i;j++)
{
    p1 = 0;
    p2 = 1;
    strcpy((char*)buffer, "3");
    retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write: 3\n");
    p1 = 0;
    p2 = 1;
    memset(buffer, 0, 64);
    retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

```



```

        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        printf("Read: %s\n", buffer);

        num=atoi(buffer);
        if(num)
        {
            printf("HelloFEITIAN!\n");
            num--;
        }
        else
        {
            return;
        }
        p1 = 0;
        p2 = 1;
        sprintf(buffer, "%ld", num);
        retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        printf("Write: %ld\n",num);

        retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        printf("\n");

    }
}

```

```
}
```

2. Seed code advanced applications.

Step 17: You may use different seed codes for different software modules or in different places in the application. Then verify the seed codes in the applications.

```
#include <windows.h>
```

```
#include <stdio.h>
```

```
#include <conio.h>
```

```
#include "../Rockey4_ND_32.h"
```

```
void ShowERR(WORD retcode)
```

```
{  
    if (retcode == 0) return;  
    printf("Error Code: %d\n", retcode);  
}
```

```
void main()
```

```
{
```

```
WORD handle[16], p1, p2, p3, p4, retcode;
```

```
DWORD lp1, lp2;
```

```
BYTE buffer[1024];
```

```
int i, j;
```

```
p1 = 0xc44c;
```

```
p2 = 0xc8f8;
```

```
p3 = 0;
```

```
p4 = 0;
```

```
retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

```
if (retcode)
```

```
{
```

```
ShowERR(retcode);
```

```
return;
```

```
}
```

```
printf("Find Rock: %08X\n", lp1);
```

```
retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

```

if (retcode)
{
ShowERR(retcode);
return;
}

i = 1;
while (retcode == 0)
{

retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE) break;
if (retcode)
{
ShowERR(retcode);
return;
}
retcode = Rockey(RY_OPEN,
&handle[i], &lp1, &lp2, &p1, &p2,
&p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}

i++;
printf("Find Rock: %08X\n", lp1); }
printf("\n");

for (j=0;j<i;j++)
{
lp2 = 0x12345678;
retcode = Rockey(RY_SEED,
&handle[j], &lp1, &lp2, &p1,
&p2, &p3, &p4, buffer); if
(retcode)
{

ShowERR(retcode);
return;
}
}

```

```

    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    if(p1==0xD03A && p2==0x94D6 && p3==0x96A9
        &&p4==0x7F54)
        printf("Hello Fei!\n");
    else
        break;

    lp2 = 0x87654321;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    if(p1==0xB584 && p2==0xD64F && p3==0xC885 && p4==0x5BA0)
        printf("Hello Tian!\n");
    else
        break;

    lp2 = 0x18273645;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

    if(p1==0x2F6D && p2==0x27F8 && p3==0xB3EE && p4==0xBE5A)
        printf("Hello OK!\n");
    else
        break;

    retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("\n");
    getch();

```

```
}  
}
```

In Step 18 we use four outputs of the seed code function to encrypt and decrypt a character string. Be sure you only include the “decrypt” portion of the code in the application version that is sent to end users.

```
#include <windows.h> #include <stdio.h>  
#include <conio.h>  
#include "../Rockey4_ND_32.h"
```

```
void ShowERR(WORD retcode)
```

```
{  
    if (retcode == 0) return;  
    printf("Error Code: %d\n", retcode);  
}
```

```
void main()
```

```
{  
    char str[20] = "Hello FEITIAN!";  
    DWORD mykey = 12345678;  
    int n, slen;  
    WORD handle[16], p1, p2, p3, p4, retcode;  
    DWORD lp1, lp2;  
    BYTE buffer[1024];  
  
    int i,j;  
  
    p1 = 0xc44c;  
    p2 = 0xc8f8;  
    p3 = 0x0799;  
    p4 = 0xc43b;  
  
    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);  
    if (retcode)  
    {  
        ShowERR(retcode);  
        return;  
    }  
    printf("Find Rock: %08X\n", lp1);  
    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);  if (retcode) {
```

```

    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    // Encrypt my data
    slen = strlen(str); lp2 = mykey;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode) // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }

    for (n=0;n<slen;n++) { str[n] = str[n] + (char)p1 + (char)p2 + (char)p3 + (char)p4; }

    printf("Encrypted data is %s\n", str);
}

```

```

// Decrypt my data
lp2 = mykey;
retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) // Error
{
    printf("Error Code: %d\n", retcode);
    return;
}
for (n=0;n<slen;n++) {str[n] = str[n] - (char)p1 - (char)p2 - (char)p3 - (char)p4; }
printf("Decrypted data is %s\n", str);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n");
getch();
}
}

```

3. User ID advanced applications

Step 19: Some developers will write the current date to the UID when initializing the dongles. During runtime the software may compare the current system time with the date stored in the UID. The program would take appropriate actions or continue based on the results of the comparison.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()

```

```

{
WORD
handle[16], p1, p2,
p3, p4, retcode;
DWORD lp1, lp2;
BYTE buffer[1024];
BYTE buf[1024];
int i, j;
SYSTEMTIME st;

p1 = 0xc44c;
p2 = 0xc8f8;
p3 = 0x0799;
p4 = 0xc43b;


retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
|ShowERR(retcode);
return; }
printf("Find Rock: %08X\n", lp1);
retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

if (retcode)
{ ShowERR(retcode); return;
}

i = 1; while (retcode == 0)
{
retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE) break;
if (retcode)
{
ShowERR(retcode);
return;
}

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

```



```

        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        i++;
        printf("Find Rock: %08X\n", lp1); }
        printf("\n");
        for (j=0;j<i;j++)
        {
            lp1 = 0x20021101;
            retcode = Rockey(RY_WRITE_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
            if (retcode)
            {
                ShowERR(retcode);
                return;
            }
            printf("Write User ID: %08X\n", lp1);

            lp1 = 0;
            retcode = Rockey(RY_READ_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
            if (retcode)
            {
                ShowERR(retcode);
                return;
            }
            printf("Read User ID: %08X\n", lp1);

            sprintf(buffer,"%08X",lp1);
            GetLocalTime(&st);
            printf("Date:%04d%02d%02d\n",st.wYear,st.wMonth,st.wDay);

            sprintf(buf,"%04d%02d%02d",st.wYear,st.wMonth,st.wDay);
            if(strcmp(buf,buffer)>=0)
            {
                printf("ok!\n");
            }
            else

```

```

        break;

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

    printf("\n");
    getch();

}
}

```

4. Module advanced applications.

Step 20: Module encryption allows you to selectively control portions of your application with the ROCKEY4ND modules.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;

```

```

p3 = 0x0799;
p4 = 0xc43b;

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1; while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
} printf("\n");

for (j=0;j<i;j++)
{

```

```

p1 = 0;
p2 = 0x2121;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

p1 = 0;
retcode = Rockey(RY_CHECK_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Check Moudle 0: ");
if (p2)
printf("Run Modul 1!\n");
else
break;

printf("\n");

```

```

p1 = 8;
p2 = 0xFFFF;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 8: Pass = %04X Decrease no allow\n", p2);
p1 = 8;
retcode =
Rockey(RY_CHECK_MOUDLE,
&handle[j], &lp1, &lp2, &p1, &p2,
&p3, &p4, buffer);
if (retcode)
{

```

```

ShowERR(retcode);
return;
}
printf("Check Moudle 8: ");
if (p2)
printf("Run Modul 2!");
else break;

```

```

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}
printf("\n");
}
}
}

```

Step 21: This program discussed how to perform multi-module encryption and check the status of the modules. Many applications are segmented into program modules that users may choose or purchase separately. For example, a user may purchase three of four available application modules and the licensing policy would allow the user to execute only those modules that were purchased. ROCKEY4ND modules may be used to enforce this licensing scheme.

```

#include <windows.h>
#include <stdio.h>
#include "../Rockey4_ND_32.h" // Include Rockey Header File

```

```

void main()
{

```

```

int i, j, rnum;
WORD retcode;
DWORD HID[16];
WORD handle[16], p1, p2, p3, p4, retcode;

```

```

p1 = 0xc44c; // Rockey Demo Password 1
p2 = 0xc8f8; // Rockey Demo Password 2
p3 = 0; // Program needn't Password 3, Set to 0
p4 = 0; // Program needn't Password 4, Set to 0

```

```

// Try to find all Rockey
for (i=0;i<16;i++)
{
    if (0 == i)
    {
        retcode = Rockey(RY_FIND, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
    }
    else
    {
        // Notice : lp1 = Last found hardID
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
    }

    if (retcode) // Error
    { printf("Error Code: %d\n", retcode); return;
    }

    printf("Found Rockey: %08X\n", lp1); HID[i] = lp1; // Save HardID retcode =
    Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer); if (retcode)
    // Error {
    printf("Error Code: %d\n", retcode);

    return; } }
    printf("\n");

    rynum = i;

// Do our work
for (i=0;i<rynum;i++)
{
    printf("Rockey %08X module status: ", HID[i]);
    for (j=0;j<16;j++)
    {

        p1 = j; // Module No retcode = Rockey(RY_CHECK_MOUDLE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode) // Error {
            printf("Error Code: %d\n", retcode);
            return;
        }
        if (p2) printf("O");
        else printf("X");
    }
}

```

```

    }
    printf("\n");
}

// Close all opened Rockey for (i=0;i<rynum;i++)
{
    retcode = Rockey(RY_CLOSE, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {

        printf("Error Code: %d\n", retcode);
        return; } } }

```

The above program searches all dongles with the same passwords attached to the computer and displays the status of every module in every listed dongle. “O” means that the module may be used and is not zero; “X” means that the module cannot be used. In a protection scheme that relies on ROCKEY4ND modules this program would help the developer identify modules that are usable from ones that should be terminated.

5. The same code dongle advanced applications.

If you have several software products but only a single purchase code – meaning that the passwords are all the same – you may use the solution indicated below to differentiate the dongles.

In Step 22 the UDZ is used to distinguish the dongles with the same passwords. For example, the dongles with UDZ content of “Ver 10” correspond to software A.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)

{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    WORD handleEnd;
    DWORD lp1, lp2;

```

```

BYTE buffer[1024];

int i, j;

p1 = 0xc44c;
p2 = 0xc8f8;
p3 = 0x0799;
p4 = 0xc43b;


{

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

if (retcode)
{
    ShowERR(retcode);
    return;

}
printf("Find Rock: %08X\n", lp1);


retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode); return;
}


i = 1; while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }


    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {

```



```

        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1); }
    printf("\n");

    for (j=0;j<i;j++)
    {
        /*p1 = 0;

        p2 = 5;
        strcpy((char*)buffer, "Ver10");
        retcode = Rockey(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode) {
            ShowERR(retcode);
            return;
        }
        printf("Write:%s\n",buffer);
        */

        p1 = 0;
        p2 = 5;
        memset(buffer, 0, 64); retcode = Rockey(RY_READ, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode) {
            ShowERR(retcode);
            return;
        }
        printf("Read: %s\n", buffer);

        if (!strcmp(buffer,"Ver10"))
        {
            handleEnd=handle[j];
            break;
        }
    }

}

{ //=====A=====
    retcode = Rockey(RY_RANDOM, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

```

```

if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Random: %04X\n", p1);

lp2 = 0x12345678;
retcode = Rockey(RY_SEED, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode); return;
}
printf("\n");

    }
}

}

```

In Step 23 the UID is used to distinguish the dongles with the same passwords. For example, dongles with UID of “11111111” (hexadecimal) correspond to software A.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

```

```

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;

```

```

        printf("Error Code: %d\n", retcode);

    }

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    WORD handleEnd;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    {
        retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        printf("Find Rock: %08X\n", lp1);

        retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }

        i = 1; while (retcode == 0)
        {
            retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
            if (retcode == ERR_NOMORE) break;
            if (retcode)
            {
                ShowERR(retcode);
                return;
            }
        }
    }
}

```

```

    }

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i++;
printf("Find Rock: %08X\n", lp1);
} printf("\n");

for (j=0;j<i;j++)
{
/*lp1= 0x11111111;
retcode = Rockey(RY_WRITE_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write User ID: %08X\n", lp1);
*/

lp1 = 0;
retcode = Rockey(RY_READ_USERID, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
if(lp1==0x11111111)
{
    handleEnd=handle[j];
    break;
}

}

{ //=====A=====

```

```

p1 = 0;  p2 = 12;
strcpy((char*)buffer, "Hello FEITIAN!");
retcode = Rockey(RY_WRITE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write: %s\n",buffer);


p1 = 0;  p2 = 12;
buffer[512]=0;
retcode = Rockey(RY_READ, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Read: %s\n",buffer);


retcode = Rockey(RY_RANDOM, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Random: %04X\n", p1);


lp2 = 0x12345678;
retcode = Rockey(RY_SEED, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);


retcode = Rockey(RY_CLOSE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);

```

```
        return;  
    }  
  
    printf("\n");  
  
    }  
    }  
  
}
```

Chapter 8 ROCKEY4ND Hardware Algorithms

Developers may define their own algorithms and securely store them inside ROCKEY4ND. The dongle may then be used to calculate a result, and the result used by the application. Since the ROCKEY4ND's User Algorithm Zone (UAZ) is unreadable, even by the manufacturer, this type of software protection is potentially very powerful.

Developers may use either the ROCKEY editor or the RY_WRITE_ARITHMETIC function to write algorithms to the dongle.

8.1 ROCKEY User Defined Algorithm Introduction

8.1.1 Instruction Format

All instructions must be of the form: $\text{reg1} = \text{reg2} \text{ op } \text{reg3}/\text{value}$ reg1 , reg2 and reg3 are registers, value is a figure, op is an operator. For example: $A = A + B$

ROCKEY supports the following operations:

+ Addition

-Subtraction

< Cyclic left shift

* Multiplication

^ XOR

& And

| Or

? Compare *value*

value is a decimal figure between 0 and 63.

Note:

1 "?" operator is for comparison, for example, $C = A ? B$, the results are listed below:

C	A?B	B?A
A<B	0	FFFF
A=B	FFFF	FFFF
A>B	FFFF	0

It will write either "0xFFFF" or "0" to parameter C according to the table above.

First let us have a look at the algorithm example we will write to ROCKEY: $A = A + B$, $B = B + E$, $C = A * F$, $D = B + C$, $H = H ^ H$ A, B, C... are registers in ROCKEY. There are a total of eight 16-bit registers in ROCKEY and they are designed: A, B, C, D, E, F, G and H.

8.1.2 Internal Algorithms & Application Interface

FEITIAN offers 3 calculation functions to call the user-defined algorithms:

RY_CALCULATE1, RY_CALCULATE2, RY_CALCULATE3

These three functions are structurally similar. Data is passed and received by way of the memory addresses p1, p2, p3, and p4.

When passing data to registers:

Register A = p1 Register B =
p2 Register C = p3 Register
D = p4

Register variables vary according to the calculation type: Register E Register F Register G
Register H

When receiving data from registers: p1 =

Register A p2 = Register B p3 =
Register C p4 = Register D

Register A, B, C and D are user interface variables, register E, F, G and H are internal variables.

8.1.3 Differences between the Three Functions

p1, p2, p3 and p4 correspond to registers A, B, C and D in all three calculation functions. These registers are used nearly identically by the three calculation functions. The differences between the functions can be seen by reviewing the results written to registers E, F, G and H.

When a developer's ROCKEY4ND internal program is called, registers A, B, C and D will be populated with data from p1, p2, p3 and p4. The content of registers E, F, G and H will be initialized according to the calculation function in use. See below:

Variable	RY_CALCULATE1
A	P1
B	P2
C	P3
D	P4
E	HiWord of hardware ID
F	LoWord of hardware ID
G	Value stored in module *lp2
H	Random number

Variable	RY_CALCULATE2
A	P1

B	P2
C	P3
D	P4
E	Seed Result 1
F	Seed Result 2
G	Seed Result 3
H	Seed Result 4

Variable	RY_CALCULATE3
A	P1
B	P2
C	P3
D	P4
E	Value in module *lp2
F	Value in module (*lp2 + 1)
G	Value in module (*lp2 + 2)
H	Value in module (*lp2 + 3)

8.1.4 API Interface of the User's Applications

Below is the definition and description of the three calculation functions.

Function	RY_CALCULATE1 (Calculation 1)
Objective	Perform specified calculation
Input parameters	function = RY_CALCULATE1 *handle = ROCKEY's handle *lp1 = Start point of calculation *lp2 = Module number

	*p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4
Return value	A return value = "0" indicates that the function worked correctly. Any other return value indicates an error. When success, *p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4

Note	If the internal algorithm is $A = B + C$, then the call result is $*p1 = *p2 + *p3$. For example: The internal algorithm is $A = A + G$, if $*p1 = 0$, then when returning you may guess the content of module $*p1 = *lp2$. Though you cannot read the content of modules directly, you may determine the content by algorithm. If possible, you had better check the content with an algorithm, not only compare in the program.
------	---

Function	RY_CALCULATE2 (Calculation 2)
Objective	Perform specified calculation
Input parameter	function = RY_CALCULATE2 *handle = ROCKEY4ND's handle *lp1 = Start point of calculation *lp2 = Seed Code *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4
Return value	A return value = "0" indicates that the function worked correctly. Any other return value indicates an error. When success, *p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4
Note	When performing calculation 2, the initial values of register E, F, G and H are the return values of seed code *lp2, to make it simple, ROCKEY calls function RY_SEED with seed code *lp2, and writes the return values to register E, F, G and H for next operation.

Function	RY_CALCULATE3 (Calculation 3)
Objective	Perform specified calculation
Input	function = RY_CALCULATE3

parameter	*handle = ROCKEY4ND's handle *lp1 = Start point of calculation *lp2 = Module number *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4
-----------	---

Return value	A return value = "0" indicates that the function worked correctly. Any other return value indicates an error. When success, *p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4
Note	When performing calculation 3, the initial values of register E, F, G and H are the content of module *lp2 and *lp2+1/2/3, for example: When calls calculation 3 with *lp2 = 0, the initial values of register E, F, G and H are: E = Content of module 0 F = Content of module 1 G = Content of module 2 F = Content of module 3 <i>Note: The address will return to "0" when the module address call exceeds 15. For example: Calculation 3 with *lp2 = 14, the initial values of register E, F, G and H are: E = Content of module 14 F = Content of module 15 G = Content of module 0 H = Content of module 1</i>

8.2 Writing User Defined Algorithms into ROCKEY

8.2.1 Writing Algorithm

Developers may use the *RY_WRITE_ARITHMETIC* algorithm to write algorithms to the ROCKEY4ND User Algorithm Zone (UAZ). The ROCKEY editor is another option for writing algorithms to the UAZ.

Function	RY_WRITE_ARITHMETIC (Write algorithm)
Objective	Write user defined algorithm to ROCKEY
Input	function = RY_WRITE_ARITHMETIC
parameter	*handle = ROCKEY4ND's handle *p1 = Start point of calculation *buffer = Instruction string
Return	A return value = "0" indicates that the function worked correctly. Any other return value indicates an error.

For example: strcpy(buffer, "A=A+E, A=A+F, A=A+G, A=A+H"); p1 = 3; retcode = Rockey(RY_WRITE_ARITHMETIC, handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

The "buffer" is the place for you to temporarily store the algorithm. One instruction is separated

from another by a ",". ROCKEY will automatically assign the first instruction in the algorithm, "Start" and the last instruction, "End". Taking this program as an example:

Address 3 in Algorithm Zone: $A=A+E$ Address 4 in Algorithm Zone: $A=A+F$ Address 5 in Algorithm Zone: $A=A+G$ Address 6 in Algorithm Zone: $A=A+H$

Then 3 is the starting point of the algorithm in the User Algorithm Zone (UAZ). 6 is the end point. ROCKEY will return to the user application after performing the instruction in address 6. The users must call the program in the dongles from the starting point of the algorithm. Otherwise the results are 4 random numbers.

8.2.2 Instruction Conventions

There are some conventions when developers write algorithm instructions: $A = A + B$ Valid instruction $D = D \wedge D$ Valid instruction $A = B$ Invalid instruction, $A = B | B$ would be correct. $A = 0$ Invalid instruction, $A = A \wedge A$ would be correct $C = 3 * B$ Invalid instruction, $C = B * 3$ would be correct $D = 3 + 4$ Invalid instruction, there can not be two constants $A = A / B$ Invalid instruction, ROCKEY does not support division operator $H = E * 200$ Invalid instruction, constant must be less than 64 $A = A * 63$ If it is the first or last instruction it is an invalid instruction, otherwise valid.

8.3 User Defined Algorithm Examples

8.3.1 Basic Algorithm Application Examples

Calculation 1 example

First we write the algorithm (We only need to write the algorithm once. The code used to write the algorithm(s) to the dongle does not appear in the application delivered to the end user.)

```
p1 = 0;strcpy(buffer, "H=H^H, A=A*23, F=B*17, A=A+F, A=A+G, A=A<C, A=A^D, B=B^B, C=C^C,D=D^D");
retcode = Rockey(RY_WRITE_ARITHMETIC, handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

Then call this algorithm from the program: $lp1 = 0$; // Start point of calculation $lp2 = 7$; // Module number
 $p1 = 5$; // Initial value of A $p2 = 3$; // Initial value of B $p3 = 1$; // Initial value of C $p4 = 0xffff$; // Initial value of D
 $retcode = Rockey(RY_CALCULATE1, handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);$

The command begins to execute from instruction 0 ($lp1$) of the UAZ and the registers are initialized as follows: $A = 5$ ($p1$) $B = 3$ ($p2$) $C = 1$ ($p3$) $D = 0xffff$ ($p4$) E = the upper 16-bit of HID F = the lower 16-bit of HID G = the value in module #7 ($lp2$) H = random number (16-bit) Assuming that the value in module 7 is 0x2121, the result of this calculation will be: $((5*23 + 3*17 + 0x2121) < 1) \wedge 0xffff = 0xbc71$

Calculation 1 example codes – Step 24:

```
#include <windows.h>
```

```

#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)

{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    int i, j;
    char cmd[] = "H=H^H, A=A*23, F=B*17, A=A+F, A=A+G, A=A<C, A=A^D, B=B^B, C=C^C,

D=D^D"; p1 = 0xc44c; p2
=0xc8f8; p3 = 0x0799; p4 =
0xc43b;

retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;

```

```

while (retcode == 0)
{

retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer); if (retcode ==
ERR_NOMORE) break;
if (retcode)
{
ShowERR(retcode);
return;
}

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}
i++;
printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{ /* p1 = 7;
p2 = 0x2121;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}
printf("Set Moudle 7: Pass = %04X Decrease no allow\n", p2);
printf("\n");
*/
p1 = 0;
strcpy((char*)buffer, cmd); retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,
&p4,buffer);
if (retcode)
{
ShowERR(retcode); return;
}
}

```

```

printf("Write Arithmetic 1\n");

lp1 = 0;    lp2 = 7;
p1 = 5;    p2 = 3;    p3 = 1;    p4 = 0xffff;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=5, p2=3, p3=1, p4=0xffff\n");

printf("\n");
printf("Result = ((5*23 + 3*17 + 0x2121) < 1) ^ 0xffff = 0xBC71\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n");
getch();
}
}

```

Calculation 2 example

In Step 25 we write algorithm ("A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H") to the UAZ, and the calculation result is 0x7b17.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)

```

```

    {
        if (retcode == 0) return;
        printf("Error Code: %d\n", retcode);
    }

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd1[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer); if (retcode ==
        ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);

```



```

return;
}

```

```

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
i++;
printf("Find Rock: %08X\n", lp1);
}
printf("\n");

```

```

for (j=0;j<i;j++)
{
    /*lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);
    printf("\n");

```

```

*/
p1 = 10;
strcpy((char*)buffer, cmd1);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 2\n");

```

```

lp1 = 10;
lp2 = 0x12345678;
p1 = 1;
p2 = 2;

```

```

p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE2, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode) {

    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("\n");
printf("Result =d03a + 94d6 + 96a9 + 7f54 + 1 + 2 + 3 + 4=0x7b17\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("\n"); getch(); } }

```

Calculation 3 example

In Step 26 we write algorithm ("A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H") to UAZ, and the calculation result is 0x14.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()

```

```

{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd2[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i = 1;
    while (retcode == 0)
    {
        retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE) break;
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {

```

```

        ShowERR(retcode);
        return;
    }

    i++;

    printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    /* p1 = 0;;
    p2 = 1;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,  buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

    p1 = 1;
    p2 = 2;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 1: Pass = %04X Decrease no allow\n", p2);
    p1 = 2;    p2 = 3;    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 2: Pass = %04X Decrease no allow\n", p2);

```

```

    p1 = 3;    p2 = 4;    p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 3: Pass = %04X Decrease no allow\n", p2);
printf("\n");
*/

p1 = 17;
strcpy((char*)buffer, cmd2);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 3\n");

lp1 = 17;    lp2 = 0;    p1 = 1;    p2 = 2;    p3 = 3;    p4 = 4;
retcode = Rockey(RY_CALCULATE3, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("\n");
printf("Result = 1+2+3+4+1+2+3+4=0x14\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);

retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

```

```
printf("\n"); getch(); } }
```

8.3.2 Complex Algorithm Application Examples

Complex example 1

In Step 27 we first search the dongle and get its hardware ID. Then we use the calculation 1 function in the program to get the hardware ID again. Compare the two hardware IDs. If they are different the program will be terminated.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD findlp1, truelp1;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;

    char cmd[] = "A=E|E,B=F|F,C=G|G,D=H|H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
    }
    return;
}
```

```

}

printf("Find Rock: %08X\n", lp1);
findlp1=lp1;

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i = 1;
while (retcode == 0)
{

    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;
    printf("Find Rock: %08X\n", lp1);
}
printf("\n");
for (j=0;j<i;j++)
{
    /*  p1 = 7;    p2 = 0x2121;    p3 = 0;

    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

```

```

if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 7: Pass = %04X Decrease no allow\n", p2);
p1 = 0;
strcpy((char*)buffer, cmd);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 1\n");

    */

lp1 = 0; lp2 = 7; p1 = 1; p2 = 2; p3 = 3; p4 = 4;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);


printf("\n");
printf("Moudle 7 : %x\n", p3);
truelp1=MAKELONG(p2,p1);

printf("truelp1 : %x\n",truelp1);
if (findlp1==truelp1)

    printf("Hello FEITIAN!\n");    else
    break;


retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

```



```

    }

    printf("\n");
    getch();
}
}

```

Complex example 2

In Step 28 we get the return codes of a seed code with the calculation 2 function. Then we compare these return codes with the return codes we get with the same seed code at the beginning of the program. If they are different the program will be terminated.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)

{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];
    WORD rc[4];

    int i, j;

    char cmd1[] = "A=E|E,B=F|F,C=G|G,D=H|H";

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rocky(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

```

```

if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Find Rock: %08X\n", lp1);


retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode); return;
}

i = 1; while (retcode == 0)
{
    retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    i++;

    printf("Find Rock: %08X\n", lp1); }
printf("\n");
for (j=0;j<i;j++)
{
    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {

```

```

        ShowERR(retcode);
    return;
}

printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);


rc[0] = p1; rc[1] = p2;
rc[2] = p3; rc[3] = p4;
p1 = 0;
strcpy((char*)buffer, cmd1);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 2\n");


lp1 = 0;
lp2 = 0x12345678;
p1 = 1;
p2 = 2;
p3 = 3;
p4 = 4;
retcode = Rockey(RY_CALCULATE2, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);


printf("\n");
if(rc[0]==p1 && rc[1]==p2 && rc[2]==p3 && rc[3]==p4)
printf("Hello FEITIAN!\n");
else    break;
retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

```

```
printf("\n"); getch(); } }
```

Complex example 3

In Step 29 we get the values stored in the 64 modules by using the calculation 3 function. Remember that the modules may not be read, even with the Advanced passwords. You may write some important data to the modules or perform some other operations.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD
    handle[16], p1, p2,
    p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE
    buffer[1024];

    int i, j;

    char cmd2[] = "A=E|E,B=F|F,C=G|G,D=H|H";
    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
```

```

ShowERR(retcode);
return;
}
printf("Find Rock: %08X\n", lp1);

retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode); return;
}

i = 1; while (retcode == 0)
{
retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE) break;
if (retcode)
{
ShowERR(retcode);
return;
}

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
ShowERR(retcode);
return;
}

i++;
printf("Find Rock: %08X\n", lp1); } printf("\n");

for (j=0;j<i;j++)
{
/*
p1 = 0;
p2 = 1;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

if (retcode)

```

```

{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

p1 = 1;
p2 = 2;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 1: Pass = %04X Decrease no allow\n", p2);

p1 = 2;
p2 = 3;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Set Moudle 2: Pass = %04X Decrease no allow\n", p2);

p1 = 3;    p2 = 4;
p3 = 0;
retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

printf("Set Moudle 3: Pass = %04X Decrease no allow\n", p2);
// :    */

p1 = 0;
strcpy((char*)buffer, cmd2);

```

```

    retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Write Arithmetic 3\n");

    lp1 = 0;   lp2 = 0;   p1 = 0;   p2 = 0;   p3 = 0;   p4 = 0;
    retcode = Rockey(RY_CALCULATE3, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Calculate Input: p1=0, p2=0, p3=0, p4=0\n");

    printf("\n");
    printf("Moudle 0: %x\n",p1);
    printf("Moudle 1: %x\n",p2);
    printf("Moudle 2: %x\n",p3);
    printf("Moudle 3: %x\n",p4);
    retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

    printf("\n"); getch(); } }

```

Complex example 4

In Step 30 we use all the three calculation functions and we write 4 calculation sections to the ROCKEY dongle. The results of the three calculations are used for additional calculations. Of course you may let ROCKEY perform much more complex calculations according to your situation.

```

#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "../Rockey4_ND_32.h"

```

```

void ShowERR(WORD retcode)
{
    if (retcode == 0) return;
    printf("Error Code: %d\n", retcode);
}

void main()
{
    WORD handle[16], p1, p2, p3, p4, retcode;
    DWORD lp1, lp2;
    BYTE buffer[1024];

    int i, j;
    int t1,t2,t3;

    char cmd[] = "H=H^H, A=A*23, F=B*17, A=A+F, A=A+G, A=A<C, A=A^D, B=B^B, C=C^C, D=D^D";
    |char cmd1[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";
    char cmd2[] = "A=A+B, A=A+C, A=A+D, A=A+E, A=A+F, A=A+G, A=A+H";
    char cmd3[] = "H=H^H,A=A|A, B=B|B, C=C|C,D=A+B,D=D+C";
    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0x0799;
    p4 = 0xc43b;

    retcode = Rockey(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Find Rock: %08X\n", lp1);

    retcode = Rockey(RY_OPEN, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode); return;
    }

    i = 1; while (retcode == 0)
    {

```



```

retcode = Rockey(RY_FIND_NEXT, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode == ERR_NOMORE) break;
if (retcode)
{
    ShowERR(retcode);
    return;
}

retcode = Rockey(RY_OPEN, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

i++;

printf("Find Rock: %08X\n", lp1);
}
printf("\n");

for (j=0;j<i;j++)
{
    p1 = 7;    p2 = 0x2121;    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 7: Pass = %04X Decrease no allow\n", p2);
    printf("\n");

    lp2 = 0x12345678;
    retcode = Rockey(RY_SEED, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
    }
}

```

```

        return;
    }
    printf("Seed: %04X %04X %04X %04X\n", p1, p2, p3, p4);
    printf("\n");
    p1 = 0;    p2 = 1;    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 0: Pass = %04X Decrease no allow\n", p2);

```

```

    p1 = 1;    p2 = 2;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 1: Pass = %04X Decrease no allow\n", p2);

```

```

    p1 = 2;
    p2 = 3;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Set Moudle 2: Pass = %04X Decrease no allow\n", p2);

```

```

    p1 = 3;
    p2 = 4;
    p3 = 0;
    retcode = Rockey(RY_SET_MOUDLE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

```

```

}
printf("Set Moudle 3: Pass = %04X Decrease no allow\n", p2);
printf("\n");

p1 = 0;
strcpy((char*)buffer, cmd);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 1\n");

lp1 = 0;   lp2 = 7;   p1 = 5;   p2 = 3;   p3 = 1;   p4 = 0xffff;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=5, p2=3, p3=1, p4=0xffff\n");

printf("Result = ((5*23 + 3*17 + 0x2121) < 1) ^ 0xffff = 0xBC71\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);
t1=p1;

p1 = 10;
strcpy((char*)buffer, cmd1);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 2\n");

```

```

lp1 = 10;
lp2 = 0x12345678;
p1 = 1;    p2 = 2;    p3 = 3;

p4 = 4;
retcode = Rockey(RY_CALCULATE2, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("Result =d03a + 94d6 + 96a9 + 7f54 + 1 + 2 + 3 + 4=0x7b17\n");
printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);
t2=p1;

p1 = 17;
strcpy((char*)buffer, cmd2);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic 3\n");

lp1 = 17;    lp2 = 0;    p1 = 1;    p2 = 2;    p3 = 3;    p4 = 4;
retcode = Rockey(RY_CALCULATE3, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Calculate Input: p1=1, p2=2, p3=3, p4=4\n");

printf("Result = 1+2+3+4+1+2+3+4=0x14\n");

```

```

printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4); t3=p1;

printf("\n");    p1 = 24;
strcpy((char*)buffer, cmd3);
retcode = Rockey(RY_WRITE_ARITHMETIC, &handle[j], &lp1, &lp2, &p1, &p2, &p3,&p4, buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("Write Arithmetic \n");

```

```

lp1 = 24;    lp2 = 7;    p1 = t1;    p2 = t2;    p3 = t3;    p4 = 0;
retcode = Rockey(RY_CALCULATE1, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4,buffer);
if (retcode)
{
    ShowERR(retcode);
    return;
}

```

```

printf("Calculate Output: p1=%x, p2=%x, p3=%x, p4=%x\n", p1, p2, p3, p4);
retcode = Rockey(RY_CLOSE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);    if (retcode)
{
    ShowERR(retcode);
    return;
}
printf("\n");
getch();
}
}

```

8.3.3 Advanced Algorithm Application Examples

In Step 31 we will write the core algorithms or codes of the application to the ROCKEY dongle. Below are three programs: the original program, the ROCKEY initializing program and the final program for the end users.

The original program:

```

#include "stdafx.h"
#include "DrawCircle.h"

```

```

#include "DrawCircleDoc.h"
#include "DrawCircleView.h"
#include "DrawParamDlg.h"
#include "DrawMethodDlg.h"

```

```

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

```

```

void CDrawCircleView::DrawCircleMidPoint(CDC *pDC, int iCenterX, int iCenterY, int r)

```

```

{
    int x=0;  int y=r;  int p=1-r;
    TRACE("Origin\n");
    CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);
    m_lpCircleBuf[0].x = x; m_lpCircleBuf[0].y = y; m_nPointCount=1;
    while(x<y)
    {
        x++;
        if(p<0)
        {
            p+=2*x+1;
        }
        else
        {
            y--;
            p+=2*(x-y)+1; }
        TRACE("%d,(%d,%d);",p,x,y);
        CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);
        m_lpCircleBuf[m_nPointCount].x = x;
        m_lpCircleBuf[m_nPointCount].y = y; m_nPointCount++;
    }
    TRACE("\n");

```

```

}

```

Initialize ROCKEY:

```

#include "stdafx.h"
#include <windows.h>

```

```

#include "..\inc\rockey4_ND_32.h"

void ReportErr(WORD wCode)
{ printf("ERROR:%d\n",wCode); }

int main(int argc, char* argv[])
{
    WORD p1=0xc44c,p2=0xc8f8,p3=0x0799,p4=0xc43b;
    DWORD lp1,lp2;
    WORD handle[16];
    BYTE buffer[1024];
    BYTE cmdstr[] = "B=B|B,B=B+1,B=B*2,B=B+1,A=A+B,C=C-1,C=C*2,B=A-C";
    WORD   retcode;

    retcode = Rockey(RY_FIND,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
    if(retcode)
    {

        ReportErr(retcode);
        return 0;

    }
    printf("Find successfully\n");

    retcode = Rockey(RY_OPEN,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);
    if(retcode)
    {
        ReportErr(retcode);
        return 0;
    }
    printf("Open successfully\n");

    p1 = 10;
    retcode = Rockey(RY_WRITE_ARITHMETIC,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,cmdstr);
    if(retcode)
    {
        ReportErr(retcode);
        return 0;
    }
    printf("Write arithmetic successfully\n");

    retcode = Rockey(RY_CLOSE,&handle[0],&lp1,&lp2,&p1,&p2,&p3,&p4,buffer);

```

```
return 0; }
```

The final program for the end users:

```
#include "stdafx.h"
#include "DrawCircle.h"

#include "DrawCircleDoc.h"
#include "DrawCircleView.h"
#include "DrawParamDlg.h"
#include "DrawMethodDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

WORD p1=0xc44c,p2=0xc8f8,p3=0x0799,p4=0xc43b;
DWORD lp1,lp2; WORD handle[16];
BYTE buffer[1024];

void CDrawCircleView::DrawCircleMidPoint_Rockey(CDC *pDC, int iCenterX, int iCenterY, int r)
{
    int x=0; int y=r; int p=1-r; int seed=0;;
    short p1,p2,p3,p4;;
    CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);
    TRACE("Hardware\n");
    m_lpCircleBuf[0].x = x; m_lpCircleBuf[0].y = y; m_nPointCount=1;
    while(x<y)
    {
        p1 = p;   p2 = x;   p3 = y;   p4 = seed;

        if(!RunRockey((WORD&)p1,(WORD&)p2,(WORD&)p3,(WORD&)p4))
        {
            AfxMessageBox("Error during run time");
            break;
        }

        if(p<0)
        {

```



```

        p = p1;
    }
    else
    {
        p = p2;
        y--;
    }
    x++;
    TRACE("%d,(%d,%d);",p,x,y);
    CirclePlotPoints(pDC,iCenterX,iCenterY,x,y);
    m_lpCircleBuf[m_nPointCount].x = x;
    m_lpCircleBuf[m_nPointCount].y = y; m_nPointCount++;
} TRACE("\n"); }

BOOL CDrawCircleView::RunRockey(WORD &A, WORD &B, WORD
&C, WORD &D)
{
    WORD retcode;
    lp1 = 10;
    retcode =
    Rocky(RY_CALCULATE1,&handle[0],&lp1,&lp2,&A,&B,&C,&D,buffer);
    if(retcode)
    {
        return FALSE;
    }else return TRUE;
}

```

8.4 Note

ROCKEY4ND has as many as 128 instructions. Developers do not need to consider the start and end attributes of an algorithm. ROCKEY will automatically assign a Start/End attribute to the instructions. In practice this means that if the developer writes a two-instruction algorithm to the User Algorithm Zone (UAZ), and then a three instruction algorithm, the result will not be a single five instruction algorithm. Algorithms that begin with “Null” or “E” will produce unpredictable results.

8.5 Tips

- 1 **Make randomized calls to the ROCKEY API** - Randomly scatter calls to the ROCKEY API from within your application. Calls made to the API from time-to-time will make it very difficult to mimic the behavior of the protection method or hack the application.
- 2 **Use dynamic information with the seed code function** -The use of dynamic information with the seed code function, such as system date, makes the protection method stronger because the results can change with the input and calculation.
- 3 **Do not repeatedly use the same protection method in your application** -If you use the same protection


method several times in your application it will be easier for the cracker to find the rule and crack your application. Protection methods that are complex and rely on a number of different checks and calculations are the most difficult to crack.

4. **Encrypt the character string and data** – In “Step 18” of this document we showed an encryption method using information stored inside the dongle. Encrypting a character string in the manner described is a strong method because a failure to properly decrypt the string can cause the application to terminate or take other actions in accordance with the licensing agreement.

5. **Use API encryption and Envelope encryption together** – The strongest protection method will have the developer first using a complex and dynamic implementation of the ROCKEY API, and then protecting this new file with the ROCKEY Envelope.

Please keep the end user environment in mind when you design the software protection solution. You should flexibly adopt the methods suggested here within the limitations and objectives of your environment and licensing policy.

ROCKY4ND Technical Specification

Picture	
Dimensions	51 x 16 x 7 mm
Color	Blue, Green, Gray, Brown
Memory	1k
Min. Operating Voltage	5V
Current Consumption (active/idle)	<= 50 mA
Max. Write Cycle	>100,000
Connector	USB Type A
Operating Temp.	0-70oC
Data Retention	10 Years
Onboard Algorithm	128 User-defined simple algorithms
Hardware ID	Globally Unique
Driver	Driverless
Remoter Update	High-level Secure Remoter Update, Multiple-Schemes
Production Tool	Batch Production Tool
Module Management	Multiple-Module and Management Schemes
Operating Systems	Windows 98SE/ME/2000/XP/ Server 2003, Linux, Mac OS
Enveloper	Yes
Invisible Module Zone	64 modules to control software release schemes
Random Generator	Hardware Random Generator
Abundant APIs	VB, VC, Delphi, FoxPro, Java, PB, VS.Net...