

FEITIAN



Rockey4ND Developer's Guide

V1.1

Feitian Technologies Co., Ltd.

Website: www.FTSafe.com

Revision History:

Date	Revision	Description

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 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.
3. Warranty – Feitian warrants that the hardware 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.

5. 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.
6. 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.

Contents

Rockey4ND	1
User's Guide	1
Chapter 1. ROCKEY4ND APIs	1
1.1 ROCKEY4ND Function Prototype and Definition	1
1.2 ROCKEY4ND API Services.....	3
1.3 Error Codes	8
1.4 Basic Application Examples	9
1.5 Advanced Application Examples.....	30
Chapter 2. ROCKEY4ND Hardware Algorithms	62
2.1 ROCKEY User Defined Algorithm Introduction	62
2.2 User Defined Algorithm Examples.....	67
2.3 Tips.....	101

Chapter 1. ROCKEY4ND APIs

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.

1.1 ROCKEY4ND Function Prototype and Definition

```
WORD Rocky
(
    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 = Rocky(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

Note: All parameters must be defined in the program. Do not pass a Null pointer. Otherwise, an error will occur.

- “function” is a 16-bit number. It stands for the specific function and it is defined below:

Parameter Name	Parameter Type	Parameter Meaning
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

- “handle” is the pointer for ROCKEY operation's handle.
- “lp1” and “lp2” are the pointers for long integer parameters. Their content depends on the functions.

- “p1”, “p2”, “p3” and “p4” are the pointers for short integer parameters. Their content depends on the functions.
- “buffer” is the pointer for character buffer. Its content depends on the functions.

1.2 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.

1.2.1 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.

1.2.2 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.

1.2.3 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

1.2.4 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.

1.2.5 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.

1.2.6 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.

1.2.7 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.

1.2.8 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

1.2.9 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.

1.2.10 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.

1.2.11 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".

1.2.12 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).

1.2.13 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.

1.2.14 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.

1.2.15 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.

1.2.16 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.

1.2.17. 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".

1.3 Error Codes

ERR_SUCCESS	0	//No error
ERR_NO_ROCKEY	3	//No ROCKEY4ND found
ERR_INVALID_PASSWORD	4	//ROCKEY4ND found, with incorrect basic password
ERR_INVALID_PASSWORD_OR_ID	5	//Incorrect password or hardware ID
ERR_SETID	6	//Error setting hardware ID
ERR_INVALID_ADDR_OR_SIZE	7	//Incorrect target address or length
ERR_UNKNOWN_COMMAND	8	//Unknown command
ERR_NOTBELEVEL3	9	//Internal error
ERR_READ	10	//Error reading data
ERR_WRITE	11	//Error writing data
ERR_RANDOM	12	//Random number error
ERR_SEED	13	//Seed error
ERR_CALCULATE	14	//Calculation error
ERR_NO_OPEN	15	//Dongle not opened before operation
ERR_OPEN_OVERFLOW	16	//Too many dongles opened (>16)
ERR_NOMORE	17	//No more dongle(s) can be found
ERR_NEED_FIND	18	//Use FindNext without Find
ERR_DECREASE	19	//Decrement error
ERR_AR_BADCOMMAND	20	//Algorithm command error
ERR_AR_UNKNOWN_OPCODE	21	//Algorithm operator error
ERR_AR_WRONGBEGIN	22	//The first command of algorithm contains a constant
ERR_AR_WRONG_END	23	//The last command of algorithm contains a

		constant
ERR_AR_VALUEOVERFLOW	24	//The value of the constant of the algorithm is greater than 63
ERR_TOOMUCHTHREAD	25	//The number of the threads that open a dongle in a process is greater than 100
ERR_RECEIVE_NULL	0x100	//Cannot receive
ERR_UNKNOWN_SYSTEM	0x102	//Unknown operating system
ERR_UNKNOWN	0xffff	//Unknown error

1.4 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 1.5 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:

- P3 and P4 are Advanced passwords enabling the developers to write to the dongles. Each of them should be set to a valid value only in this case. They should be set to 0 in the software delivered to end users.
- Be sure that none of the address parameters in the ROCKEY4ND 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.

1.4.1 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");

}
```

1.4.2 Find Dongle – 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 ROCKEY4ND Header File

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

    p1 = 0xc44c;
    p2 = 0xc8f8;
    p3 = 0;          // Program needn't Password3, Set to 0
    p4 = 0;          // Program needn't Password4, Set to 0

    // Try to find specified ROCKEY4ND
    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.

1.4.3 Open Dongle – Step 2

We add an operation to open ROCKEY with specified passwords at the beginning of the program. If the dongle can be opened the program continues. Otherwise, the program exits.

```
#include <windows.h>

#include <stdio.h>
#include "Rockey4_ND_32.h"          // Include ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];              // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, 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;
}

    // =====

    printf("Hello FeiTian!\n");

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

```
}
```

1.4.4 User Memory – Steps 3 and 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 ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];             // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, Set to 0

    // Try to find 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;
    }
}
```



```

    p1 = 0; // Pos
    p2 = 14; // Length

    strcpy((char*)buffer, "Hello FeiTian! ");
    retcode = Rocky(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 = Rocky(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 ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4; // ROCKEY4ND Variable
    DWORD lp1, lp2; // ROCKEY4ND Variable
    BYTE buffer[1024]; // ROCKEY4ND Variable

    p1 = 0xc44c; // ROCKEY4ND Demo Password1
    p2 = 0xc8f8; // ROCKEY4ND Demo Password2
    p3 = 0; // Program needn't Password3, Set to 0
    p4 = 0; // Program needn't Password4, 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;
}

p1 = 0; // Pos
p2 = 14; // Length

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

1.4.5 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 ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];              // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, 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 = Rocky(RY_RANDOM, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)    // Error
    {
        printf("Error Code: %d\n", retcode);
        return;
    }
}
```

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

sprintf(buffer, "%04X", p1);
p1 = 0; // Pos
p2 = 4; // Length
p3 = 1;
    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;
    }
}
```

```
}
```

1.4.6 Seed – Steps 6 and 7

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 ROCKEY4ND Header File

void main()
{
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];              // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, 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 - **Step 7:**

```

#include <windows.h>

#include <stdio.h>
#include "Rockey4_ND_32.h"    // Include ROCKEY4ND Header File

void main()
{
    WORD retcode;

    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];             // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, Set to 0

    // Try to find specified Rockey

```

```
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;
}

//seed Rocky
lp2 = 0x12345678;
retcode = Rocky(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 Rocky
retcode = Rocky(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);

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

1.4.7 User ID – Steps 8 and 9

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 ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];             // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0x0799;    // ROCKEY4ND Demo Password3
    p4 = 0xc43b;    // ROCKEY4ND Demo Password4

    // Try to find specified Rockey
    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;
    }
}
```



```

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 ROCKEY4ND Header File

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

    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];              // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2

    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, 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;
}

lp1 = 0;
retcode = Rocky(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 = Rocky(RY_CLOSE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)

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

}
```

1.4.8 Module – Steps 10, 11, and 12

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 ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable
    BYTE buffer[1024];             // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0x0799;    // ROCKEY4ND Demo Password3
    p4 = 0xc43b;    // ROCKEY4ND Demo Password4

    // Try to find specified Rockey
    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;
    }

    p1 = 0;
    p2 = 3;
    p3 = 0;
    retcode = Rocky(RY_SET_MOUDLE, &handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
```

```

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

    printf("Set Moudle 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 ROCKEY4ND Header File

void main()
{
    // =====
    WORD retcode;
    WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable
    DWORD lp1, lp2;                // ROCKEY4ND Variable

    BYTE buffer[1024];              // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1
    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, 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 ROCKEY4ND Header File

void main()
{
```

```
// =====  
WORD retcode;  
WORD handle, p1, p2, p3, p4;    // ROCKEY4ND Variable  
DWORD lp1, lp2;                // ROCKEY4ND Variable  
BYTE buffer[1024];             // ROCKEY4ND Variable  
  
p1 = 0xc44c;    // ROCKEY4ND Demo Password1  
p2 = 0xc8f8;    // ROCKEY4ND Demo Password2  
p3 = 0;         // Program needn't Password3, Set to 0  
p4 = 0;         // Program needn't Password4, 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;
}
}

```

1.4.9 Collaboration – Step 13

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 ROCKEY4ND Header File


void main()
{
    int i, rnum;
    WORD retcode;
    WORD handle[16], p1, p2, p3, p4; // ROCKEY4ND Variable
    DWORD lp1, lp2;                  // ROCKEY4ND Variable
    BYTE buffer[1024];               // ROCKEY4ND Variable

```

```
p1 = 0xc44c;    // ROCKEY4ND Demo Password1
p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
p3 = 0;         // Program needn't Password3, Set to 0
p4 = 0;         // Program needn't Password4, Set to 0

// Try to find all Rocky
for (i=0;i<16;i++)
{
    if (0 == i)
    {
        retcode = Rocky(RY_FIND, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode == ERR_NOMORE)
            break;
    }
    else
    {
        // Notice : lp1 = Last found hardID
        retcode = Rocky(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 Rocky: %08X ", lp1);

    retcode = Rocky(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++)
{
    // Read Rocky user memory
    p1 = 0; // Pos
    p2 = 12; // Length

    buffer[12] = 0;
    retcode = Rocky(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 = Rocky(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 = Rocky(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 ");
    else
printf("No Allow ");
    if (p3)
```

```
printf("Allow Decrease\n");
    else
printf("Not Allow Decrease\n");

}

// Close all opened Rocky
for (i=0;i<rynum;i++)
{
    retcode = Rocky(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 other programs may also simultaneously operate with the dongle.

You can find the source code of the samples above from the CD-ROM or under *Samples* directory.

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.

1.5 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.)

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

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

    retcode = Rocky(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;
p3 = 1;
        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;
p3 = 1;

strcpy((char*)buffer, "FeiTian!");
retcode = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
if (retcode)
```

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

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

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

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

```
retcode = Rocky(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;
p3 = 1;
    strcpy((char*)buffer, "a1b2c3d4e5f6");

retcode = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(RY_CLOSE, &handle[jj], &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 = 2;
p3 = 1;

        strcpy((char*)buffer, "03");
        retcode = Rocky(RY_WRITE, &handle[j], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);

            return;
        }
        printf("Write: 03\n");
        p1 = 0;
        p2 = 1;
        memset(buffer, 0, 64);
        retcode = Rocky(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("Hello FeiTian!\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");
    }
}
}

```

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

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

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

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

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

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

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

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

}
```

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

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

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

    retcode = Rocky(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();

}

}
```

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

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

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

    return;
}

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

    retcode = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(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 ROCKEY4ND Header File

void main()
{
    int i, j, rynum;
    WORD retcode;
    DWORD HID[16];

    WORD handle[16], p1, p2, p3, p4; // ROCKEY4ND Variable
    DWORD lp1, lp2;                  // ROCKEY4ND Variable
    BYTE buffer[1024];               // ROCKEY4ND Variable

    p1 = 0xc44c;    // ROCKEY4ND Demo Password1

    p2 = 0xc8f8;    // ROCKEY4ND Demo Password2
    p3 = 0;         // Program needn't Password3, Set to 0
    p4 = 0;         // Program needn't Password4, Set to 0
```



```
// Try to find all Rocky
for (i=0;i<16;i++)
{
if (0 == i)
{
retcode = Rocky(RY_FIND, &handle[i], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode == ERR_NOMORE) break;
    }
    else
    {
        // Notice : lp1 = Last found hardID
        retcode = Rocky(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 Rocky: %08X\n", lp1);
    HID[i] = lp1;    // Save HardID
retcode = Rocky(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("Rocky %08X module status: ", HID[i]);
for (j=0;j<16;j++)
{
p1 = j;    // Module No
```

```
retcode = Rocky(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 Rocky

for (i=0;i<rynum;i++)
{
    retcode = Rocky(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.

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

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

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

        {

            ShowERR(retcode);
            return;
        }

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

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

        retcode = Rocky(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;
p3 = 1;
        strcpy((char*)buffer, "Ver10");
        retcode = Rocky(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 = Rocky(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 = Rocky(RY_RANDOM, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }
    printf("Random: %04X\n", p1);

    lp2 = 0x12345678;
    retcode = Rocky(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 = Rocky(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 = Rocky(RY_FIND, &handle[0], &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
            ShowERR(retcode);
            return;
        }
        printf("Find Rock: %08X\n", lp1);

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

```
        ShowERR(retcode);
        return;
    }

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

            return;
        }

        retcode = Rocky(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 = Rocky(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 = Rocky(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;
        p3 = 1;
        strcpy((char*)buffer, "Hello Feitian!");
        retcode = Rocky(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 = Rocky(RY_READ, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
                ShowERR(retcode);
                return;
        }
        printf("Read: %s\n",buffer);

        retcode = Rocky(RY_RANDOM, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
        if (retcode)
        {
                ShowERR(retcode);
```



```
        return;
    }
    printf("Random: %04X\n", p1);

    lp2 = 0x12345678;
    retcode = Rocky(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 = Rocky(RY_CLOSE, &handleEnd, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
    if (retcode)
    {
        ShowERR(retcode);
        return;
    }

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

Chapter 2. 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.

2.1 ROCKEY User Defined Algorithm Introduction

2.1.1 Instruction Format

All instructions must be in the following format:

```
reg1 = reg2 op reg3/value
```

reg1, *reg2* and *reg3* are registers. *value* is a figure. *op* is an operator. For example:

```
A = A + B
```

ROCKEY supports the following operators: +, -, <, *, ^, &, |, and ?.

value is a decimal number between 0 and 63.

Note: "?" operator is used for comparison.

For example, the results of "C = A ? B" are listed below:

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

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

First let us have a look at the algorithm example we will write to the dongle:

```
A= A+B, B = B + E, C = A * F, D = B + C, H = H ^ H
```

A, B, C... are registers in the dongle. There are a total of eight 16-bit registers in the dongle and they are designed: A, B, C, D, E, F, G and H.

2.1.2 Internal Algorithms & Application Interface

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

RY_CALCULATE1, RY_CALCULATE2, and 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*, and *Register H*.

When receiving data from registers:

p1 = Register A

p2 = Register B

p3 = Register C

p4 = Register D

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

2.1.3 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:

Internal Variable	RY_CALCULATE1
A	P1
B	P2
C	P3
D	P4
E	High 16 bits of hardware ID
F	Low 16 bits of hardware ID
G	Value stored in module *lp2
H	Random number

Internal Variable	RY_CALCULATE2
A	P1
B	P2
C	P3
D	P4
E	Seed return value 1 (depending on the seed code in *lp2)
F	Seed return value 2 (depending on the seed code in *lp2)
G	Seed return value 3 (depending on the seed code in *lp2)
H	Seed return value 4 (depending on the seed code in *lp2)

Internal Variable	RY_CALCULATE3
A	P1
B	P2
C	P3
D	P4

E	Content of the module specified by *lp2
F	Content of the module of *lp2 + 1
G	Content of the module of *lp2 + 2
H	Content of the module of *lp2 + 3

2.1.4 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 = Handle of the dongle *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 value of 0 indicates that the function works properly. Any other value indicates an error. When the function is executed successfully; *p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4
Notes	For example, if the algorithm in the dongle is: $A = B + C$ Then, the result by calling calculation 1 is: $*p1 = *p2 + *p3$ For example, if the algorithm in the dongle is: $A = A + G$ If $*p1 = 0$ when inputting, then you may figure out the content of the module specified by $*p1 = *lp2$ when returning, although you cannot read the content of the module directly. If possible, you'd better verify the content of the module with an algorithm, instead of comparing in the program.

Function	RY_CALCULATE2 (Calculation 2)
Objective	Perform specified calculation
Input parameters	function = RY_CALCULATE2 *handle = Handle of the dongle *lp1 = Start point of calculation

	<p>*lp2 = Seed code *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4</p>
Return value	<p>A value of 0 indicates that the function works properly. Any other value indicates an error.</p> <p>When the function is executed successfully;</p> <p>*p1 = Return value 1 *p2 = Return value 2 *p3 = Return value3 *p4 = Return value 4</p>
Notes	<p>When the dongle is calling the algorithm by calculation 2, the initial values of registers E, F, G, and H are the seed return value of *lp2. In other words, the dongle uses *lp2 as the seed code and calls RY_SEED function, and puts the return values into registers E, F, G, and H respectively for processing by the user.</p>

Function	RY_CALCULATE3 (Calculation 3)
Objective	Perform specified calculation
Input parameters	<p>function = RY_CALCULATE3 *handle = Handle of the dongle *lp1 = Start point of calculation *lp2 = Start address of module *p1 = Input value 1 *p2 = Input value 2 *p3 = Input value 3 *p4 = Input value 4</p>
Return value	<p>A value of 0 indicates that the function works properly. Any other value indicates an error.</p> <p>When the function is executed successfully;</p> <p>*p1 = Return value 1 *p2 = Return value 2 *p3 = Return value 3 *p4 = Return value 4</p>
Notes	<p>When the dongle is calling the algorithm by calculation 3, the initial values of registers E, F, G, and H are the contents of the successive 4 modules starting at *lp2. For example: the initial values of registers E, F, G, and H are as follows when calling calculation 3 with *lp2 = 0: E = Content of module 0 F = Content of module 1 G = Content of module 2 F = Content of module 3 If the address of the module is greater than 63, the address will be wrapped. For example, when calling calculation 3 with *lp2 = 62, the initial values of registers E, F, G, and H are as follows: E = Content of module 62</p>

	F = Content of module 63 G = Content of module 0 H = Content of module 1
--	--

2.1.5 Writing Self-defined Algorithms to Dongle

2.1.5.1 Writing Algorithms

Aside from the use of ROCKEY4ND Editor, developers can also use the interface RY_WRITE_ARITHMETIC to develop a program for writing algorithms themselves.

Function	RY_WRITE_ARITHMETIC (Write algorithm)
Objective	Write algorithms as defined by developers
Input parameters	function = RY_WRITE_ARITHMETIC *handle = Handle of the dongle *p1 = Start point of the calculation *buffer= String of algorithm instructions
Return value	A value of 0 indicates that the function works properly. Any other value indicates an error.

For example:

```
strcpy(buffer , "A=A+E , A=A+F , A=A+G , A=A+H");

p1 = 3;

retcode= Rocky(RY_WRITE_ARITHMETIC , handle , &lp1 , &lp2 , &p1 , &p2 , &p3 , &p4 , buffer);
```

You can see that the algorithm to be written is placed into *buffer* and the instructions are separated by commas. The first instruction will be set to the start of the algorithm and the last instruction will be set to the end of the algorithm automatically. For example:

Address 3 of algorithm area: A=A+E

Address 4 of algorithm area: A=A+F

Address 5 of algorithm area: A=A+G

Address 6 of algorithm area: A=A+H

“3” is the start point of the algorithm in the dongle; and “6” is the end point of the algorithm in the dongle. After executing the instruction at address 6, the program will go to the user part. The calling of the program in the dongle must begin from the start point of the algorithm. If the calling point is not the start point of the algorithm, 4 random numbers will be returned.

2.1.5.2 Restrictions on Algorithm Instruction

There are some restrictions on algorithm instruction. They are described below with some example instructions:

A = A + B Valid instruction

D = D ^ D Valid instruction

A = B Invalid instruction, must be in the format of algorithm, such as A = B | B

A = 0 Invalid instruction, must be in the format of algorithm, such as A = A ^ A

C = 3 * B Invalid instruction, the constant must be postfix, for example, C = B * 3

$D = 3 + 4$	Invalid instruction, only 1 constant is allowed in an instruction
$A = A / B$	Invalid instruction, the division operator is not supported
$H = E * 200$	Invalid instruction, the constant must be lower than 64
$A = A * 63$	Valid or invalid instruction, constants are not allowed in the first and last instructions

2.2 User Defined Algorithm Examples

2.2.1 Basic Algorithm Application Examples

2.2.1.1 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 = Rocky(RY_WRITE_ARITHMETIC, handle, &lp1, &lp2, &p1, &p2, &p3, &p4, buffer);
```

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

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

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



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

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

printf("\n");
```

```
        getch();
    }

}
```

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

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

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

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

        retcode = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(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();
    }

}

```

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

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

if (retcode)

{
    ShowERR(retcode);
    return;
}

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

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

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

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

2.2.2 Complex Algorithm Application Examples

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

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

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

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

}
```

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

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

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

        retcode = Rocky(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 = Rocky(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 = Rocky(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 = Rocky(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();
    }
}
```

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

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

    return;
}

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

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

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

```

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

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

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

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

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

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

```

2.2.3 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 dongle:

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

    printf("Find successfully\n");

    retcode = Rocky(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 arithmetirc successfully\n");

    retcode = Rocky(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("Runtime error");  
            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;
}
```

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.

2.3 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.

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.