Thank you for purchasing Software Dog dedicated to the software protection produced by Rainbow China Co., Ltd.

Please read the following *License Agreement* before using this product.

By obtaining the license of this product, you accept all terms and conditions and any supplementary or special terms hereof in this Agreement. In case you do not agree to any of the terms, please return this product within three days to Rainbow China Co., Ltd. or its distributors. Use of this software indicates the acceptance of the obligations derived from the terms and conditions of this Agreement.

License Agreement

1. Right of use is authorized. You may duplicate the software in this CD for back-up purpose, and you may incorporate the software provided to your own programs in accordance with the documentation of this product for the purpose of protecting your own software.

2. In addition to the authorization as specified in Clause 1, you should agree not to duplicate, modify, reverse engineer, disassemble or re-assemble all or part of this product and not to sell, rent, permit, transfer and distribute all or part of this product or the right granted by this Agreement.

3. We guarantee that this product, if used properly, will not have substantial material or design defects within twelve months since the date when this product is delivered to you. All the obligations of Rainbow China Co., Ltd. and all the remedial measures you will obtain are to send back the product or have the defected product replaced or repaired. Beyond the12 months all the remedial measures you will obtain are to exchange the defected product or have it repaired.

4. No warranty of any kind is granted to anyone in addition to the aforementioned limited warranty provided to the original buyer of this product. The product, performance and service of Rainbow China, Co., Ltd. are provided “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

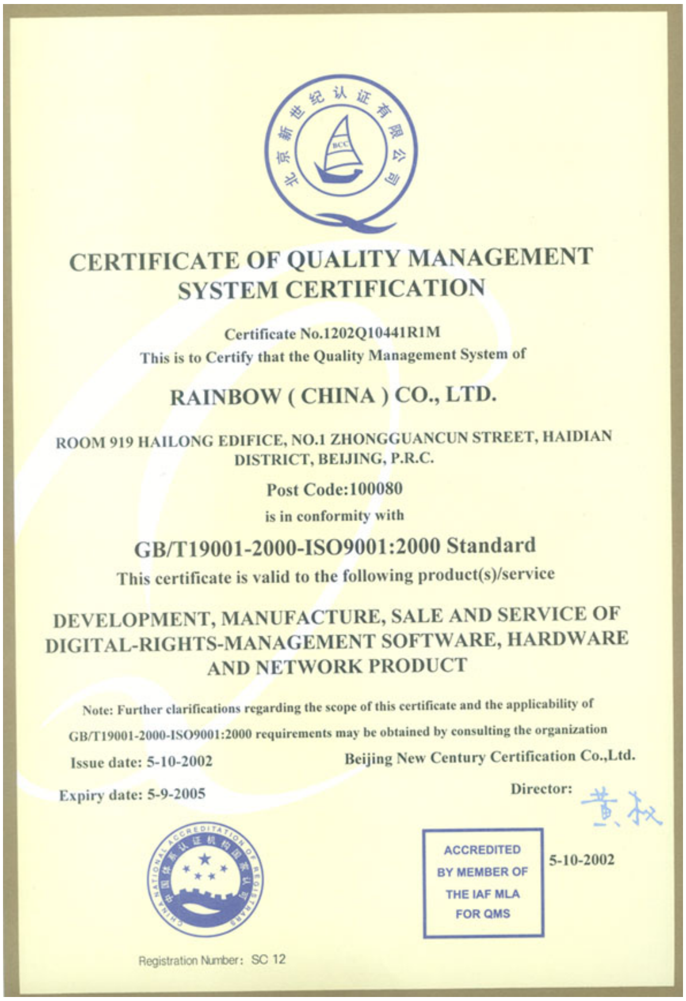
5. In no event will Rainbow China, Co., Ltd. or its distributors be liable for any damage or responsibility including special, incidental, collateral, consequential or indirect damages such as lost data, lost profits that you may obtain by the usage of the product or the disabled function of this product.

6. Ownership and copyright of all products, including Software Dog, software, documentation and all other materials accompanied with this product as well as the back-ups you have made are reserved by Rainbow China Co., Ltd.

7. Authorization of this Agreement automatically terminates when any action of infringement occurs.

ISO9001 Quality System Certification

MicroDog SDK is developed and manufactured by Rainbow China Co., Ltd. Rainbow China Co., Ltd. has obtained the ISO9001 Certification, the globally recognized standard for quality system, by Beijing New Century Certification Center of Quality System. The development and manufacturing process is in full compliance with ISO standard.



北京新世纪认证有限公司 BCC

CERTIFICATE OF QUALITY MANAGEMENT SYSTEM CERTIFICATION

Certificate No.1202Q10441R1M

This is to Certify that the Quality Management System of

RAINBOW (CHINA) CO., LTD.

ROOM 919 HAILONG EDIFICE, NO.1 ZHONGGUANCUN STREET, HAIDIAN DISTRICT, BEIJING, P.R.C.

Post Code:100080

is in conformity with

GB/T19001-2000-ISO9001:2000 Standard

This certificate is valid to the following product(s)/service

DEVELOPMENT, MANUFACTURE, SALE AND SERVICE OF DIGITAL-RIGHTS-MANAGEMENT SOFTWARE, HARDWARE AND NETWORK PRODUCT

Note: Further clarifications regarding the scope of this certificate and the applicability of GB/T19001-2000-ISO9001:2000 requirements may be obtained by consulting the organization

Issue date: 5-10-2002

Beijing New Century Certification Co.,Ltd.

Expiry date: 5-9-2005

Director: 黄权

CHINA NATIONAL ACCREDITATION OF REGISTRARS 中国质量认证机构国家认可

ACCREDITED BY MEMBER OF THE IAF MLA FOR QMS

5-10-2002

Registration Number: SC 12

Contents

Company Introduction

Rainbow China Co., Ltd. is a subsidiary of Rainbow Technologies, Inc. in Beijing, P. R. China. Founded in 1984, Rainbow Technologies, Inc. (NASDAQ Code "RNBO") is a worldwide leading provider of information security solutions for the Internet, eCommerce, mCommerce, content protection, software distribution, and high-assurance communications. Rainbow applies its core technology to a variety of Internet applications from securing software to the acceleration of secure communication for eCommerce and Virtual Private Networks (VPNs). Rainbow's products include: secure Web server and VPN acceleration boards; anti-piracy and Internet software licensing and distribution solutions; PKI-based security solutions; voice, data and satellite security systems; and USB-based authentication tokens. The company is headquartered in Irvine, California, and has offices throughout the United States, United Kingdom, France, Germany, Australia, China, India, The Netherlands, Brazil, Mexico, Taiwan, Hong Kong and Korea. A network of more than 80 authorized distributors sells Rainbow products worldwide.

Rainbow Technologies is comprised of the corporate groups – Rainbow eSecurity Group and Rainbow Mykotronx. As a subsidiary company, Rainbow China Co., Ltd owns the intellectual property of "Software Dog" completely. With the mission of "providing the best product of DRM in the world" and business guideline of "building up core competency of quick response" and marketing strategy of "Centralizing on customers, satisfying customers and product tells Rainbow", Rainbow China has established a complete sales, management and operating system. Our business scope has expanded into four main areas: software protection, information security, digital content protection and online software distribution. In 2000 Rainbow China got ISO 9001 Certified. Its products got certificates for FCC and CE certification. In 2001 it was awarded "Strongly Recommended Excellent Software Product Prize" issued by China Software Association. The one millionth Software Dog was produced in 2001. On behalf of a new-generation software protection product, SmartDog was released in Apr.2002. Our persistent aim is helping customers seize every possible opportunity, maximize their investment return and seize every opportunity in business with our excellent products.

Product

On the basis of excellent brand of Software Dog, Dog family retains its technologies advanced and quality outstanding. With the new introduction of SmartDog, Dog family consists of SmartDog, MicroDog, USB Dog, UMH SDK, SoftDog, PowerDog, TimeDog, CF Dog, NetDog and UNH SDK.

Common Features

- Hardware Dog cannot be duplicated;
- EEPROM memory for developer use;
- One and above serial numbers for each customer;
- The Shell protection provides protection for EXE/COM/DLL files, which can be executed only with presence of the Hardware Dog;
- More than 60 compilers are supported with OBJ interfaces and source codes of examples are provided;
- Multi-module and remote upgrade are supported.;
- Time bar is built-in the Hardware Dog, which enhances anti-tracing. Tracing may lead to unpredicted result.
- In full compliance with IEEE-1284, USB1.1, ISO-7816, PC/SC standard and compatible with all peripherals.

Technologies

- **License management:** Flexible license management and distribution processing, the software developers may control the execution period of software, times to be started and the usage of different modules for each customer. At the same time, it’s easy and fast to upgrade the license.
- **Virtual Machine:** The protected piece of code is executed in a virtual machine

based on Smartcard chip. The code is encrypted with the software.

- **DERNT** (Data Exchange Random Noise Technology)**:** The proprietary technology disables all logical analyzers and various capturing kits. It makes secured transmission and obstructs all potential hackers.

- **Maze Technology:** Many jumping instructions that are included at the points of entry and exit in programs dynamically change the order of executing applications to improve the anti-tracing ability.

- **Protection Algorithm:** The Software developer may select his own algorithms by defining the bytes for algorithm and algorithm descriptors flexibly. Hardware Dogs contain a 200-bytes nonvolatile memory. You can use Developer’s tool or API functions to program the memory. The bytes for defining descriptor are from 196th to 198th and that of algorithm is 199th. So there are 256 different algorithm could be selected, algorithmic descriptor has 24-bits, the variations could be more than 4000 mega by changing both the bytes for algorithm and algorithmic descriptor.

Outstanding Manufacturing Facilities

- COB is employed and the IC chip couldn’t be replaced.

- Ultra-sonic welding machine enables the plastic cover firm and reliable.

- With the integrated design and reasonable structure, the USB port can endure as many as ten thousand times of plug in and out operations.

- Customized logo and mark are acceptable to be printed on the Dog cover.

- A variety of colors of cover are available.

Software Dog Family

- **SmartDog** is the high secure solution enabled by secure Smartcard chip. Part of software code is executed only within the chip. Besides, SmartDog delivers license management. SmartDog is an USB device.

- **MicroDog** is a flexible, programmable and intelligent software protection solution for PC applications. It contains a microprocessor and a 200-byte nonvolatile memory and supports cascade. It adopts AS technology, and can be widely used in PC applications.

- **SoftDog** is the most cost-effective protection solution for PC applications. It adopts DRNT technology and contains a 100-byte nonvolatile memory.

- **TimeDog** is a software protection product that can fully control demo software's expiration date and define the running time of software products. It has a built-in clock and 5K-byte memory.

- **NetDog** is an intelligent software protection solution for NOVELL and WINDOWS network applications. It contains built-in 200-byte nonvolatile memory and supports TCP/IP, IPX and NETBIOS protocols. It can penetrate routers, switches and hubs and define up to 500 concurrent users.

- **CFDog** is the most secure, flexible, programmable software protection solution for palmtop applications. It follows CompactFlash standard and can be plugged in Palm-size CF card slot.

- **NetDog SDK** provides a combination of software protection for application with both parallel port and USB port drivers in network environment. It avoids repetition of installation and can switch between different types of ports easily and flexibly.

Performance Comparative Analysis

Key Functions	**MicroDog**	**SoftDog**	**NetDog**	**PowerDog**	**TimeDog**	**CF Dog**
Built-in 8-bit CPU	Yes		Yes	Yes	Yes	
Built-in Timer	Yes		Yes	Yes	Yes	

Clock chip			Yes		Yes	
Supports Linux	Yes	Yes	Yes	Yes	Yes	
Cascade Hardware Dog with same SN	Yes		No	Yes	Yes	
Supports low voltage on parallel port (2.5V)	Yes	Yes	Yes	Yes	Yes	Follow CF standard
Supports Win9X/NT/2K/XP	Yes	Yes	Yes	Yes	Yes	WinCE
Encrypts EXE DLL files directly	Yes	Yes	Yes	Yes	Yes	
Store the algorithm of user's program in the hardware				Yes		
Developer-defined protection algorithm descriptor	Yes		Yes		Yes	
Provide EEPROM	Yes	Yes	Yes	Yes	Yes	Yes
Data Exchange Random Noise Technology	Yes	Yes	Yes	Yes	Yes	
Maze Technology	Yes	Yes	Yes	Yes	Yes	
Extended Memory					Yes	
Unduplicated Hardware	Yes	Yes	Yes	Yes	Yes	Yes
Anti-sharing by hardware	Yes	Realizable with program	Yes	Yes	Yes	

Limitless software to be protected	Yes	Yes	Yes	Yes	Yes	Yes
Compatible with printers	Good	Good	Good	Good	Good	
Unique manufacture code	Yes		Yes	Yes	Yes	Yes
Type of port	Parallel/USB	Parallel/ USB	Parallel/ USB	Parallel	Parallel	CF
Multiple modules and Remote upgrade	Yes	Yes	Yes	Yes	Yes	Yes
Free upgrade Downloadable module	Yes	Yes	Yes	Yes	Yes	Yes

Quality

Commitment

- Hardware failure rate is lower than 0.1%
- One year guarantee for defective products.

Certification

- ISO 9001 Certified
- FCC and CE Certification
- Strongly Recommended Excellent Software Product

Services

Tech Support and Service

- Strictly-trained professional tech-supports provide quick response;
- Free upgrade software;
- Provide source code with customized case;
- Instant delivery;
- Hotline for complaints in Quality Assurance Department;

Market

As a long-standing partner with many renowned companies, we have provided software protection solutions for many customers throughout dozens of countries in the world as well as more than 6,000 domestic customers. The enterprises adopting Software Dog are active in various fields of software, such as CAD, finance, MIS, education, medicine, multi-media software, etc.

Distribution

“Software Dog” has an easy- accessible distribution networks around the domestic market. Customers can know our products from branches and agents in Beijing, Shanghai, Guangzhou, Shenzhen, Wuhan, Haerbin, Hangzhou, Suzhou, Changsha, Nanjing, Chengdu, Chongqing, Zhengzhou, Xi’an. We hope that our Software Dog will become your friends and protect your legal benefits.

Contact us

Rainbow China Co., Ltd.
Address: Room 1603, JinYu Plaza, No.100 Xisanhuan North Road, Haidian District, Beijing, China, 100037

Telephone: 86-10-8851-9191
Fax: 86-10-6872-7342
Toll-free Hotline: 800-810-0804
Complaint Line: 86-10-6872-7343
Website: http://www.isecurity.com.cn/english/
Email: softDog@cn.rainbow.com

About This Manual

This manual is MicroDog SDK V3.2 Developer’s Manual. The following passages will make you familiar with this manual as well as the usage of MicroDog SDK in a shorter time.

MicroDog SDK includes one Parallel Dog, one USB Dog (UMC type), one USB extension cable, one Installation CD (all protection tools and Utility Tools), one Developer’s Manual, one envelope enclosing Dog Key, one business card and quality certificate.

Hardware Dog includes Parallel Dog and USB Dog.

The “Glossary” will help you better understand this Developer’s Manual.

“Introduction” briefly introduces the principle, main functions and features of the MicroDog SDK.

To install Parallel Dog, please power off your computer first. Insert Parallel Dog into the parallel port and screw it securely and tightly. You can connect the parallel port data cable to Parallel if you use peripherals. You need to reinstall drivers if you used the old versions of MicroDog before.

Please install USB Dog driver if you choose to use USB Dog. Insert USB Dog into the USB port as instructed. USB is a plug and play device, so you do not need to reboot your computer. You can insert USB Dog to either port on your computer or the USB extension cable.

Place the Installation CD in the driver, and the installation program will run automatically. Please complete installation as instructed according to the interfaces. For more information, please refer to “Installing MicroDog SDK”.

Use DogEdt32.EXE to edit data on Hardware Dog. You can read data from or write data into the memory, and change the password. The directory is

\UTILITY\DogEdt32.EXE. For more information, please refer to "Using Developer's tool".

Use Win16 and Win32 Shell to protect EXE files without source code. Win32 Shell is located in \WIN32\EXETOOL\ WIN32SHL.EXE and Win 16 Shell is located in \WIN3X\EXETOOL\WIN16EXE.EXE. For more information, please refer to "Using Shell Protection".

Use API functions to set hidden Software Lock in the source code. You can use this protection mode to protect special programs such as CPL or OCX. For more information, please refer to the "Using API calls".

Use DLL to protect multiple modules and limit their using times. For more information, please refer to "Using Multiple Modules DLL".

More protection strategies, simple or complicated, will be introduced in "Making Optimal Use of MicroDog". You can protect your software at different levels with our API functions.

Please provide end users the MicroDog driver when you release your software. The driver is located in \DRIVER\ MicroDogInstdrv.exe. You can either offer it to the end users directly, or by calling RCMicroDogSetup.dll located in \DRIVER to install the driver. For more information, please refer to "Distributing Your Software".

Legend

Warning 💣 Notes 👓

Glossary

MicroDog SDK

MicroDog SDK includes three types of SDK: Parallel Dog SDK, USB Dog SDK, and Parallel Dog/USB Dog Suite.

What does the abbreviation MicroDog stand for?

MicroDog stands for a combination of USB Dog and Parallel Dog. Two kinds of hardware forms share the same middleware. MicroDog should be SDK. When customers order the product, they should clearly claim what kind of hardware they want to buy, USB Dog or Parallel Dog.

Developer

It refers to the people who read this manual and program for protecting their software.

Hardware Dogs

MicroDog Hardware Dogs are referring to one Parallel Dog and one USB Dog. In this document we refer to the Parallel Dog and USB Dog when we mention Hardware Dogs.

Dog Key

Dog Key is enclosed in the envelope along with the SDK. You must enter Dog Key to bind the software with Hardware Dogs if you install this product for the first time. Please use DogEdt32.EXE to convert Dog Key to Number.Dog. **Please take good care of this file and keep it confidential**.

Serial Number

A string of digit printed on the shell of the Parallel Dog and USB Dog, such as **Serial No. xxxxxxx**. The MicroDog SDK has the only Serial NO.

Hardware Dogs with the Same Serial Number

The Hardware Dogs with the same serial number, for example the Parallel Dog and USB Dog in the same MicroDog SDK. After purchasing the MicroDog SDK the developers can buy the Hardware Dogs corresponding to the SDK (the Hardware Dogs with the same serial number) from Rainbow China or our agency.

Bind

During the installation of the MicroDog SDK the installation will bind the encryption modules with the corresponding Hardware Dogs. The bound module can only operate the Hardware Dogs with the corresponding serial number. Without being bound the modules can’t be used.

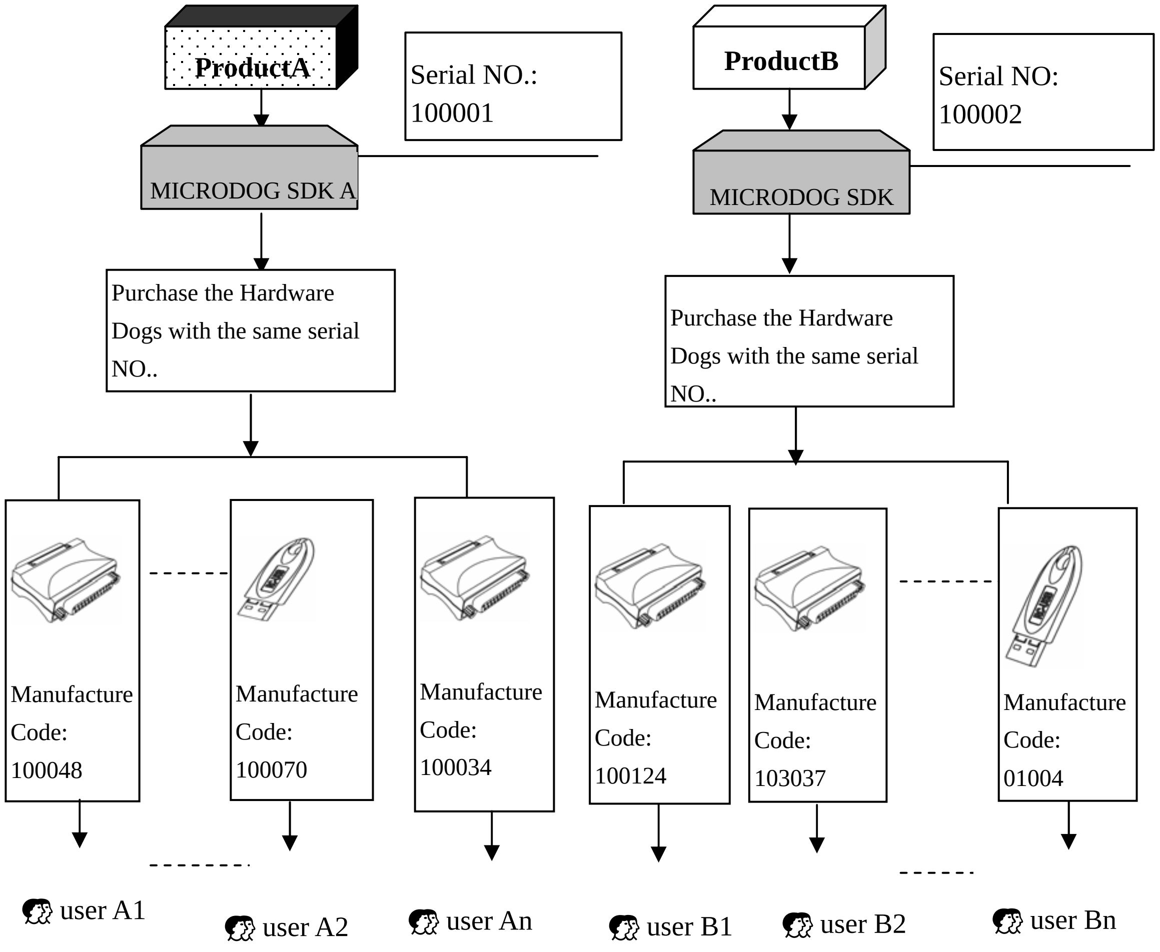
Cascade

Cascade is referring to the series-wound Parallel Dog. The MicroDog supports the cascade of the hardware with different serial number.

Manufacture Code

Every Hardware Dog has a unique Manufacture Code, which is sequential in the manufacture process. Even Hardware Dogs with the same Serial Number have different Manufacture Code. The Manufacture Code is for customers’ management. You can get Manufacture Code by the developer tool (DogEdt32.EXE) or interface function of the Dog.

It is highly recommended to use the Dog with the different serial number to protect different series of the software product avoiding the conflict occurred in the release. it is illustrated as the following figure.



Number.Dog

One MicroDog SDK has only one Serial Number that printed on the envelope enclosing Dog Key. Number.Dog contains the Serial Number and all other necessary information for ordering Hardware Dogs with the same Serial Number use DogEdt32.EXE to convert Dog Key to Number.Dog. The developer should provide Number.Dog when they purchase the Hardware Dogs with the same serial number.

Please keep Dog Key confidential.

Memory

Hardware Dogs contain a 200-bytes nonvolatile memory for developers to save important data and configure parameters. You can use Developer's tool or API functions to program the memory. Generally, user's cells are from 0 to 195; descriptors cells are from 196 to 198 and algorithm cell is 199. If the method of the multiple modules is used the user's cells are from 0 to 127.

The developer can read, write memory by the developer tool (DogEdt32.EXE) or the interface functions, and meanwhile set the algorithm descriptors. The different descriptors can produce different result of the conversion.

Password

The default password is 0. The password should be a decimal number ranging from 0 to 4, 294, 967, 295. You can use DogEdt32.EXE to change the password. of Hardware Dogs. You can't do any operations of accessing the Hardware Dogs without the correct password. Password has nothing to do with the intensity of the encryption. Please take good care of this file and keep it confidential.

Conversion (DogConvert())

There is a function we call it 'conversion'. It is like challenging/response. The conversion is one-way. It converts a string of characters to a 4-bytes result. There is an algorithm for the conversion. Different customers hope having different results for a same input string. We design it that the developer could get different result by setting their own algorithm and algorithm descriptor (refer to the 'Chapter 3 Using Developer's Tool' for more information about algorithm and algorithm descriptor). To do the conversion, the function 'DogConvert' could be used.

New Functions

Updates from MicroDog SDK3.3 to MicroDog SDK3.4

MicroDog SDK 3.4 adopts the Improved Security Engine II we developed independently, which integrates the latest cryptology and the advanced algorithm, it will promote the anti-cracked capability of software, and the stability will be boosted up at the same time.

The detailed updated context as follows:

- UMC Hardware Dog has passed the Microsoft WHQL certification.
- Updating all Win32/Linux modules and it will not support the old version of Hardware drivers. If developers using the new version of encryption modules, the corresponding version drivers must be installed.
- Updating all Hardware drivers and the new driver will support the old version of encryption modules.
- Update the Win32 shell tool and improve the encryption intensity.
- During the installation of SDK, Developer will not choose the priority of visiting the Hardware Dog more; the encryption modules will judge it automatically.
- Add Kylix encryption module in Linux system.
- This version of SDK will not support the UMA/UMB type of Hardware Dogs, which have stopped production.
- Hardware Dog can support the Cascade function with the same Serial Number, but this function only be supported by the Hardware from UMC-PA5 upward.

Notice:

In the end client, the old drivers maybe cover over the new version drivers, if this case happened, it will cause the software encrypted by MicroDog V3.4 can't run correctly, and it will return 30012. The solution is to reinstall the new driver. Developer can use the Diagnose tool in the MicroDog SDK to check the driver information.

Updates from MicroDog SDK 3.2 to MicroDog SDK3.3

- Updating all Win32 modules and enhance the stabilities of multiple USB Dog used at the same time.
- The Win32 modules written in C#, VBNET and OMNI languages are added.
- The Win16 module written in Toolsbook 4.0 is added.
- The bran-new Win32 shell tool is provided which is easier to use and the function of time constraints is enhanced.
- Updating the Linux module. The USB driver based on the Linux kernel v2.4x is provided. The Dogs plugged both on the parallel port and USB port work on Linux kernel v2.4x.
- Updating the installation and uninstall programs for Hardware Dog's driver and the DLL for the installation and uninstall of the driver. The uninstall for USB driver become more perfect and the DLL interface for uninstalling the driver is added.

Updates from MicroDog SDK2.0 to MicroDog SDK3.2

Compared with MicroDog SDK 2.0, great changes have been made to MicroDog SDK V3.2. MicroDog SDK V3.2 adopts new hardware key --UMC. Originally, USB Dog driver integrates with Parallel Dog driver to avoid incompatibility of the two different types of Hardware Dog, which adversely influences the plug-and-play function.The USB Dog driver of MicroDog SDK V3.2 is separated from that of Parallel Dog driver, which supports plug-and-play indeed. MicroDog SDK V3.2 supports not only UMC Hardware Dog but also UMA and UMB Hardware Dog. It is strongly recommended to use UMC Hardware Dog. And MicroDog SDK V2.0 does not support new UMC Hardware Dog. All changes do not influence operations of Parallel Dog.

MicroDog SDK V3.2 has new added functions as follows:

- Easier driver installation and distribution. All process needs only InstDrv.exe.

  You can run Instdrv.exe in MS-DOS, please see Readme.txt document for more details.

- C language DLL provides higher security for calling DLL.
- Protection for multiple modules
- The new added DLL can effectively protect multiple modules and limit the using times. Adopting security protocol of DLL authentication, it ensures security when calling DLL.
- Remote upgrade for multiple modules

  Remote upgrade offers a setting method over Internet to developers. It is composed of two parts: Client end DLL and interfaces and Developer end DLL and interfaces.

- Support ActiveX

  MicroDog SDK V3.2 can support ActiveX objects. Developers can manage Hardware Dog with ActiveX in VB or VC language. For more information, please refer to Readme.txt and example program.

- Win32 module support cascade performance of Parallel Dog with the same Serial Number.
- Provide Parallel Dog driver on Linux 2.2 and 2.4. Developers can protect software products on Linux.
- Visual Fortran module is added
- AutoCAD module is added
- Examples of using Dll to install driver in Delphi, Visual Basic and PowerBuilder are

added.

- Examples of using Win32dll.dll in Delphi, Installshield are added.
- New DLL examples of including package in Win32/JAVA module are added.
- New DogEditor.

Chapter 1 Overview

The MicroDog SDK is an intelligent software protection solution. It is developed based on Parallel Dog and USB Dog and retains all features of these two products. MicroDog SDK includes the following components:

1. Hardware Dog

MicroDog SDK includes Parallel Dog for parallel ports and UMC Hardware Dog (USB Dog) for USB ports.

Hardware Dog is a programmable device, containing 200-byte memory. It is accessible with API functions when Hardware Dog is connected with the computer.

You can choose to use Parallel Dog or USB Dog in operating systems such as Windows 98/Me/2000/XP.

Parallel Dog complies with IEEE-1284 Parallel Port Standard, and is compatible with all standard peripherals.

USB Dog has the following features:

- **Transparent to all other USB peripherals:** USB port is designed to connect multiple devices simultaneously. It is helpful to avoid conflicts between the devices. Parallel port is designed for standalone devices and can hardly avoid conflicts between them.

- **More Stability:** USB port comes with a power cable, and can provide stable working power. This helps improve the working conditions for USB Dog. In contrast, parallel port does not have a power cable, and depends on signal cable or data cable for working power. This makes Parallel Dog hardly work normally on parallel port with low load capability.

2. API(Application Program Interface)

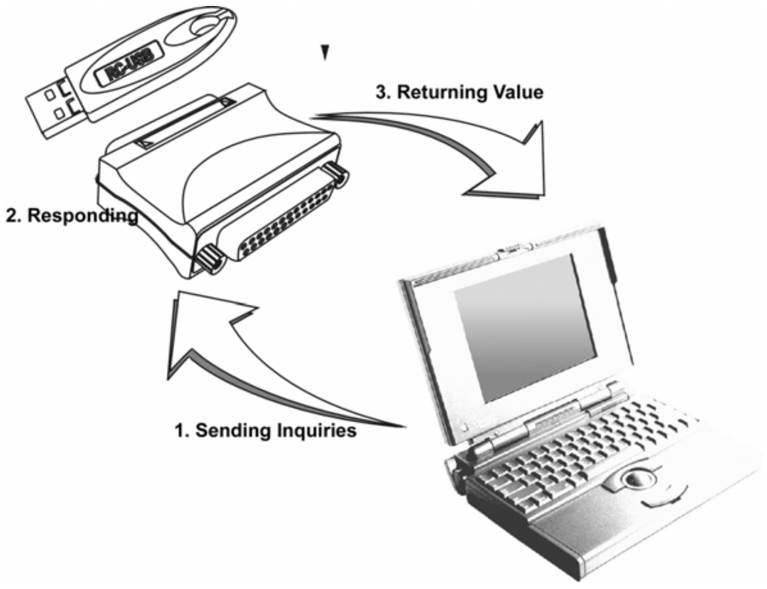
MicroDog SDK has the API for all kinds of computer languages, which can be embedded in the source code of your own program. With the API you can access the Hardware Dogs in your own program, please see Chapter 5 for more details about API.

3. Developer's tool

With the developer's tool(Dogedt32.exe) you can edit the memory of the Hardware Dogs, get the Serial Number and initialize the Hardware Dogs in batch, please see Chapter 3 for more details about the developer's tool.

How Does MicroDog SDK Work?

Applications manage Parallel Dog or USB Dog with API functions. Hardware Dog responds to this operation and returns values to applications. Applications verify the returned values and act accordingly. Please see the chart below:



MicroDog SDK offers two protection methods: obj, dll and ActiveX objects and Shell Protection. You can choose either to design your protection scheme.

Method1: Using obj, dll and ActiveX Objects

Obj, dll and ActiveX objects provide API functions to manage Hardware Dog. You can add API functions to the source code to protect the applications.

You can set the times to call API functions and how to act if correct Hardware Dog does not exist as you need. In principle, more API functions in source code will be more effective to prevent potential hackers.

Method2: Shell Protection

Shell protection provides a simple and fast solution. It “wraps” your applications automatically without modifying the source code. When the protected application runs, it will verify the presence of the correct Hardware Dog. Once the correct Hardware Dog does not exist, error message will display and the application will be terminated abnormally.

Shell Protection provides other functions such as setting the expiration date.

Features And Benefits

MicroDog SDK can run in the operating systems such as DOS, Windows 3X/9X/ME/NT/2000/XP, Linux and can support most programming languages and development tools. If you need to use USB Dog on Windows NT 4.0, please contact us for technical support. The MicroDog SDK has the following features.

- **Protection Algorithm** The Software developer may select his own algorithms by defining the bytes for algorithm and algorithm descriptors flexibly. Hardware Dogs contain a 200-bytes nonvolatile memory. You can use Developer’s tool or API functions to program the memory. The bytes for defining descriptor are from 196th to 198th and that of algorithm is 199th. So there are 256 different algorithm could be selected, algorithmic descriptor has 24-bits, the variations could be more than 4000 mega by changing both the bytes for algorithm and algorithmic descriptor.

- **Microprocessor** Hardware Dog contains a microprocessor. The physical program cannot be read or changed. This helps guarantee Hardware Dog cannot be duplicated illegally.

- **DERNT** Rainbow China Data Exchange Random Noise Technology enables Hardware Dog to effectively resist possible cracking from hardware or software debuggers such as a Logic Analyzer.

- **Maze Technology** Maze technology will effectively change the executive order of the programs to enhance UMH SDK's anti-tracing ability.

- **Timer** Hardware Dog contains a timer. This timer defines the maximum time that all operations may take to access the Dog. Normal operations will not exceed the time limit, but operations tracing the data in Hardware Dog usually will. If this happens, the Dog will return error values.

- **AS Technology** MicroDog SDK adopts the AS technology (the combination of API calls and Shell Protection) in its protection procedures. AS technology, in addition to inserting API calls into your program for protection, provides a shell protection tool (for executable files) to add one more level of protection (called "shell") to your program. The two levels of protection are correlated in shell protection. The API calls in your program will not be functional without the correlated "shell." In other words, if the hacker decrypts the "shell," the API calls in the program will not work correctly, and the cracking cannot continue.

- **Anti-sharing** Hardware Dog contains an anti-sharing device for parallel ports. It is up to you to decide whether to share Parallel Dog or not.

- **Password** You can set a 32-bit password in Hardware Dog. Without the correct password, data in memory cannot be read or written.

- **Memory** 200-byte nonvolatile memory for developers to store important data. The data can be read or written with API functions or DogEditor.

- **Manufacture code** A Hardware Dog has a unique Manufacture code that is sequential in the assembly line. Manufacture code can be used to distinguish each Hardware Dog. The Manufacture code of Parallel Dog and USB Dog, or Hardware Dog with same Serial Number may be different.

- **Cascade performance** You may cascade multiple Hardware Dog or other software protection hardware keys with different Serial Numbers to achieve more reliable security. Win32 modules support Parallel Dog cascade performance rather.

- **The Improved Security Engine II** We developed independently it, which integrates the latest cryptology and the advanced algorithm; it will promote the anti-cracked capability of software.

How to Purchase MicroDog SDK or Hardware Dogs?

You can choose to purchase MicroDog SDK or MicroDog Hardware Dogs. MicroDog Hardware Dogs includes Parallel Dog and USB Dog.

You can also choose to purchase PMH SDK including only Parallel Dog, or UM SDK including only USB Dog.

You can find Dog Key enclosed in an envelope, which you can use DogEdt32.EXE to convert Dog Key to Number.Dog. Number.Dog file is necessary when you order Hardware Dog.

One SDK matches with its own Serial Number. After protecting your software with the SDK, you need to purchase MicroDog Hardware Dog with the same Serial Number. If you decide to protect other software, you should purchase another SDK.

Developers must present Number.Dog when order Hardware Dog with the same Serial Number. Number.Dog is also needed when upgrading MicroDog modules. ***Please take good care of this file and keep it confidential.***

Chapter 2 Installing MicroDog SDK

Package List

MicroDog Suite includes one Parallel(PMH) Dog, one USB Dog (UMC type), one USB extension cable, one Installation CD (all protection tools and Utility Tools), one Developer’s Manual, one envelope enclosing Dog Key, one business card and quality certificate.

Parallel Dog SDK includes one Parallel Dog, and without one USB extension cable. USB Dog SDK includes one USB Dog.

Installing Hardware Dog

MicroDog Hardware Dog includes Parallel Dog and USB Dog.

Installing Parallel Dog

Parallel Dog can be inserted into any parallel port on the computer. Turn off the computer and the connected printer (where applicable). Attach the Hardware Dog to a parallel port on the computer and tighten the screws securely. If the computer connects with a printer or other parallel-port peripherals, please use a data cable at the back of Parallel Dog. Win32 module supports multiple Hardware Dogs with same Serial Number.

Installing USB Dog

USB Dog is a plug-and- play product. You can insert Hardware Dog into either of the two USB ports on the computer.

Note: You’d better not insert the USB Dog before installing the Hardware Dogs’ driver. Otherwise, the driver installation may fail.

If you have never installed USB Dog driver before, the operating systems including Windows 98/ME/2000/XP will display that a new hardware called "UMC (FV5.1)" is found once you insert the USB Dog(If you use the old Hardware Dog, it maybe display "USB Dog (Ver.C)") . The system requires you to install this driver. Please cancel this requirement first, and install MicroDog SDK. Select MicroDog drivers, and the installation program will begin to install USB Dog driver automatically.

Warning

- Please avoid plugging in or pulling out Parallel Dog with power on, especially when MicroDog Hardware Dog connects with a printer. This operation may make damage to the Parallel Dog, printers and the parallel ports of the computers. USB Dog is a plug-and-play product. However, do not shake USB Dog while pulling it out. It may damage USB Dog or the motherboard.

- Please avoid operating Parallel Dog while other peripherals connected with the parallel port are working simultaneously. It may induce parallel port conflicts. Please choose to use USB Dog or contact us for technical support.

- Only Operating systems such as Windows 98/Me/2000/XP/Linux can support USB Dog drivers. Please contact us if you need to run USB Dog on Windows NT 4.0.

- Please check if "Universal Serial Bus" exists in Control Panel/System/Device Manager to judge if your computer supports USB-port peripherals. Please note CMOS of some computers has ENABLE and DISABLE options.

- DOS16 and WIN16 modules in MicroDog SDK V3.2 can only operate USB Dog (UMA, UMB type) on Windows 98/ME/2000/XP but not UMC type. Operating Parallel Dog is not restricted by operating systems.

Installing Platform

MicroDog can be installed on following platforms: Windows® 9X, Windows® Me, Windows® NT, Windows® 2000 and Windows® XP.

Installing MicroDog SDK

Exit out of all running applications, especially anti-virus software in Windows before installation. Please place the CD onto the disc driver, the installation program will run automatically. If the program fails to run, please run SETUP.EXE file in the CD manually and complete the installation as instructed. The installation procedure includes: The installation of MicroDog SDK consists of the installation of the Hardware Dogs' driver and modules written in all kinds of the computer languages and the binding of the modules and the Hardware Dogs.

● Installing Hardware Dogs' driver

1. Install MicroDog Hardware Dogs' driver on Windows 9x/ME/2000

All those who used old versions driver must install MicroDog v3.4 driver again.The self-testing system in the installation program will check if the driver has been installed.

Note: You'd better not insert the USB Dog before installing the Hardware Dogs' driver. Otherwise, the driver installation may fail.

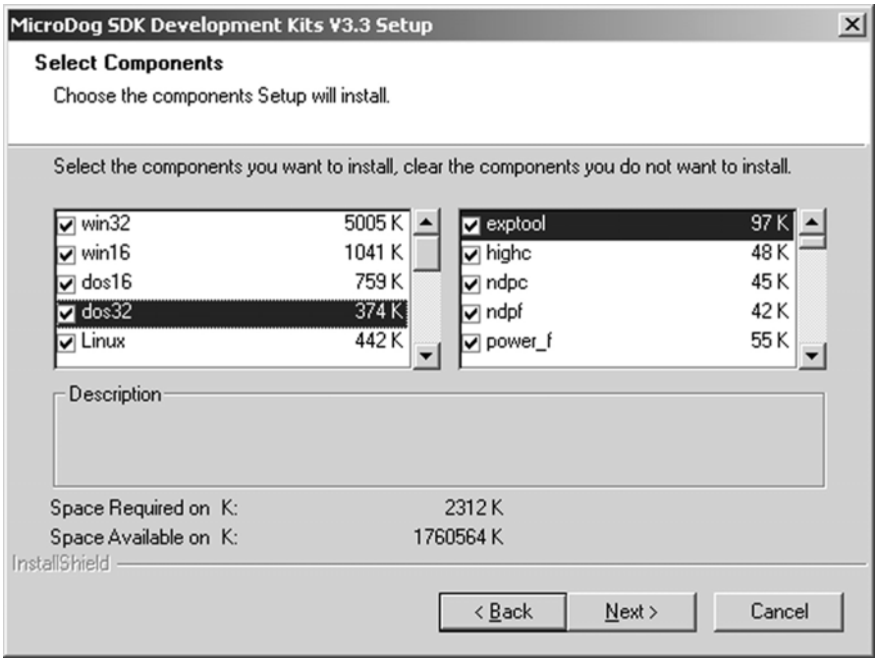
2. Install UMC Hardware Dog's driver on WINDOWS XP

1) Run the installation of the MicroDog SDK to install the Hardware Dog's driver or run the driver installation program(MicroDogInstdrv.exe) which lies in the Driver directory of SDK disk;

2) Plug USB Dog into the USB port, then the system will update the driver of USB Dog automatically because USB Dog has passed the Microsoft WHQL

certification.

- **Installing modules for various computer languages.**



MicroDog SDK Development Kits V3.3 Setup

Select Components

Choose the components Setup will install.

Select the components you want to install, clear the components you do not want to install.

✓ win32	5005 K
✓ win16	1041 K
✓ dos16	759 K
✓ dos32	374 K
✓ Linux	442 K

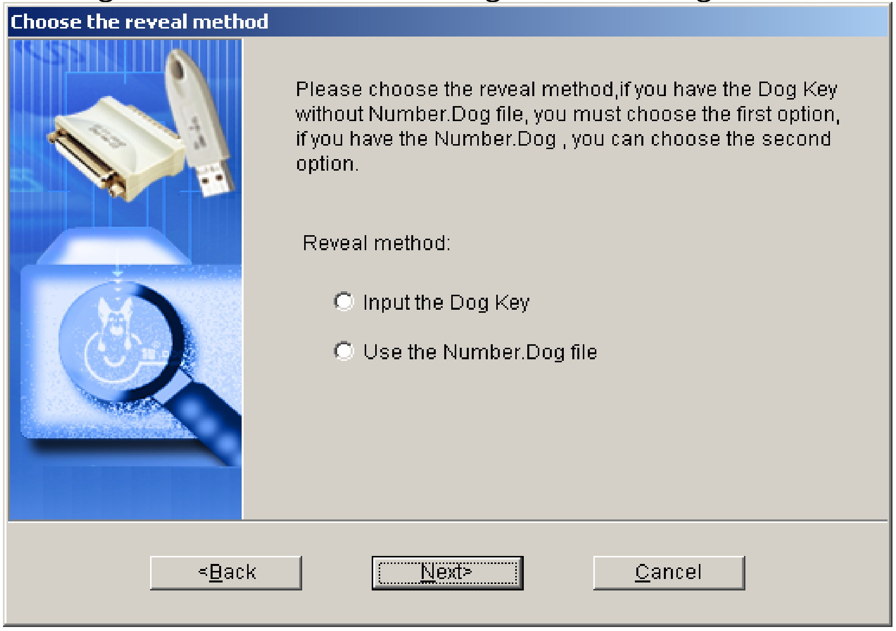
✓ exptool	97 K
✓ highc	48 K
✓ ndpc	45 K
✓ ndpf	42 K
✓ power_f	55 K

Description

Space Required on K: 2312 K

Space Available on K: 1760564 K

- **Binding the modules with MicroDog Hardware Dog.**



Choose the reveal method

Please choose the reveal method,if you have the Dog Key without Number.Dog file, you must choose the first option, if you have the Number.Dog , you can choose the second option.

Reveal method:

- ○ Input the Dog Key
- ○ Use the Number.Dog file

Use Dog Key or Number.Dog to bind encryption modules with MicroDog Hardware Dog.

Frequently Asked Questions on Installation

- The blue screen on Windows 9X shows that your computer has been infected with virus. Please kill the virus before installation.

- Please log in as system administrator during installation on Windows NT/2000.

Uninstalling MicroDog SDK

Uninstalling MicroDog SDK

Method 1:

Click “Start”, “Program”, “MicroDog SDK V3.4” and “Uninstall MicroDog SDK”.

Method 2:

Click “Start”, “Set”, “Control Panel” and “Add/Delete Program”. On the list select “MicroDog SDK” and then click “Add/Delete Program”.

Uninstall MicroDog Driver

Please run MicroDogInstdrv.exe located in \DRVIER\, in pop-up dialogue box click “Uninstall”.

Chapter 3 Using Developer’s Tool

On the tool bar please click “Start”, “Program”, “MicroDog SDK V3.4”, “MicroDog Utilities” and “DogEditor” to start the developer’s tool. Or you can run DogEdt32.EXE located in the subdirectory of UTILITY.

The developer’s tool provides the following functions:

Editing Memory

The Editing Memory Box offers functions such as editing, writing or reading data in user’s memory. The Editing Multiple-Module Memory Box offers functions such as editing multiple module memory, validity of modules and setting using times.

Hardware Dog Information

Hardware Dog Information Box offers functions such as setting passwords and cascade code, reading and writing Manufacture code and getting converted results. Error Code Inquiry Box can list all corresponding error codes.

Advanced Operation

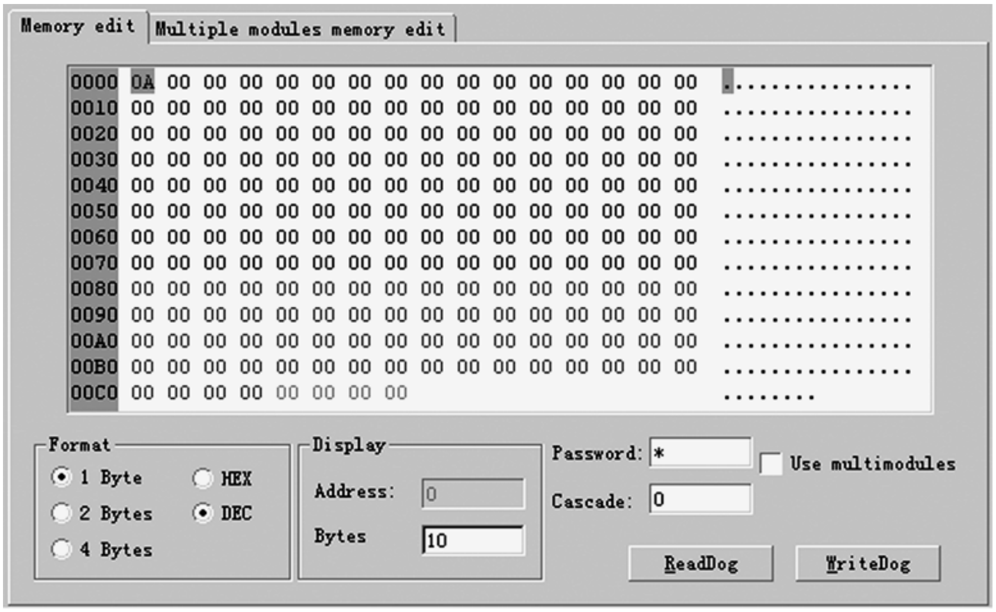
Initialization Box allows developers to write data into Hardware Dog in bulk conveniently and efficiently, and to change passwords or cascade codes during initialization. Developers can have a good and quick understanding to the API functions with API Testing Box.

Upgrade

You can find functions such as upgrading protection modules, multiple module remote upgrade, and the conversion between Number.Dog and Dog Key on the main upgrade menu.

Editing Memory

1) Editing Memory Box



```
0000 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00C0 00 00 00 00 00 00 00 00                          ........
```

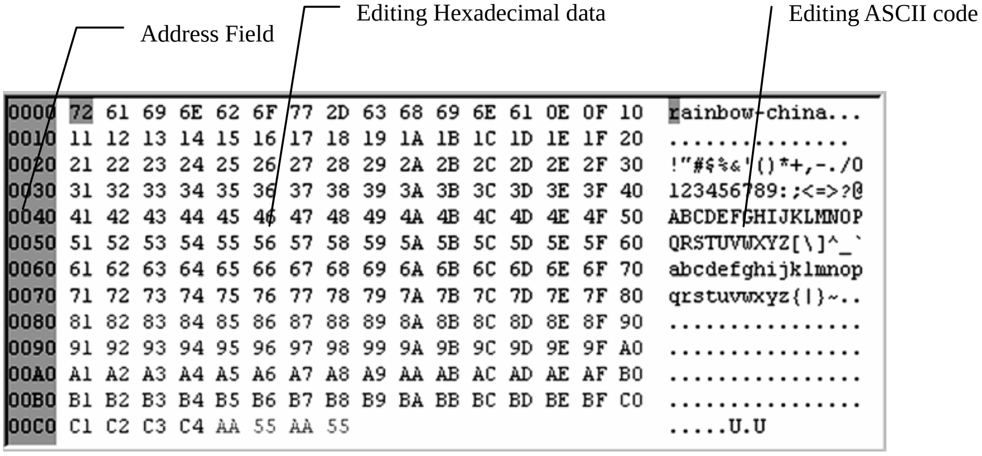
Format: [✓ 1 Byte] 2 Bytes 4 Bytes; HEX [✓ DEC]

Display: Address: 0; Bytes: 10

Password: *

Use multimodules ☐

Cascade: 0



```
0000 72 61 69 6E 62 6F 77 2D 63 68 69 6E 61 0E 0F 10  rainbow-china...
0010 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20  ................
0020 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30  !"#$%&'()*+,-./0
0030 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40  123456789:;<=>?@
0040 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50  ABCDEFGHIJKLMNOP
0050 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60  QRSTUVWXYZ[\]^_`
0060 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70  abcdefghijklmnop
0070 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80  qrstuvwxyz{|}~..
0080 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90  ................
0090 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0  ................
00A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0  ................
00B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0  ................
00C0 C1 C2 C3 C4 AA 55 AA 55                          .....U.U
```

Before editing, you should know more about the memory structure.

When using multiple module functions in the current user's cells, some cells are used to protect multiple modules and remote upgrade. The structure is as follows:

Size(Byte)	Content	Description
200-72 = 128	Common user's cells	All users' cells unused.
2	Multiple module valid restrictive number	Valid for reading status
2	Multiple module decrement restrictive number	Valid for reducing 1 from the current module using times
16*2 = 32	Current using times of 16 modules	Support 16 modules
16	Authentication key	Set by the developers. Used for authentication
16	Authentication key for remote upgrade	Used for remote upgrade authentication only.
4	Algorithm descriptors	A value of ULONG

The cells in black are common user's cells; cells in blue are for editing multiple modules, and cells in red are for defining descriptors.

Common User's Cells

Users can program these cells or save important data as required. The applications will read all relevant data from Hardware Dog.

Multiple Module Memory Cells

When protect multiple modules, do not repeat using 128-196 bytes. In other words, 128-196 bytes are not programmable if you check "Multiple Module"; otherwise this part can be used as common user's cells. If you do not select "Multiple Module", you cannot enter “Editing Multiple Module Memory” box.

Cells For Defining Algorithm And Algorithm Descriptors

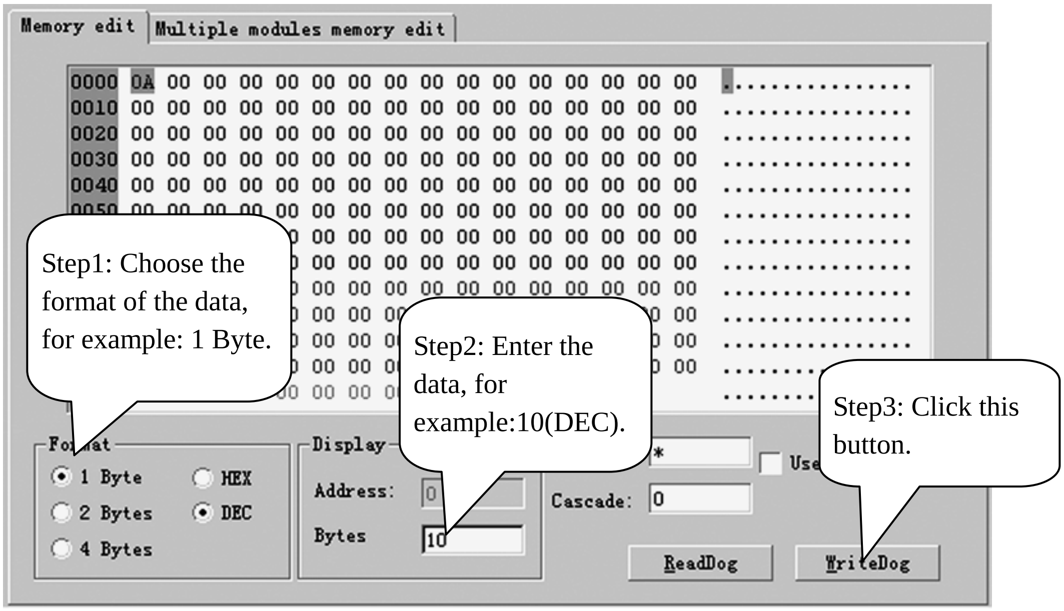
The Software developer may select his own algorithms by defining the bytes for algorithm and algorithm descriptors flexibly. You can use Developer’s tool or API functions (WriteDog()) to program the memory. The bytes for defining descriptor are from 196th to 198th and that of algorithm is 199th. So there are 256 different algorithm could be selected, algorithmic descriptor has 24-bits, the variations could be more than 4000 mega by changing both the bytes for algorithm and algorithmic descriptor.

The converted results of conversion functions (DogConvert()) are related to algorithm and algorithm descriptors.

There are two kinds of operations: ReadDog and WriteDog.

Writing All Data Into Hardware Dog

1. Follow the Step1, 2, 3 described on the Figure 3-1 to write data into the Hardware Dog.


Memory edit
Multiple modules memory edit
Step1: Choose the format of the data, for example: 1 Byte.
Step2: Enter the data, for example:10(DEC).
Step3: Click this button.
ReadDog
WriteDog

```
0000 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Format: [✓ 1 Byte] 2 Bytes 4 Bytes; HEX [✓ DEC]

Display — Address: 0 Bytes: 10

Cascade: 0

Figure 3-1

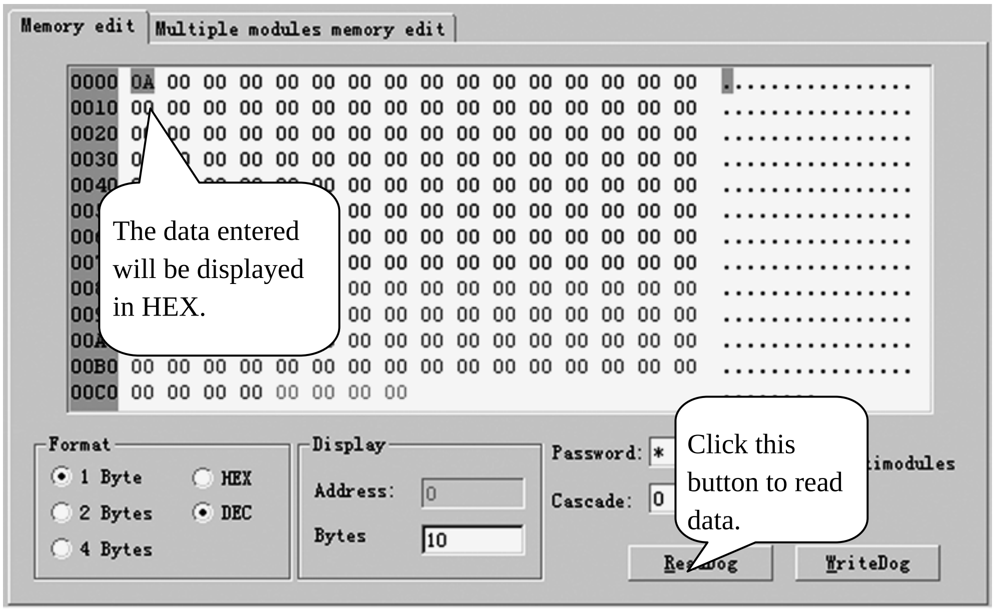
Caution:

After entering the data in the **Bytes** text box, the developer should click **WriteDog** button to write the data into the Dog, otherwise the data entered in the **Bytes** text box will be lost.

Please plug the Hardware Dog into the port of the computer before writing the Dog.

Reading All Data from Hardware Dog

Click "ReadDog" to read all data from the Hardware Dog.


Memory edit
Multiple modules memory edit
The data entered will be displayed in HEX.
Click this button to read data.
Format
1 Byte
2 Bytes
4 Bytes
HEX
DEC
Display
Address:
Bytes
Password:
Cascade:
ReadDog
WriteDog

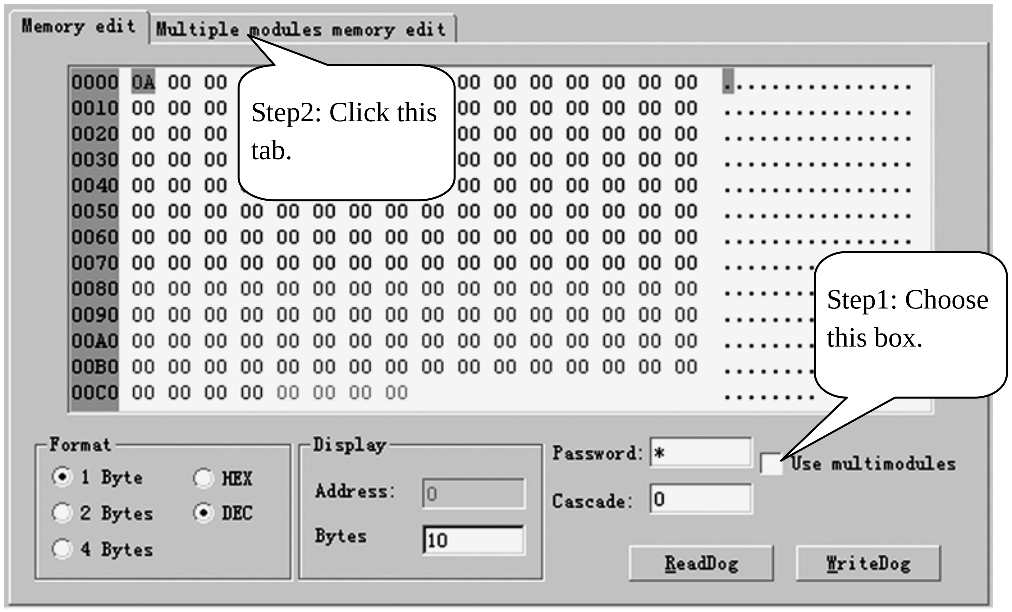
```
0000 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00C0 00 00 00 00 00 00 00 00
```

Format: [✓ 1 Byte] 2 Bytes 4 Bytes; HEX [✓ DEC]
Address: 0
Bytes: 10
Password: *
Cascade: 0

Figure 3-2

2) Multiple modules memory edit box

Please follow the steps described on the figure 3-3 and 3-4 to edit the multiple modules memory.



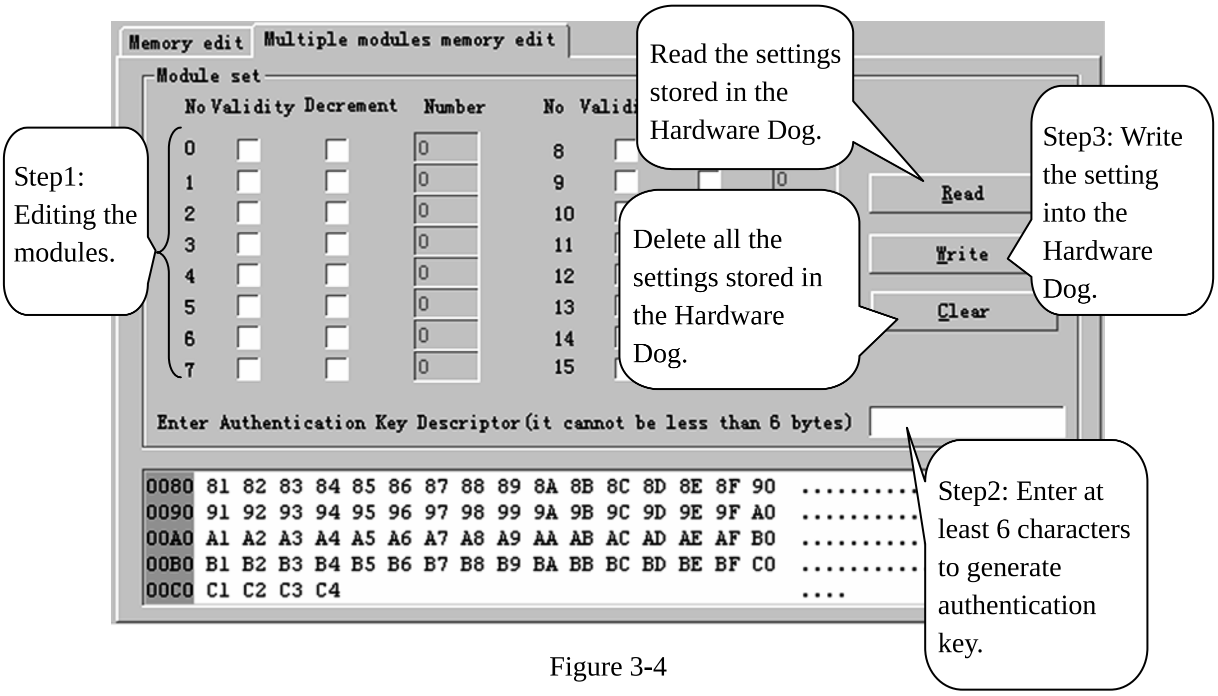
```
0000 0A 00 00                 00 00 00 00 00 00 00   ...............
0010 00 00 00                 00 00 00 00 00 00 00   ...............
0020 00 00 00                 00 00 00 00 00 00 00   ...............
0030 00 00 00                 00 00 00 00 00 00 00   ...............
0040 00 00 00                 00 00 00 00 00 00 00   ...............
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...............
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ...............
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .......
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .......
0090 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .......
00A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .......
00B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .......
00C0 00 00 00 00 00 00 00 00                           ........
```

Format: [✓ 1 Byte] 2 Bytes 4 Bytes HEX [✓ DEC]

Display: Address: 0 Bytes: 10

Password: * Cascade: 0 Use multimodules

Figure 3-3


Memory edit
Multiple modules memory edit
Module set
No Validity Decrement Number
No Validi
Read
Write
Clear
Read the settings stored in the Hardware Dog.
Step1: Editing the modules.
Step3: Write the setting into the Hardware Dog.
Delete all the settings stored in the Hardware Dog.
Step2: Enter at least 6 characters to generate authentication key.

No	Validity	Decrement	Number
0	☐	☐	0
1	☐	☐	0
2	☐	☐	0
3	☐	☐	0
4	☐	☐	0
5	☐	☐	0
6	☐	☐	0
7	☐	☐	0
8	☐		
9	☐	☐	0
10			
11			
12			
13			
14			
15			

Enter Authentication Key Descriptor(it cannot be less than 6 bytes)

```
0080 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F 90   .........
0090 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F A0   .........
00A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF B0   .........
00B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF C0   .........
00C0 C1 C2 C3 C4                                       ....
```

Figure 3-4

You can decide how to release the module in your application with this box.

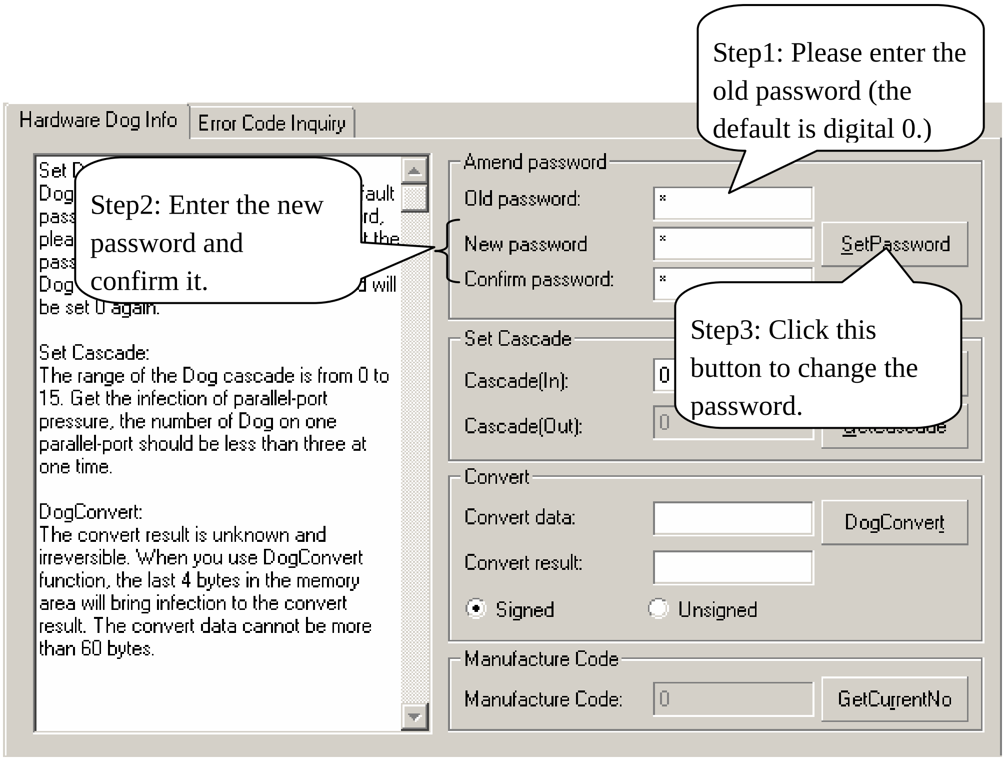
You can set up to 16 modules. If you need to set using times of a certain module, check "Decrement" and enter the using times. You can call API functions for multiple modules in your own program to verify the validity of the module. For more information about how to call API, please refer to the Readme file.

You can click "Clear" to delete all data in the multiple module memory and then re-configure data in it. Please note that you must click "Write" to place the new data in the multiple memory. Before that, you must enter up to 6 characters in the "Enter Authentication Key Descriptor" field to generate a secret key. This secret key will be used by multiple module API functions and high security DLL.

Hardware Dog Information

1) Hardware Dog Information Box

A. Follow the steps 1, 2, 3 described on the figure 3-5 to change the password.



Set D... Dog ... pass... plea... pass... Dog ... will be set 0 again.

Set Cascade:
The range of the Dog cascade is from 0 to 15. Get the infection of parallel-port pressure, the number of Dog on one parallel-port should be less than three at one time.

DogConvert:
The convert result is unknown and irreversible. When you use DogConvert function, the last 4 bytes in the memory area will bring infection to the convert result. The convert data cannot be more than 60 bytes.

Amend password
Old password: *
New password: *
Confirm password: *

Set Cascade
Cascade(In): 0
Cascade(Out): 0

Convert
Convert data:
Convert result:
[● Signed] Unsigned

Manufacture Code
Manufacture Code: 0

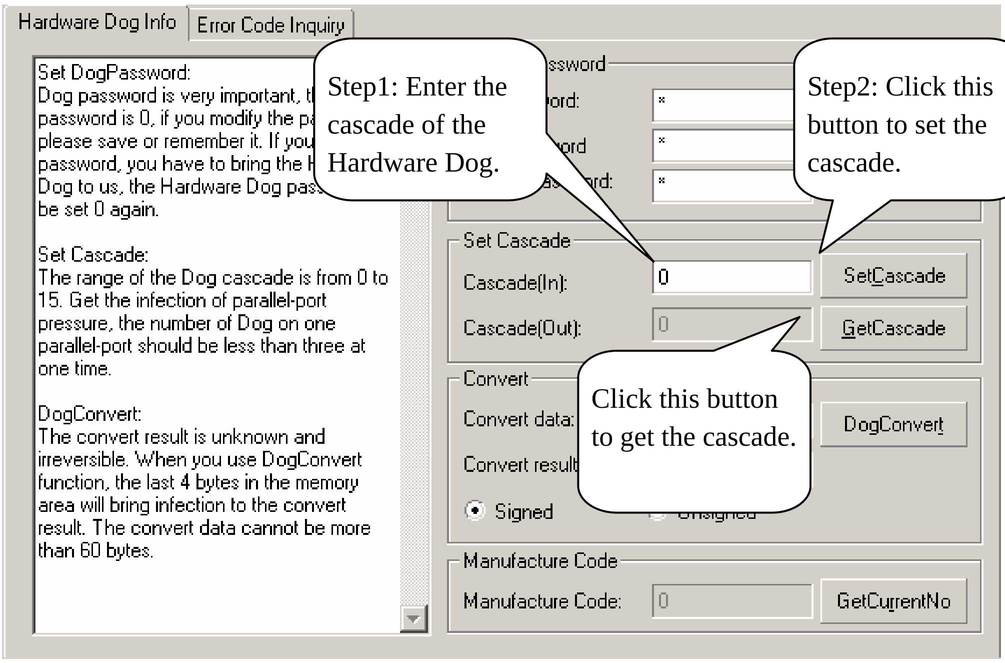
Hardware Dog Info
Error Code Inquiry
SetPassword
GetCascade
DogConvert
GetCurrentNo
Step1: Please enter the old password (the default is digital 0.)
Step2: Enter the new password and confirm it.
Step3: Click this button to change the password.

Figure 3-5

The default password is 0. The password should be a decimal number ranging from 0 to 4, 294, 967, 295. You can't do any operations of accessing the Hardware Dogs without the correct password. Password has nothing to do with the intensity of the encryption.

By setting the password you can forbid someone else to get or change the settings in the Hardware Dog, but the password has nothing to do with the encryption intensity.

B. Follow the steps 1, 2 described on the figure 3-6 to set the cascade.



Set DogPassword:
Dog password is very important, t password is 0, if you modify the pa please save or remember it. If you password, you have to bring the H Dog to us, the Hardware Dog pas be set 0 again.

Set Cascade:
The range of the Dog cascade is from 0 to 15. Get the infection of parallel-port pressure, the number of Dog on one parallel-port should be less than three at one time.

DogConvert:
The convert result is unknown and irreversible. When you use DogConvert function, the last 4 bytes in the memory area will bring infection to the convert result. The convert data cannot be more than 60 bytes.

Cascade(In): 0

Cascade(Out): 0

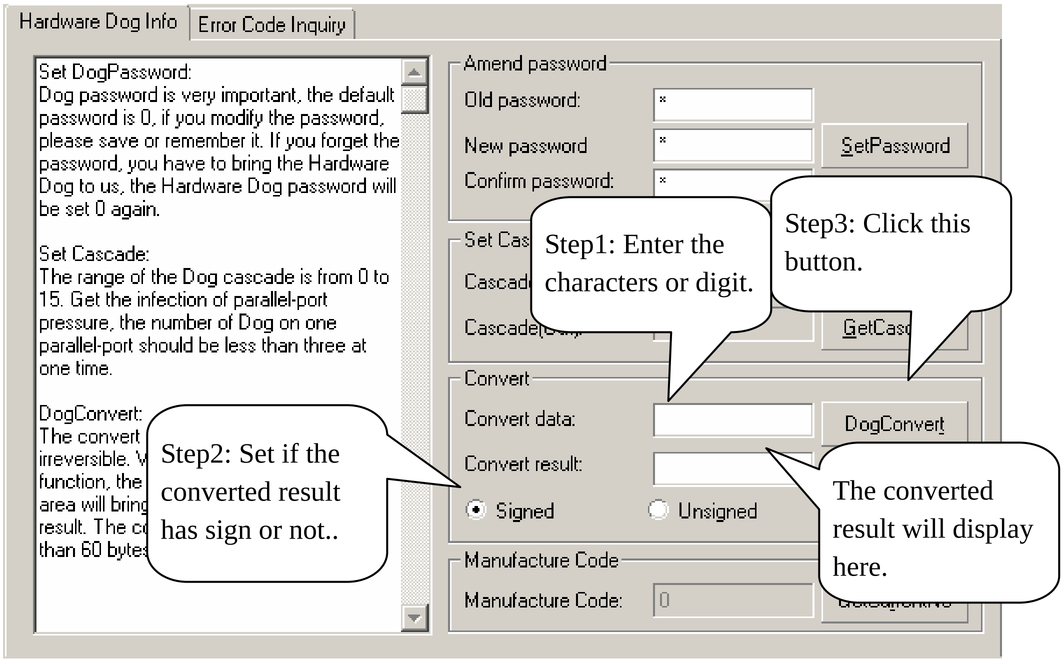
Signed [✓]

Manufacture Code: 0

Figure 3-6

C. Follow the Step1, 2 and 3 described on the Figure 3-7 to convert the data.

Convert(DogConvert()) is a process of converting the characters or digit according to the algorithm to be a Long-type data. Even you send the same values to different Dogs, the Dogs will generate different converted results. It is also *irreversible*: you cannot reproduce the original input from the converted result. The developer may select his own algorithm. The last four bytes of the 200-byte memory cells will affect the algorithm. The evaluation of the returned value is one part of the software lock. Hide the instruction for it in the program so that pirates cannot find them. Once the running of an illegal copy of the program is detected, it's better not to let the program respond directly; just leave some marks and change some values in the program so that the program runs abnormally.



Set DogPassword:
Dog password is very important, the default password is 0, if you modify the password, please save or remember it. If you forget the password, you have to bring the Hardware Dog to us, the Hardware Dog password will be set 0 again.

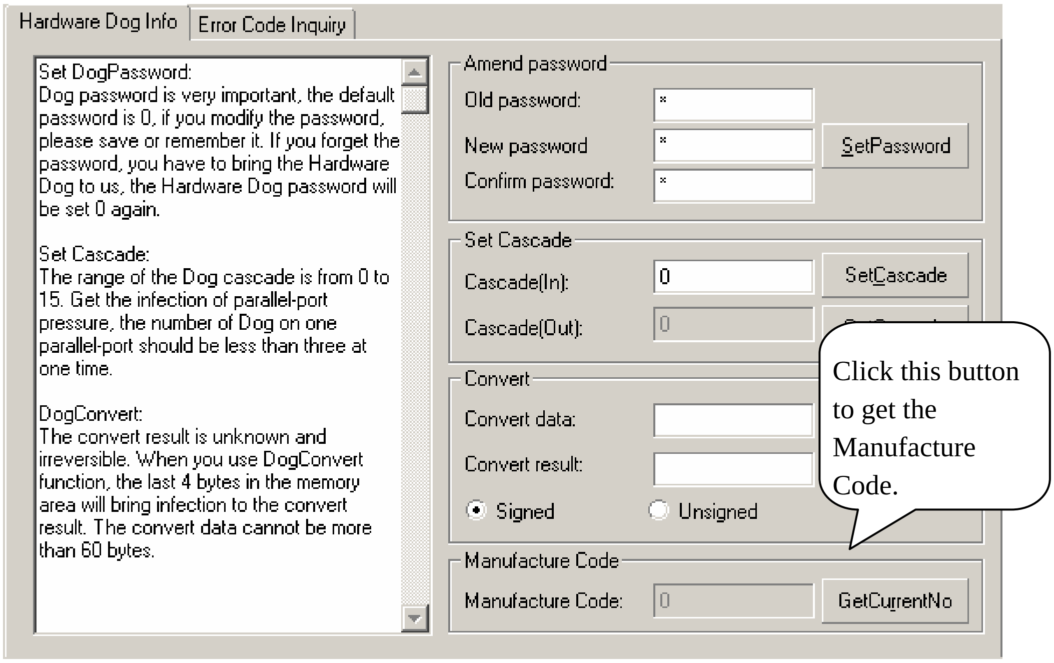
Set Cascade:
The range of the Dog cascade is from 0 to 15. Get the infection of parallel-port pressure, the number of Dog on one parallel-port should be less than three at one time.

DogConvert:
The convert
irreversible. V
function, the
area will bring
result. The co
than 60 bytes

Figure 3-7

D. Do as the description on the Figure 3-8 to get the Manufacture Code.

A parallel Dog or USB Dog has its unique Manufacture Code. Even the Dog with the same Serial Number has the different Manufacture Code. It is highly recommended to use a MicroDog SDK to protect a kind of software product and use the Manufacture Code to manage all the users of the software product.



Set DogPassword:
Dog password is very important, the default password is 0, if you modify the password, please save or remember it. If you forget the password, you have to bring the Hardware Dog to us, the Hardware Dog password will be set 0 again.

Set Cascade:
The range of the Dog cascade is from 0 to 15. Get the infection of parallel-port pressure, the number of Dog on one parallel-port should be less than three at one time.

DogConvert:
The convert result is unknown and irreversible. When you use DogConvert function, the last 4 bytes in the memory area will bring infection to the convert result. The convert data cannot be more than 60 bytes.

Old password: *
New password: *
Confirm password: *
Cascade(In): 0
Cascade(Out): 0
[✓ Signed] Unsigned
Manufacture Code: 0

Figure 3-8

2) Error Code Inquiry Box

Follow the Step1 and 2 described on the Figure 3-9, the error information will display corresponding to the error code. So the developer can solve the problems more quickly and more easily.

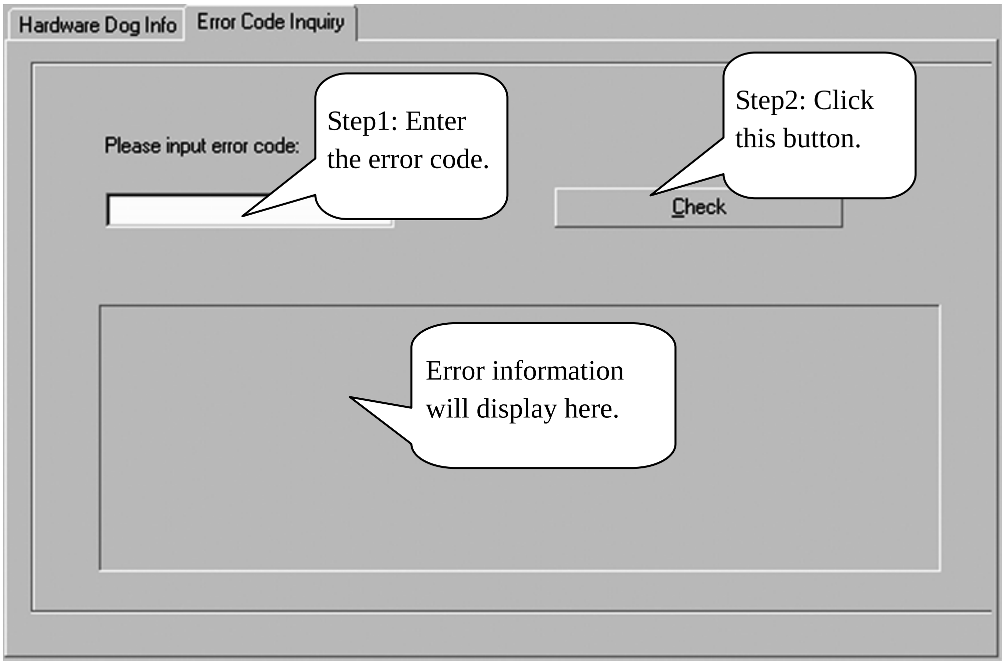
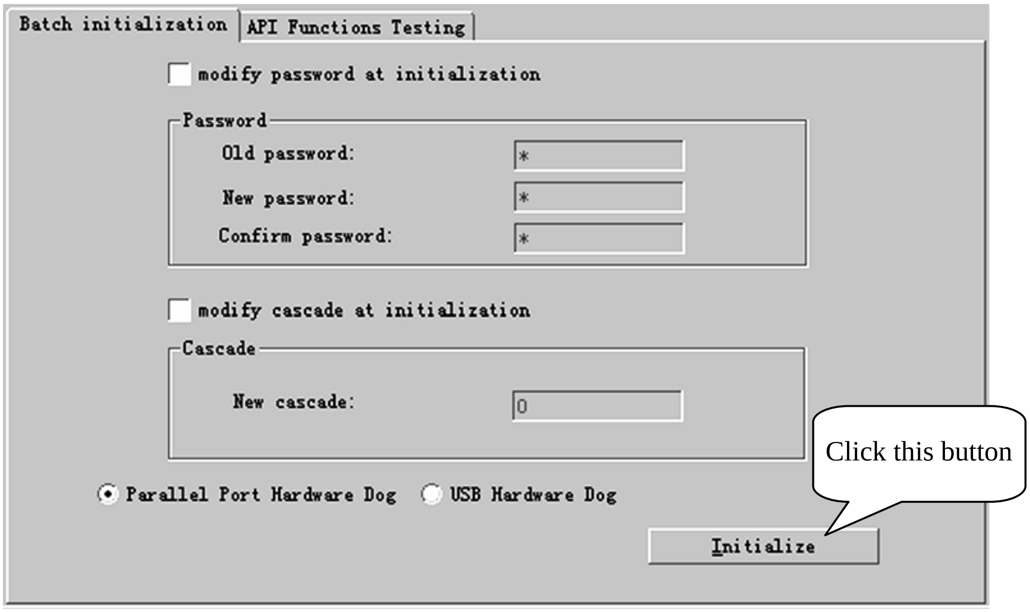

Hardware Dog Info
Error Code Inquiry
Please input error code:
Step1: Enter the error code.
Step2: Click this button.
Check
Error information will display here.

Figure 3-9

Advanced Operation

1) Initialization Box



☐ modify password at initialization

Password

Old password:	*
New password:	*
Confirm password:	*

☐ modify cascade at initialization

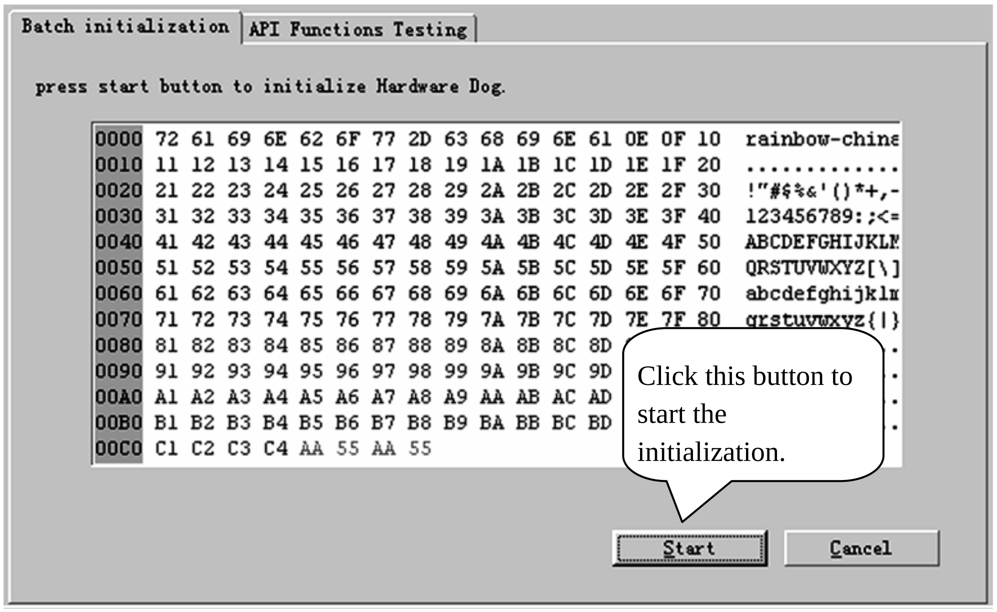
Cascade

New cascade:	0

[✓ Parallel Port Hardware Dog] USB Hardware Dog

Figure 3-10

You can initialize Hardware Dog in batches in this box after you have finished configuring the data. Before initialization, you can choose whether to initialize the password or cascade code or not. Check "Parallel Port Hardware Dog" or "USB Dog" and click "Initialize", and the following box will appear.



press start button to initialize Hardware Dog.

```
0000 72 61 69 6E 62 6F 77 2D 63 68 69 6E 61 0E 0F 10  rainbow-chine
0010 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20  .............
0020 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30  !"#$%&'()*+,-
0030 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F 40  123456789:;<=
0040 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50  ABCDEFGHIJKLM
0050 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60  QRSTUVWXYZ[\]
0060 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70  abcdefghijklm
0070 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F 80  qrstuvwxyz{|}
0080 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D           .
0090 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D           .
00A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD           .
00B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD           .
00C0 C1 C2 C3 C4 AA 55 AA 55
```

Figure 3-11

Please insert Hardware Dog, and click "Start" to initialize Hardware Dog in batches.

2) API Functions Testing Box

You can have an overall view of all API functions with this box by following the steps described on the figure 3-12.

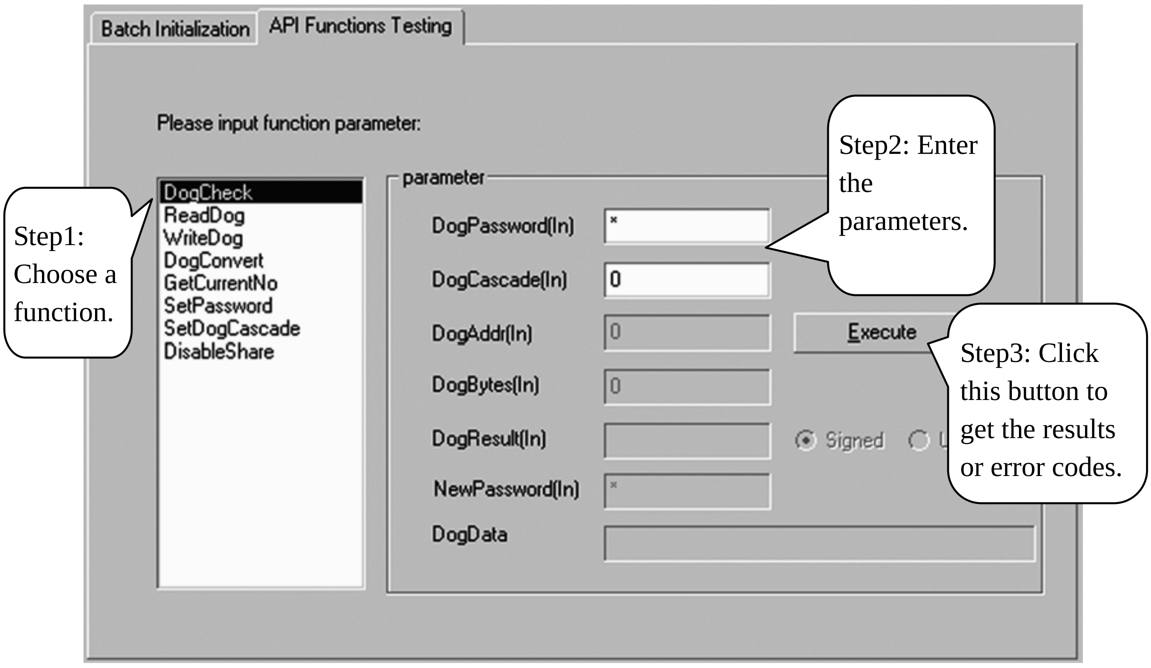


Figure 3-12

Upgrade

1) Protection Module Upgrade Box

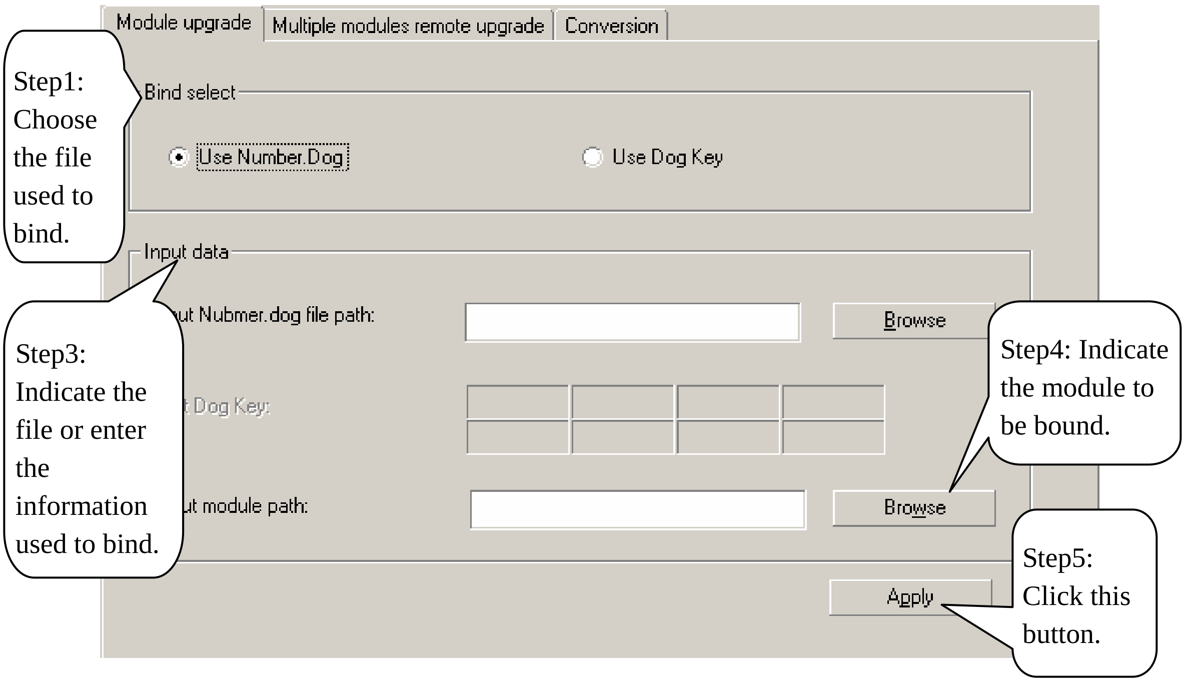

Module upgrade
Multiple modules remote upgrade
Conversion
Bind select
Use Number.Dog
Use Dog Key
Input data
Input Nubmer.dog file path:
Browse
Input Dog Key:
Input module path:
Browse
Apply
Step1: Choose the file used to bind.
Step3: Indicate the file or enter the information used to bind.
Step4: Indicate the module to be bound.
Step5: Click this button.

Figure 3-13

You cannot use upgraded modules directly you download from our corporate website because the protected modules cannot match with Hardware Dog. Please use the box above to process the downloaded module before using.

Please note "Preference of Hardware Dog" works only in Win32 module.

Please make sure that you hold Number.Dog or Dog Key when binding Hardware Dog with the modules

2) Multiple Module Remote Upgrade Box

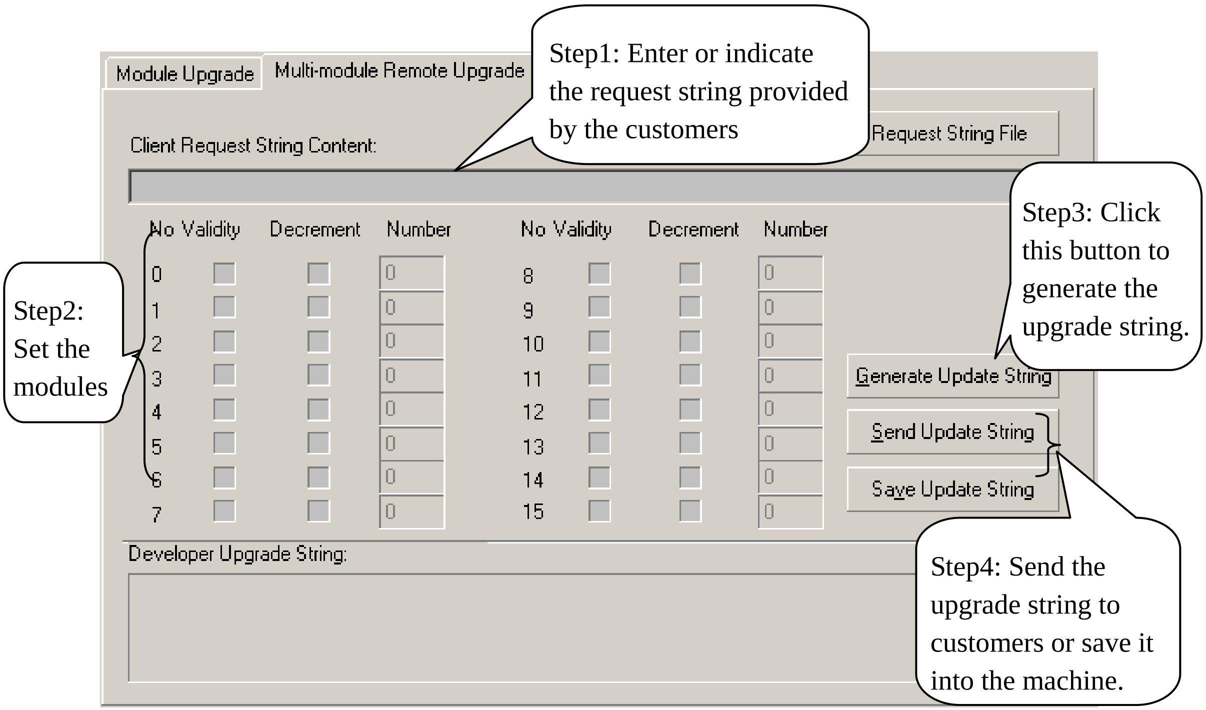

Module Upgrade
Multi-module Remote Upgrade
Client Request String Content:
Request String File
No Validity Decrement Number
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15
Generate Update String
Send Update String
Save Update String
Developer Upgrade String:
Step1: Enter or indicate the request string provided by the customers
Step2: Set the modules
Step3: Click this button to generate the upgrade string.
Step4: Send the upgrade string to customers or save it into the machine.

Figure 3-14

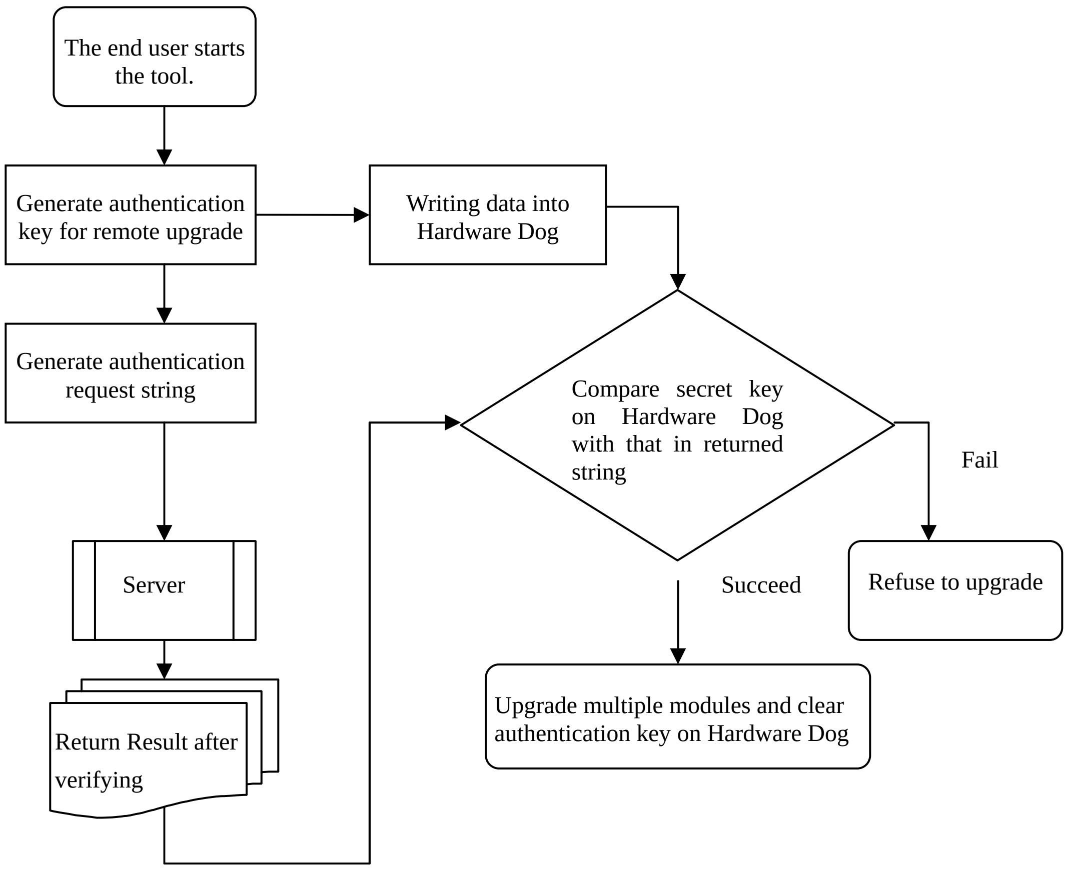
This function allows you to upgrade the configuration on Hardware Dog remotely. It has two parts: Client End DLL and Interfaces, Developer End DLL and Interfaces.

Client End DLL has two API functions. One is for generating a string requesting remote upgrade, and the other is for upgrading according to the returned result.

Developer End DLL has four API functions. They are used to verify the requesting from Client End and generate the returned result.

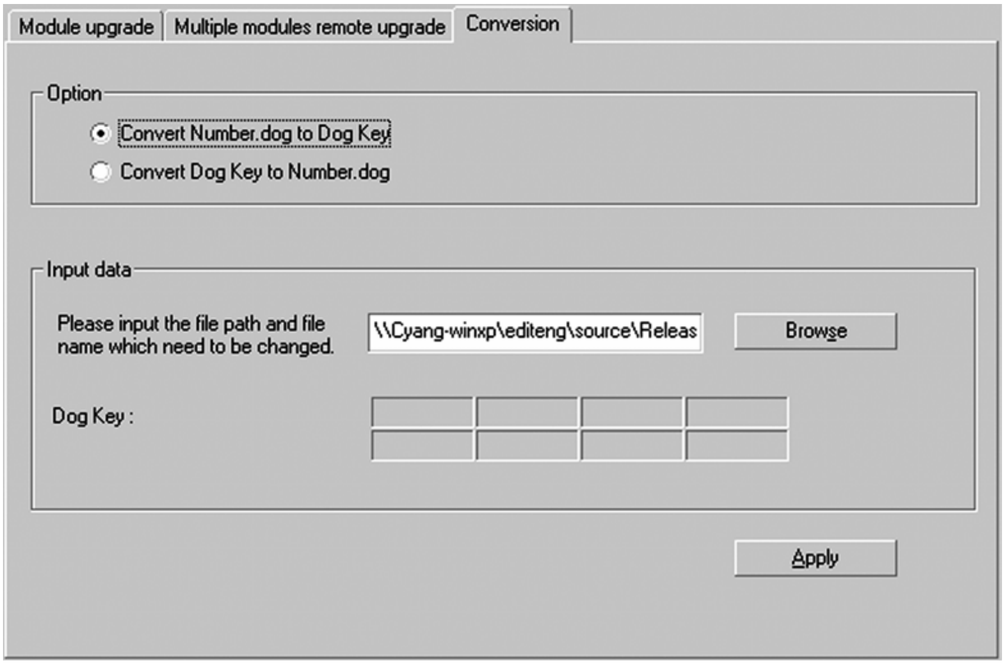
You can enter the Client End's requesting string, and release some modules as required.

After configuration, please click "Generate Update String" to create the upgrade string. Click "Send Update String" to email the string to the end users. You can also click "Save Update String" to save the configuration file.


The end user starts the tool.
Generate authentication key for remote upgrade
Writing data into Hardware Dog
Generate authentication request string
Compare secret key on Hardware Dog with that in returned string
Fail
Server
Succeed
Refuse to upgrade
Return Result after verifying
Upgrade multiple modules and clear authentication key on Hardware Dog

Remote Upgrade Chart

3) Conversion Box



Convert Number.dog to Dog Key [✓]

Convert Dog Key to Number.dog

Please input the file path and file name which need to be changed. \\Cyang-winxp\editeng\source\Releas

Dog Key :

Figure 3-15

Developers must present Number.Dog when ordering Hardware Dog with the same Serial Number.

In this box, Number.Dog can be converted to Dog Key and vice versa. If it is not your first time to install the software, you can choose to use Number.Dog rather than Dog Key for your convenience.

Convert Number.Dog to Dog Key

Step 1: Click "Convert Number.Dog to Dog Key";

Step 2: Enter the full path and name of Number.Dog in the "please enter the file path and file name which need change" field. Or you can browse the location of Number.Dog;

Step 3: Click “Convert”. Dog Key content appears in the “Dog Key is” field.

Convert Dog Key to Number.Dog

Step 1: Click “Convert Dog Key to Number.Dog”;

Step 2: Enter Dog Key content in the “Dog Key is” field;

Step 3: Please enter the location and name of Number.Dog created in the “Enter file path and name” field. Or you can click “Browse” to the designated location for the saved file. Click “Convert” to create Number.Dog in the designated location.

Chapter 4 Using Shell Protection

Shell Protection For 32-bit Windows Application

Developers can protect their applications efficiently without source code or without making any changes in the source code. Shell protection needs the presence of corresponding Hardware Dog (Parallel Dog or USB Dog).

Shell protection has the following features:

1. Three protection types can be adopted: Shell, Embed and Combination.

Shell: Shell is a small routine that is enveloped to the application. When the protected application starts, Shell will be executed firstly before the protected application runs. Shell will check presence and consistency of the correct Hardware Dog.

Embed: In whole execution period, the protected application will check presence and consistency regularly. This method is optional.

Combination: If the protected application has called APIs provided by Rainbow China, when Shell runs, it will inform APIs that Shell was executed; otherwise APIs refuse to run correctly. Therefore Shell cannot be skipped.

2. Data files such as DBF, BMP, JPG can be protected.

3. There are four constraints can be selected for protected application. If any constraint is set, 10-byte memory will be occupied in the Hardware Dog. This 10-byte memory should not be modified by other applications.

(a) Allows you to control when the application can be run, by specifying a static expiration date. When the date is reached, the application will no longer run.

(b) Allows you to limit the number of times the application can be run. When the times is reached, the application will no longer run.

(c) Allows you to control how long the application can be run for, in weeks, days, hours or minutes. The timer begins the first time the application is executed.

(d) Allows you to control how many days the application can be run from its first running. A expiration date will be calculated at its first running. When the date is reached, the application will no longer run.

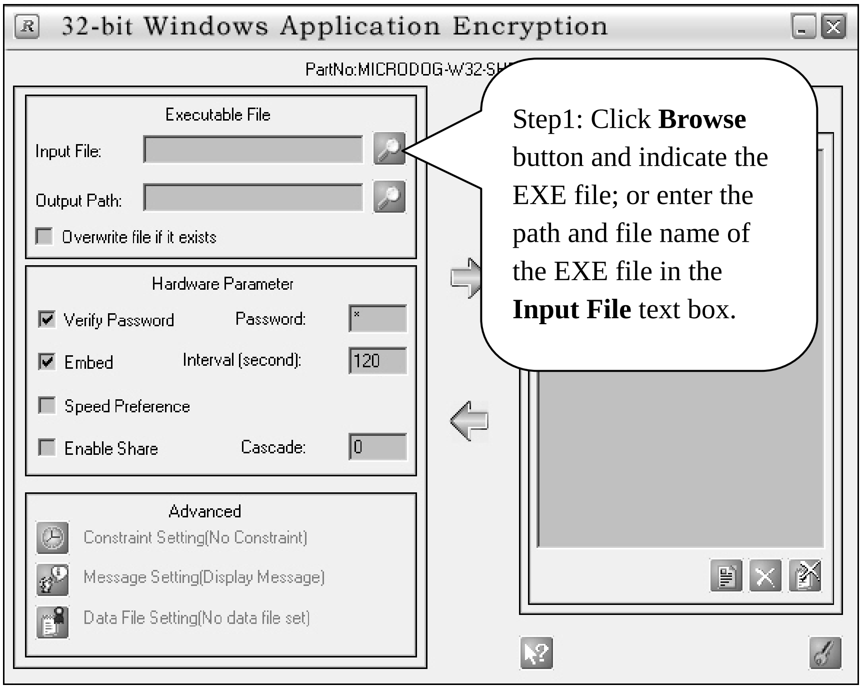
4. Enable or disable share one Parallel Dog among several PCs. Share is implemented by a device called Parallel Port Sharer which connects parallel ports of several PCs physically to one Hardware Dog. One Parallel Dog can be shared among several copies of application running on several PCs.

5. Verify password which can be set with DogEdt32.exe distributed by Rainbow China. If any constraint is set, Verify Password should be selected.

6. You can define the message to be displayed to the user when the Hardware Dog is not correct. You can also determine not to display any messages when error occurs. There are 7 messages to be displayed in different cases. Please specify these messages and the titles. Please follow the example which demonstrates how to encrypt Notepad.exe and Calc.exe by shell tool.

Please follow the example of encrypting Notepad.exe and Calc.exe by shell tool to be familiar with shell tool.

1. Plug into the Hardware Dog first, following the Step 1 described on Figure 4-1 to indicate the EXE program that will be encrypted.



Verify Password: ✓ Password: *
Embed: ✓ Interval (second): 120
Speed Preference: unchecked
Enable Share: unchecked Cascade: 0
Overwrite file if it exists: unchecked

Step1: Click **Browse** button and indicate the EXE file; or enter the path and file name of the EXE file in the **Input File** text box.

Figure 4-1

After indicating the Input file, the path of encrypted file will automatically display in the **Output Path** text box. The default Output path will be OUTPUT subdirectory of the path of the Input file. It is highly recommended to adapt the default.

Overwrite file if it exists

If the output file already exists, if Overwrite is selected, the output file will be overwritten without prompt. Otherwise, it will prompt you to decide whether to overwrite the existing file or not.

2. You can set the hardware parameters of the Dog.

Speed Preference

When Speed Preference is selected, the protected applications can run more quickly.

Enable Share

Parallel Port Sharer is a device provided by the third part. It connects parallel ports of several PCs, allowing PCs to share one parallel Dog. When several copied protected applications run on PCs, only one parallel Dog is needed by using the Parallel Port Sharer.

If you do not want the protected application shared by several PCs through Parallel Port Sharer, please do not select Enable Share.

Verify Password

You can choose whether to verify the password or not. If Verify Password is selected, the password will be verified when running the encrypted program. A Hardware Dog may be assigned a 4-bytes password. You may set and change the passwords with \Utility\Dogedit\DogEdt32.exe. The default setting is 0, which is a decimal number ranging from 0 to 4,294,967,295. You cannot access the Hardware Dog without the correct password. When the Expiration Constraint is set, this option must be selected.

Embed

During the encrypted application runs, it can continually check whether the Hardware Dog attaches to the port or not. This feature is optional. It does not check the Hardware Dog if Embed is not selected. If Embed is selected and the Hardware Dog does not exist, the application will exit abnormally.

You may specify the check interval. The unit is second, the default is 120 seconds (2 minutes).

Interval(second)

You can set the interval of the check. The unit is second and the default is 120 seconds.

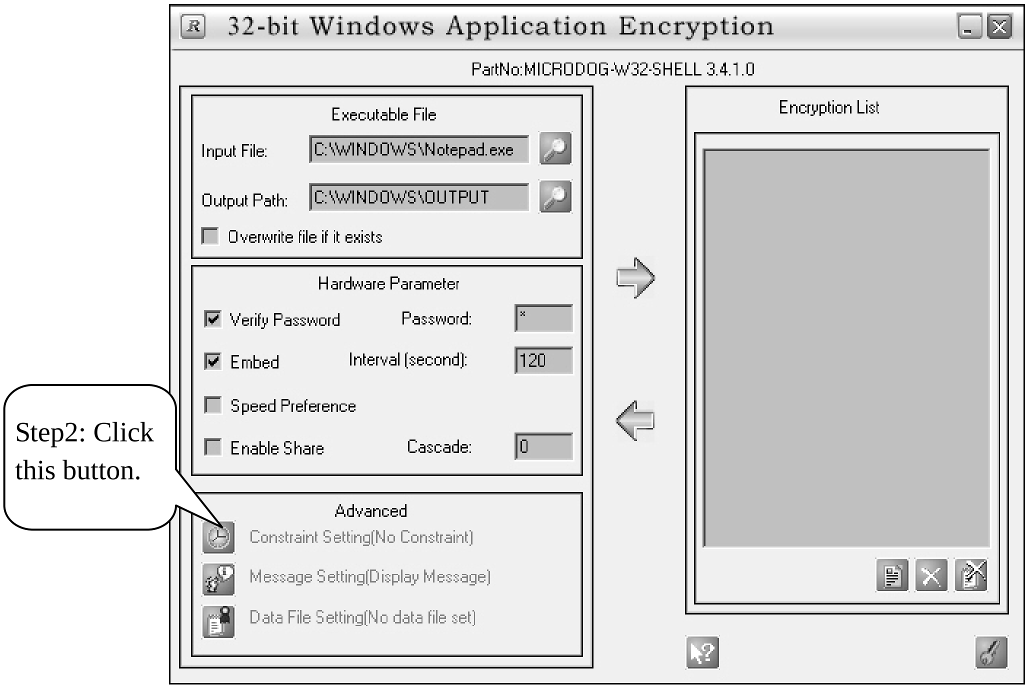
Password

A Hardware Dog may be assigned a 4-bytes password. You may set and change the passwords with \Utility\Dogedit\DogEdt32.exe. The default setting is 0, which is a decimal number ranging from 0 to 4,294,967,295. You cannot access the Hardware Dog without the correct password.

Cascade

If the cascade of the Hardware Dog is not 0, you should set this option.

3. Following the Step 2, 3, 4, 5, 6 described on the Figure 4-2, 4-3 to set expiration date.



32-bit Windows Application Encryption

PartNo:MICRODOG-W32-SHELL 3.4.1.0

Executable File

Input File: C:\WINDOWS\Notepad.exe

Output Path: C:\WINDOWS\OUTPUT

☐ Overwrite file if it exists

Hardware Parameter

☑ Verify Password Password: *

☑ Embed Interval (second): 120

☐ Speed Preference

☐ Enable Share Cascade: 0

Advanced

Constraint Setting(No Constraint)

Message Setting(Display Message)

Data File Setting(No data file set)

Encryption List

Figure 4-2

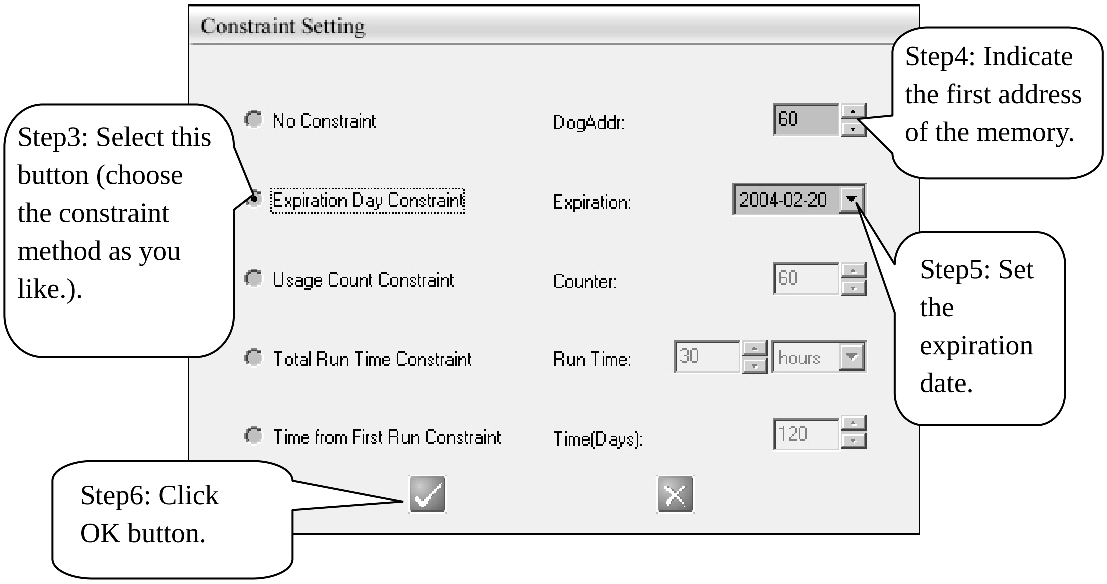

Constraint Setting
No Constraint
Expiration Day Constraint
Usage Count Constraint
Total Run Time Constraint
Time from First Run Constraint
DogAddr: 60
Expiration: 2004-02-20
Counter: 60
Run Time: 30 hours
Time(Days): 120
Step3: Select this button (choose the constraint method as you like.).
Step4: Indicate the first address of the memory.
Step5: Set the expiration date.
Step6: Click OK button.

Figure 4-3

DogAdd

If any constraint is set, 10-byte memory will be occupied in the Hardware Dog. This 10-byte memory should not be modified by other applications.

Expiration Day Constraint

Allows you to control when the application can be run, by specifying a static expiration date. When the date is reached, the application will no longer run.

Usage Count Constraint

Allows you to limit the number of times the application can be run. When the times is reached, the application will no longer run. You can select a maximum of 60000 times.

Total Run Time Constraint

Allows you to control how long the application can be run for, in weeks, days, hours or minutes. The timer begins the first time the application is executed. You can select a maximum of 360 days or equivalent weeks, hours, or minutes.

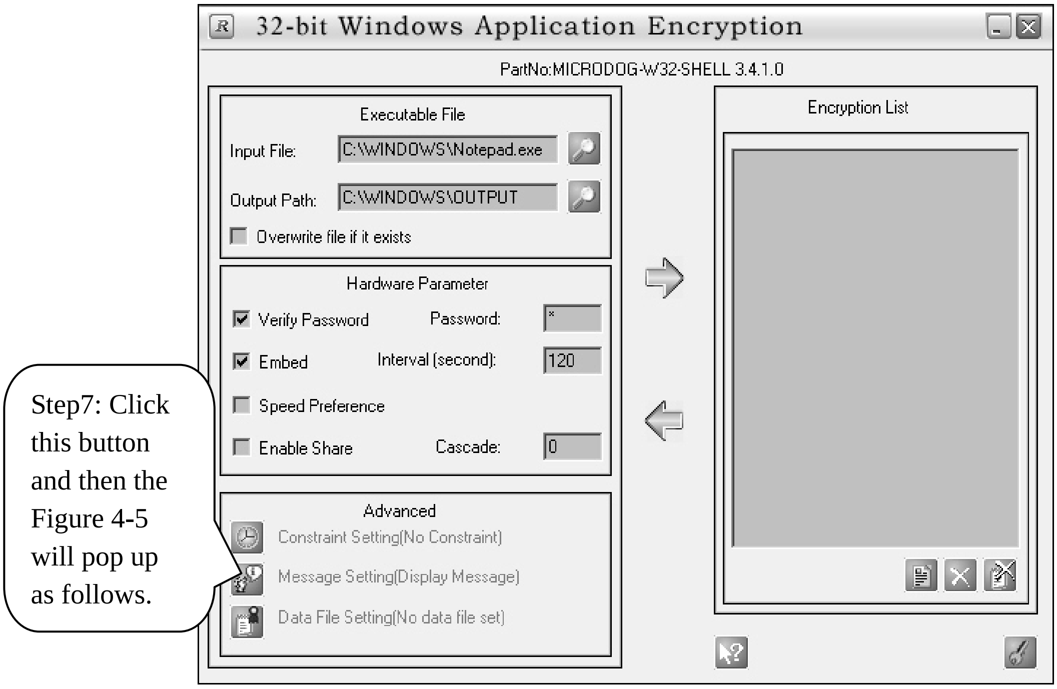
Time from First Run Constraint

Allows you to control how many days the application can be run from its first running. A expiration date will be calculated at its first running. When the date is reached, the application will no longer run. You can select a maximum of 360 days.

Notice

You should fully test the protected application before distributing it to the end user. Please initialize the Hardware Dog before testing because the protected application may modified the corresponding memory in the Hardware Dog.

4. There are 7 messages to be displayed in different cases. Please specify these messages and the title.The title's length is limited to 60 bytes and the messages' to 100 bytes. Follow step 7, 8, 9, 10, 11 described on the figure 4-4, 4-5 to set the message.


32-bit Windows Application Encryption
PartNo:MICRODOG-W32-SHELL 3.4.1.0
Executable File
Encryption List
Hardware Parameter
Advanced
Step7: Click this button and then the Figure 4-5 will pop up as follows.

Input File: C:\WINDOWS\Notepad.exe
Output Path: C:\WINDOWS\OUTPUT
Overwrite file if it exists
Verify Password ✓ Password: *
Embed ✓ Interval (second): 120
Speed Preference
Enable Share Cascade: 0
Constraint Setting(No Constraint)
Message Setting(Display Message)
Data File Setting(No data file set)

Figure 4-4

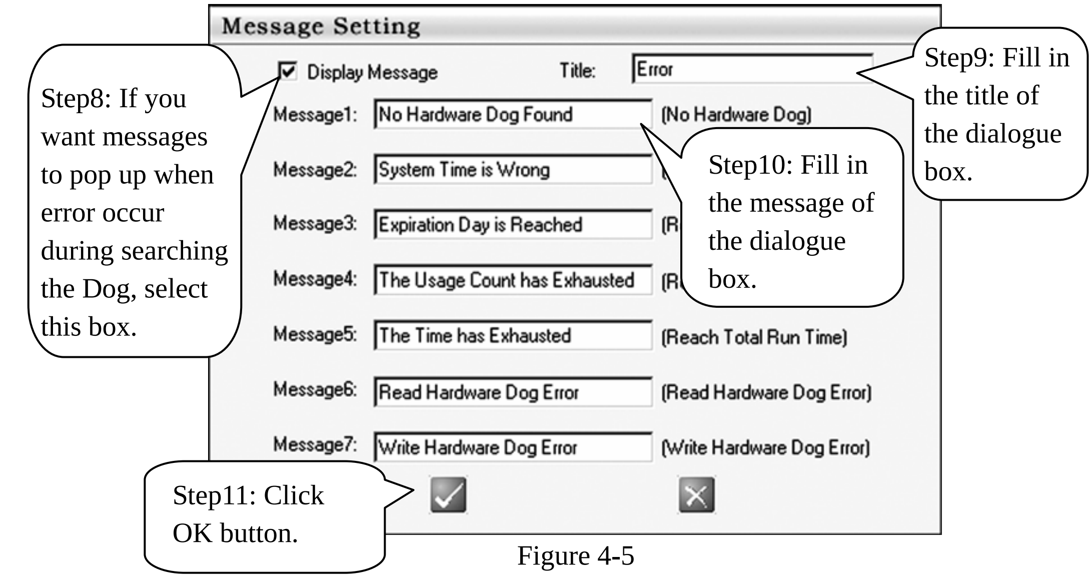

Message Setting
☑ Display Message
Title: Error
Message1: No Hardware Dog Found (No Hardware Dog)
Message2: System Time is Wrong
Message3: Expiration Day is Reached (R
Message4: The Usage Count has Exhausted (R
Message5: The Time has Exhausted (Reach Total Run Time)
Message6: Read Hardware Dog Error (Read Hardware Dog Error)
Message7: Write Hardware Dog Error (Write Hardware Dog Error)
Step8: If you want messages to pop up when error occur during searching the Dog, select this box.
Step9: Fill in the title of the dialogue box.
Step10: Fill in the message of the dialogue box.
Step11: Click OK button.

Figure 4-5

5. If you want to encrypt data file while encrypting the EXE file, follow the Step12 on Figure 4-6:

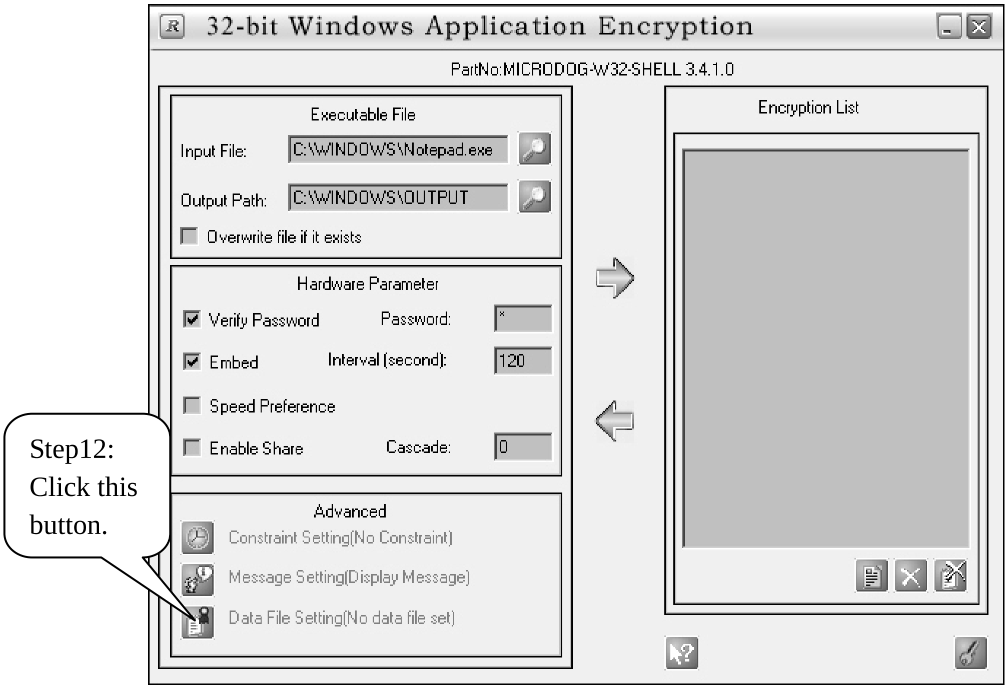
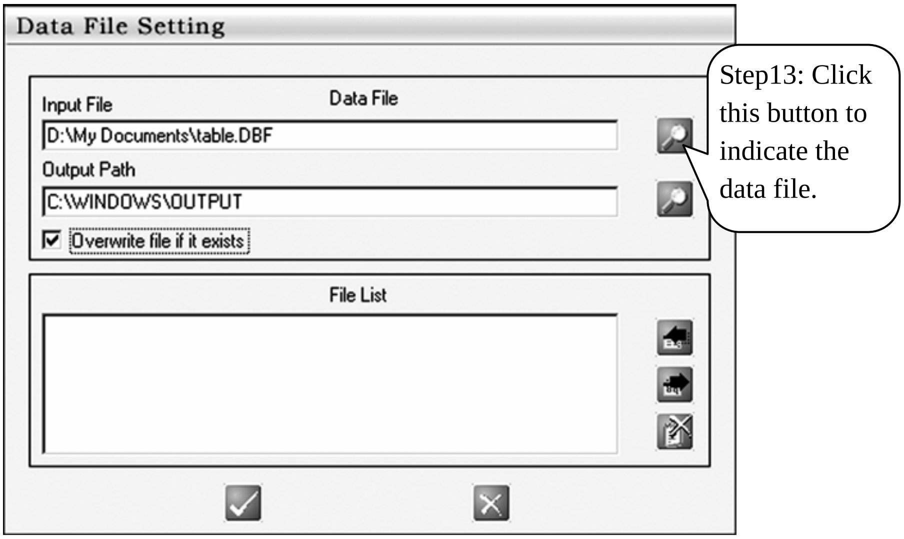

32-bit Windows Application Encryption
PartNo:MICRODOG-W32-SHELL 3.4.1.0
Executable File
Input File: C:\WINDOWS\Notepad.exe
Output Path: C:\WINDOWS\OUTPUT
☐ Overwrite file if it exists
Hardware Parameter
☑ Verify Password Password: *
☑ Embed Interval (second): 120
☐ Speed Preference
☐ Enable Share Cascade: 0
Advanced
Constraint Setting(No Constraint)
Message Setting(Display Message)
Data File Setting(No data file set)
Encryption List
Step12: Click this button.

Figure 4-6

6. Following the Step 13, 14, 15 on the Figure 4-7, 4-8 to indicate the data file (In fact, Notepad.exe needn't a data file):



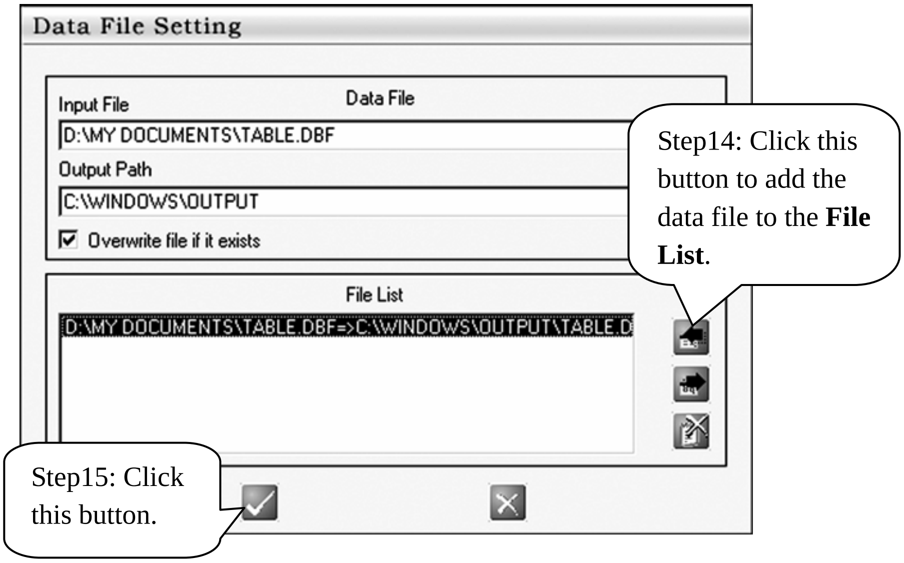
Input File: D:\My Documents\table.DBF

Output Path: C:\WINDOWS\OUTPUT

☑ Overwrite file if it exists

Figure 4-7

7. Repeat the Step 13, 14 and 15 to add multiple files to the File List


Data File Setting
Data File
Input File
Output Path
File List
Step14: Click this button to add the data file to the **File List**.
Step15: Click this button.

Input File: D:\MY DOCUMENTS\TABLE.DBF

Output Path: C:\WINDOWS\OUTPUT

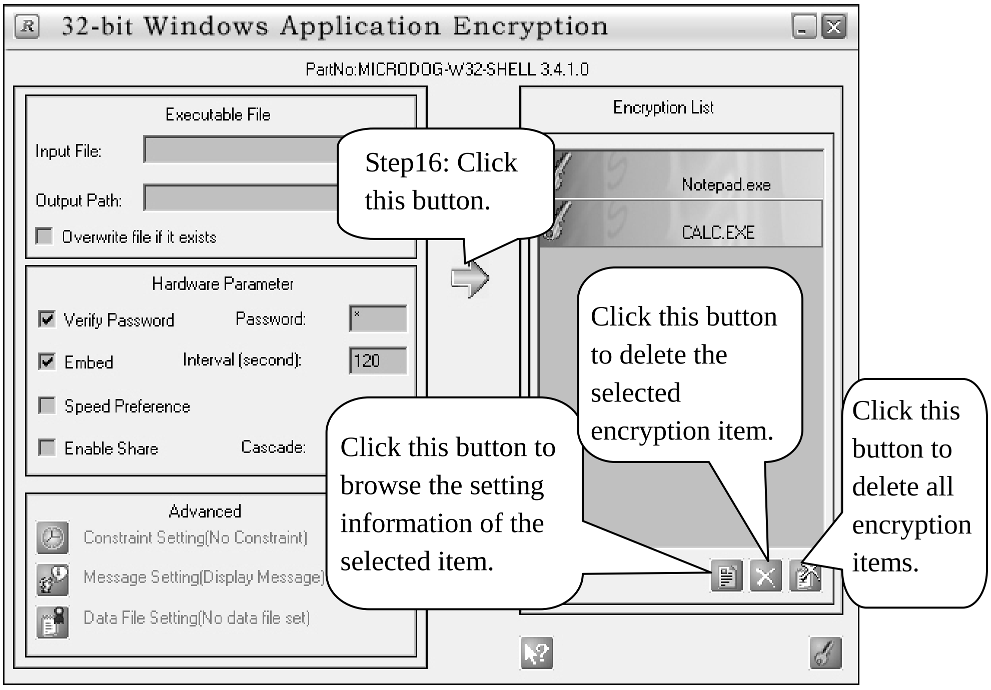
☑ Overwrite file if it exists

File List: D:\MY DOCUMENTS\TABLE.DBF=>C:\WINDOWS\OUTPUT\TABLE.D

Figure 4-8

For convenience, wildcard * can be used in the Input File.

8. Follow the Step16 on the Figure 4-9 to add Notepad.exe to the Encryption List. Repeat the options above(from Step 1 to 16) with Calc.exe. If you set a expiration constraint to Calc.exe, change the **DogAdd** option (see Figure 4-10), the value of **DogAdd** must be different from that of Notepad.exe.



PartNo:MICRODOG-W32-SHELL 3.4.1.0

Input File:

Output Path:

Overwrite file if it exists

Verify Password [✓] Password: *

Embed [✓] Interval (second): 120

Speed Preference

Enable Share Cascade:

Constraint Setting(No Constraint)

Message Setting(Display Message)

Data File Setting(No data file set)

Notepad.exe

CALC.EXE

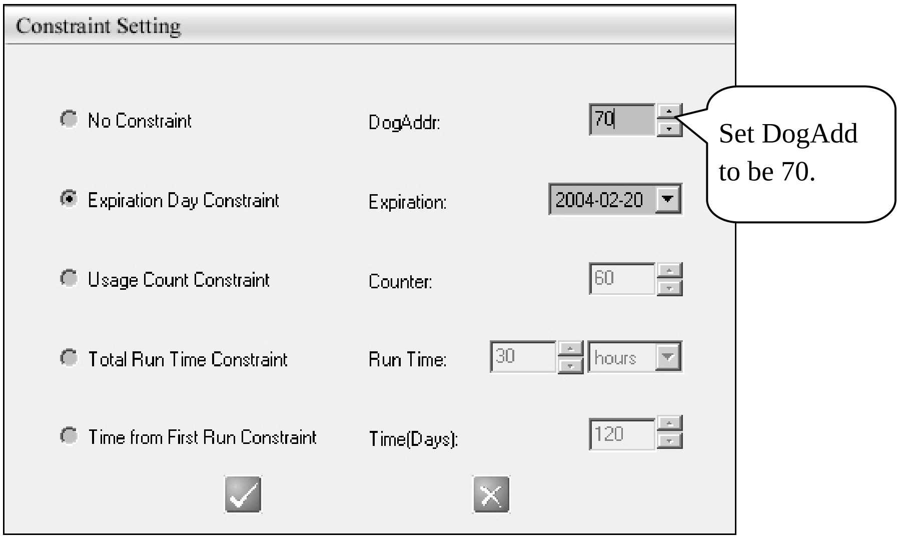
Step16: Click this button.

Click this button to delete the selected encryption item.

Click this button to browse the setting information of the selected item.

Click this button to delete all encryption items.

Figure 4-9



Constraint Setting

( ) No Constraint — DogAddr: 70

(●) Expiration Day Constraint — Expiration: 2004-02-20

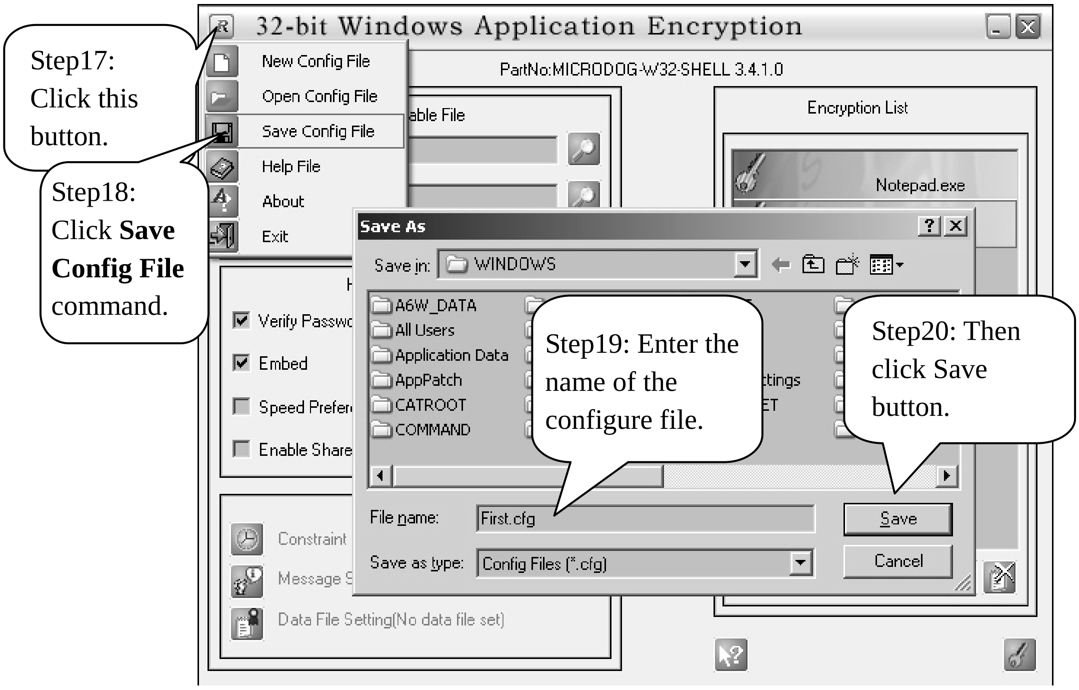
( ) Usage Count Constraint — Counter: 60

( ) Total Run Time Constraint — Run Time: 30 hours

( ) Time from First Run Constraint — Time(Days): 120

Figure 4-10

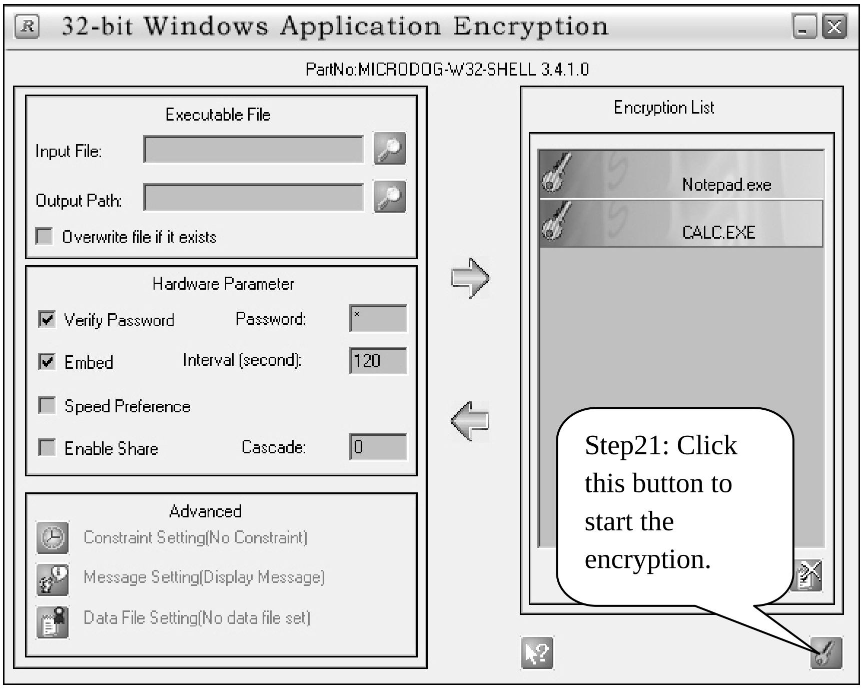
9. Follow the Step 17-19 on the Figure 4-11 to save all the settings of the encryption items in the Encryptions List to be a configure file. You can use the configure file again by opening it.


Step17: Click this button.
Step18: Click **Save Config File** command.
Step19: Enter the name of the configure file.
Step20: Then click Save button.
32-bit Windows Application Encryption
New Config File
Open Config File
Save Config File
Help File
About
Exit
PartNo:MICRODOG-W32-SHELL 3.4.1.0
Encryption List
Notepad.exe
Save As
Save in: WINDOWS
A6W_DATA
All Users
Application Data
AppPatch
CATROOT
COMMAND
Verify Passwo
Embed
Speed Prefer
Enable Share
Constraint
Message S
Data File Setting(No data file set)
Save
Cancel

File name: First.cfg
Save as type: Config Files (*.cfg)

Figure 4-11

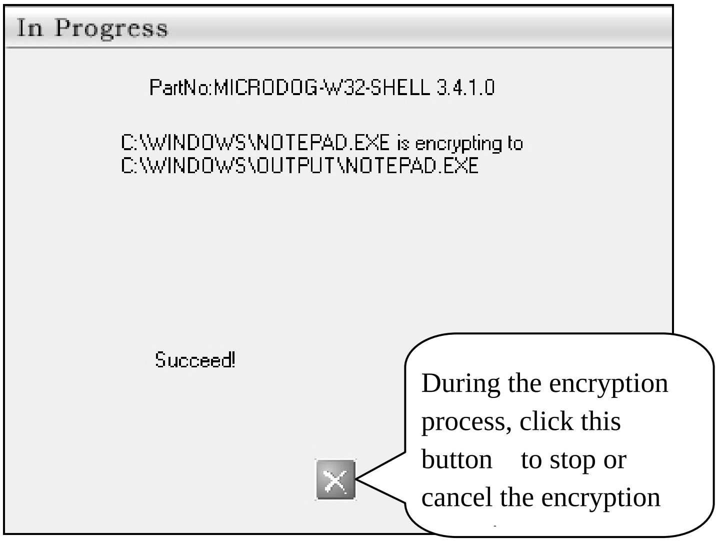
10. Follow the Step 21 on the Figure 4-12 to execute the encryption.


32-bit Windows Application Encryption
PartNo:MICRODOG-W32-SHELL 3.4.1.0
Executable File
Input File:
Output Path:
Overwrite file if it exists
Hardware Parameter
Verify Password
Password:
Embed
Interval (second):
Speed Preference
Enable Share
Cascade:
Advanced
Constraint Setting(No Constraint)
Message Setting(Display Message)
Data File Setting(No data file set)
Encryption List
Notepad.exe
CALC.EXE
Step21: Click this button to start the encryption.

Password: *
Interval (second): 120
Cascade: 0

Figure 4-12

11. During the encryption process the following dialogue box will pop up.


In Progress
During the encryption process, click this button to stop or cancel the encryption

PartNo:MICRODOG-W32-SHELL 3.4.1.0

C:\WINDOWS\NOTEPAD.EXE is encrypting to
C:\WINDOWS\OUTPUT\NOTEPAD.EXE

Succeed!

Figure 4-13

A message will display whether the encryption process is successful or not. Error message will give you a hand for solving problems. For more information about the error code see errcode-eng.txt. So far, you complete the operation of respectively protecting Notepad.exe and Calc.exe.

Notes

- For some EXE file, the corresponding DLL will be encrypted in the same way. The encrypted DLL is in the Output Path too. These files should be distributed to the end user too.

- For convenience, wildcard * can be used in the Input file name on Data File Setting Dialog.

- If any constraint is set, 10-byte memory will be occupied in the Hardware Dog. This 10-byte memory should not be modified by other applications. And Verify Password should be selected now.

- It has been known that this program does not support the following types of executable files.

1. The file that has been encrypted; you want it to be encrypted for the second time.

2. The file that has been decrypted after encrypted; you want it to be encrypted for the second time.

3. The file that has been compressed by compression program ASPACK or UPX.

- During encryption process, Hardware Dog is needed. You should fully test the protected application before distributing it to the end user. Please initialize a Hardware Dog for testing because the protected application may modify the corresponding memory in the Hardware Dog. Please initialize other Hardware Dogs using the Hardware Dog which used during encryption process when distributing your application to the end user.

  For more information please refer to Help file.

Shell Protection For Applications on Windows 3X

Shell protection for 16-bit Windows application is located in directory \WIN3X\EXETOOL\WIN16EXE.EXE. After running Shell protection, you can select EXE file type, password, interval, disable share and protect database files. Enter the Input file name and output file name, and add the file names to the file list. Click “Run” to complete protection. For more information on WIN16EXE.EXE, please refer to Readme file.

Shell Protection For Executable Files And Database Files on DOS

For executable files in DOS such as .EXE and .COM files, MicroDog offers EXE.EXE Shell protection located in \DOS16\EXETOOL. This tool also has functions such as verifying password, setting trial period and disabling share.

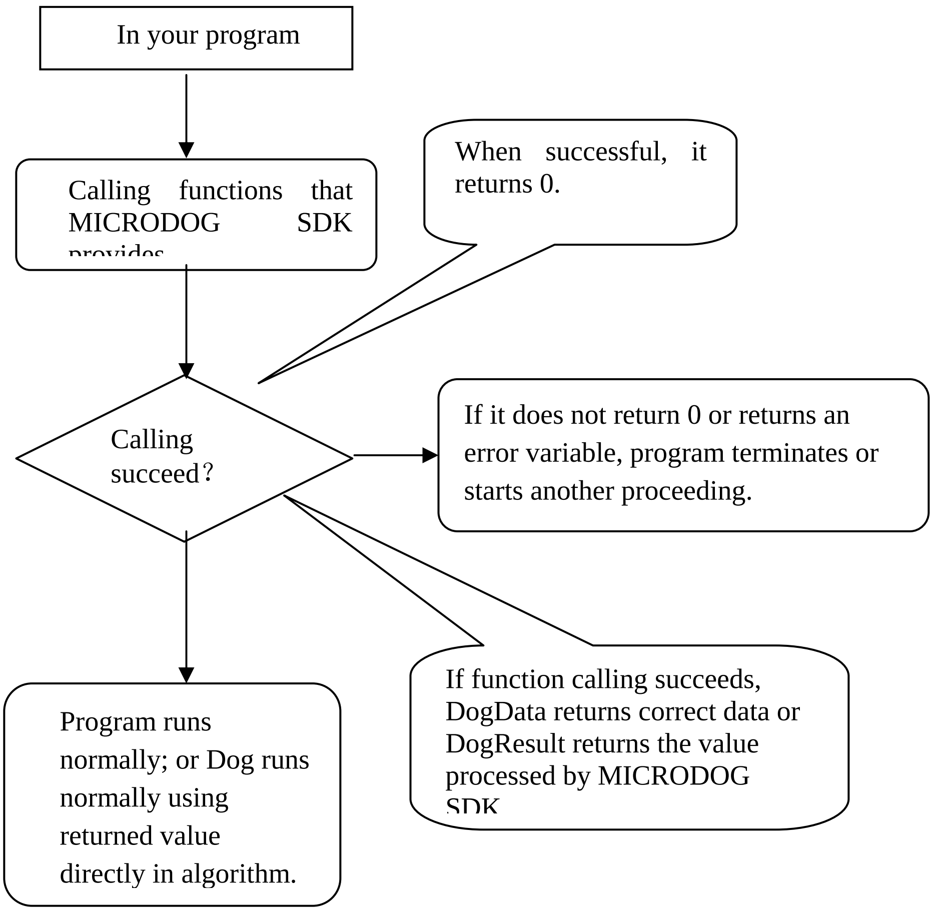
For files such as .APP, .FXP and .PRG files in Dbase, FoxBASE and FoxPro, MicroDog offers FOX.EXE Shell protection located in \DOS16\FOXTOOL. This tool also has functions such as verifying password, setting trial period and disabling share.

For more information, please refer to the Readme files. Please pay attention that when the protected application is running on WIN9X, you should copy VGS.VXD driver located in the same directory with Shell to \ SYSTEM. VGS. VXD should be provided along with your software to the end users.

In addition, MicroDog provides Shell protection for extended DOS application (EXP) \DOS32\EXPTOOL. For more information, please refer to the Readme file.

Chapter 5 Using API Calls

You can adopt API functions to set hidden and flexible software locks in your program to achieve great security. You may use API functions to make demo software that have a full control over expiration date and using times. This protection mode can also secure the special applications such as CPL and OCX that Shell protection tools cannot deal with.


In your program
Calling functions that MICRODOG SDK provides
When successful, it returns 0.
Calling succeed？
If it does not return 0 or returns an error variable, program terminates or starts another proceeding.
Program runs normally; or Dog runs normally using returned value directly in algorithm.
If function calling succeeds, DogData returns correct data or DogResult returns the value processed by MICRODOG SDK

The following global variable and function descriptions can apply to both Parallel Dog and USB Dog.

Global Variables

unsigned short DogAddr

The head address in the Hardware Dog memory for developers to write data into or read data from; ranges from 0 to 199.

unsigned short DogBytes

DogBytes refers to the number of bytes in *DogData*. When **DogBytes** is used in reading or writing operations, it ranges from 1 to 200, and the sum of **DogBytes** and *DogAddr* should not exceed 200. When **DogBytes** is used in converting operations, it ranges from 1 to 63.

void * DogData

A pointer variable. It points to the buffer used in reading, writing or converting operations.

unsigned long DogPassword

Developers use this variable to enter the password for the Hardware Dog. If a password entered is not identical to the password set on the Hardware Dog, the function will fail. The default password is 0. Developers can use DogEdt32.EXE to set their own passwords on Hardware Dog.

unsigned long DogResult

Stores the results of the conversion.

unsigned char DogCascade

The cascading number of Parallel Dog. Before accessing the Hardware Dog, you need to set the value of this variable, which ranges from 0 to 15. The default is 0. Win32 module supports cascade performance of Parallel Dog with the same Serial Number. For USB

Dog, this value must be 0.

unsigned long NewPassword

Only used to set new password.

Function Description

unsigned long far(*) DogCheck()

Input parameters: DogCascade.

Output parameters: None

Return value: Returns zero if successful; returns an error code if the function fails.

Purpose: Checks whether the Parallel Dog is attached to the parallel port or USB Dog is attached to the USB port.

*According to the different environments of developing tools, this must be defined in small mode.

unsigned long far DogConvert()

Input parameters: DogCascade,DogBytes,DogData.

Output parameters: DogResult.

Return Value: Returns zero if successful; returns an error code if the function fails

Purpose: This function sends the data pointed to by *DogData* to the Hardware Dog with the number of bytes being sent stored in *DogBytes*. The Dog converts the input data according to its algorithm (every Hardware Dog has its unique algorithm). The Dog then returns a converted result (a 32–bit integer) and stores it in *DogResult*. Developers can specify the algorithm. The last four bytes of the 200-byte memory provided by the Hardware Dog influence the type of algorithm. Byte 196 decides the algorithm, resulting in 256 different algorithms. Byte 197, 198 and 199 decide the algorithm descriptor, resulting in 16,777,216 different algorithm descriptors.

unsigned long far WriteDog()

Input parameters: DogCascade,DogAddr,DogBytes,DogData,DogPassword.

Output parameters: None

Return value: Returns zero if successful; returns an error code if the function fails

Purpose: Writes the data pointed to by *DogData* into the memory block beginning at the address *DogAddr* in the Hardware Dog. The number of written bytes is stored in *DogBytes*. Use of the function *WriteDog()* requires *DogPassword*.

unsigned long far ReadDog()

Input parameters: DogCascade,DogAddr, DogBytes, DogData,DogPassword.

Output parameters: DogData

Return value: Returns zero if successful; returns an error code if the function fails.

Purpose: Reads the value of the memory block beginning at the address *DogAddr* in the Hardware Dog, and then saves it in the buffer *DogData* points to. The number of bytes read is stored in *DogBytes*. The buffer should have enough space. Use of the function *ReadDog()* requires *DogPassword*.

unsigned long far GetCurrentNo()

Input parameters: DogCascade, DogData

Output parameters: DogData

Return Value: Returns zero if successful; returns an error code if the function fails

Purpose: Reads the Manufacture code of the Hardware Dog. Some of the Hardware Dogs may have the same ID, but each Hardware Dog has its unique Manufacture code. The Manufacture code can help developers manage user accounts.

unsigned long far DisableShare()

Input parameters: DogCascade.

Output parameters: none

Return value: Returns zero if successful; returns an error code if the function fails

Purpose: Prevents multiple computers from sharing one Hardware Dog through parallel ports. The parallel port-sharing device, usually provided by other suppliers, allows many computers to share one Parallel Dog. The default factory setting is share-enabled. You can use **DisableShare** to disable this function.

DisableShare only influences ReadDog, WriteDog and DogConvert. If you need to disable share, please select **Disable** before running those functions.

* Please pay attention not to call DisableShare for multiple times. One program can call DisableShare only once.

New added functions

Two new functions added to OBJ files in Win32 C module. You can use these functions to program or change the password and cascade.

unsigned long SetPassword(void)

Input parameters:

DogCascade: cascade Code

DogPassword: old password

NewPassword: new password

Output parameters: none

Return Value: Returns zero if successful; returns an error code if the function fails

Purpose: sets new passwords.

DWORD SetDogCascade(void)

Input parameters: DogCascade, DogPassword, DogData.

Output parameters: none

Return Value: Returns zero if successful; returns an error code if the function fails

Purpose: sets cascade code for Parallel. DogData is the address of new cascade code, and the code ranges from 0 to 15.

DOS16 and WIN16 modules can only operate USB Dog(UMA, UMB type) on Windows 98/ME/2000/XP but not USB Dog(UMC type). Operating Parallel Dog is not restricted to operating systems.

Protect Applications On DOS/Extended DOS

The Readme file located in the directory \MicroDog directory\DOS16\Readme (MicroDog default installation directory) list all modules and contents of computer languages supporting DOS applications, including BASIC, PASCAL, FORTRAN. CLIPPER, FOXPRO, C and ASM.

The Readme file located in the directory \MicroDog directory \DOS32\Readme (MicroDog default installation directory) lists all modules and contents of computer languages supporting Extended DOS applications, including HIGHT C, POWER, FORTRAN, WATCOM and NDP.

Each module of computer languages includes Readme file (compiling environment, variables and functions) BAT file (compiling environment, batch files), interface modules (OBJ) and example program calling interface programs.

Protect Applications On Windows 3X

The Readme file located in the directory \MicroDog directory \WIN3X\Readme (MicroDog default installation directory) list all modules and contents of computer languages supporting Window 3X applications, including VC, BC, DELPHI, VB, POWER BUILDER and FOXPRO FOR WINDOWS, among which VB, POWER BUILDER and FOXPRO FOR WINDOWS will call the DLL.

Each module of computer languages includes Readme file (compiling environment, variables and functions) BAT file (compiling environment, batch files), interface modules (OBJ) and example program calling MicroDog interface program. Please copy the correct DLL to SYSTEM before calling DLL and running the example program.

Protect Applications On Windows 9X/ME/NT/2000/XP

The Readme file located in the directory \MicroDog directory \WIN32\Readme (MicroDog default installation directory) list all modules and contents of computer languages supporting Windows 9X/NT/2000 applications, including VC, BC, C++Builder, DELPHI, JAVA, LAHEY FORTRAN, MS PowerStation FORTRAN, VB, POWER BUILDER and VISUAL and FOXPRO, among which VB, POWER BUILDER, VISUAL and FOXPRO are in the directory of Wind32DLL.

Each module of computer languages includes Readme file (compiling environment, variables and functions), interface modules (OBJ) and example programs calling MicroDog interface program. Please copy the correct DLL to SYSTEM before calling DLL and running the example program. 32-bit DLL adopts structure method to transmit parameters, and read or write Hardware Dog with different command code.

Protect Applications On Linux

On Linux based on Linux 2.2 and 2.4 versions, please copy files located in the installation directory to the designated directory in your Linux system. Refer to the document located in the directory \DRIVER to install the Hardware Dog's driver. MicroDog Linux driver program provides all operations to Hardware Dog on Linux. For more information, please refer to the example programs and Readme files.

Sample Programs

The following are examples using C language that calls API functions.

1) Define 8 API Functions and 7 Global Variables

8 API functions and 7 global variables must be defined in the beginning of the program.

```
extern unsigned long DogCheck(void);

extern unsigned long DogConvert(void);

extern unsigned long WriteDog(void);

extern unsigned long ReadDog(void);

extern unsigned long GetCurrentNo (void);

extern unsigned long DisableShare (void);

unsigned long SetPassword(void);

DWORD SetDogCascade(void);


short int DogAddr;

short int DogBytes;

unsigned long DogPassword;

unsigned long DogResult;

unsigned char DogCascade;
```

```
void * DogData;

unsigned long NewPassword
```

If you use C++ language, the function must be declared to define functions with extern "C" . For example,

```
extern "C" unsigned long far DogCheck(void);
```

2) API calls

The functions including DogCheck(), DogConvert(), WriteDog(), ReadDog(), GetCurrentNo(), DisableShare(), SetPassword(), SetDogCascade return a long integer.

They returns zero if successful; return error code if the functions fail. Please refer to "errcode-eng.txt" for error information.

Important: The following are simple examples of using API functions in software protection. It can only help you protect your application in a low capacity. These examples only provide programs with very limited protection. To maximize protection, refer to the following chapter "Making Optimal Use of MicroDog".

The function DogCheck() can help to find the corresponding Hardware Dog. The function returns zero if the Hardware Dog is found. Developers can use this function to test Hardware Dog.

For example, you can insert the statements below in the beginning part of the program. The program will exit if the Hardware Dog is not attached.

```
……

DogCascade = 0;
if(DogCheck())
{
```

```
                printf("No Hardware Dog found!");
                exit(2);
        }
        ……
        ……
```

The function DogConvert () can convert a string or other values. The converted result corresponds to input data. The converted result cannot be predicted, but you can use DogEdt32.EXE or other programs to get the converted result, and check it in the program.

For example, if the converted result of the input string "MicroDog" is "12345678," then:

```
……
        char ConvertBuff[10]={"MicroDog"};
        ……
        DogData = ConvertBuff;
        DogBytes = strlen(ConvertBuff);
        if(DogConvert())
        exit(2);
        else
        if(DogResult ! = 12345678)
        exit(2);
        ……
        ……
```

The function WriteDog() can write a string or other values to the Hardware Dog, and the function ReadDog() can read the data in the Hardware Dog. When using the two functions, be careful about the beginning address being read or written. For example, if you write 10 bytes of data to the memory beginning at address 0 in the Dog, next time you should write the data to the memory beginning at address 10 or higher. Otherwise, the previously written data will be destroyed.

When reading or writing a string, you should have DogData point to the buffer storing the string. When reading or writing an integer value, a long integer value or a floating number value, you should have DogData point to the address storing the value.

When reading data from Hardware Dog, make sure that the buffer *DogData* points to has enough space to store the read bytes. The function does not check the buffer size. **Be careful, it is a common mistake.**

Example 1: Write the string “MicroDog” to Address 10. Then, read the string and create a comparison for it.

```
……
char WriteBuff[10]={“MicroDog”};
char ReadBuff[10];
……
DogCascade = 0;
DogAddr = 10;
DogData = WriteBuff;
DogBytes = 8;
if(WriteDog())
exit(2);
……
……
memset(ReadBuff,0,10);    /* Clear ReadBuff */
DogAddr = 10;
DogData = ReadBuff;
DogBytes = 8;
if(ReadDog())
exit(2);
else
if(strncmp(ReadBuff,WriteBuff,8))
exit(2);
……
……
```

Example 2: Write the integer “1234” to Address 100. Then, read the integer and create a comparison for it.

```
……
int wint,rint;
……
wint = 1234;
DogCascade = 0;
DogAddr = 100;
DogData = &wint;
DogBytes = 2;        /* integer takes 2 bytes*/
if(WriteDog())
exit(2);
……
……
rint = 0;
DogAddr = 100;
DogData = &rint;
DogBytes = 2;
if(ReadDog())
exit(2);
else
if(rint !=1234)
exit(2);
……
   ……
```

The function GetCurrentNo() can read the Manufacture code of a Hardware Dog. Every Dog’s Manufacture code is unique. It may not be used in the protection process, but developers can use it to manage their user accounts. Each Hardware Dog’s Manufacture code is sequential and unique. Developers can collect the users’ registry information with the Manufacture code. The Manufacture code can be used to generate passwords for multi-module software package.

The following is an example of using the function *GetCurrentNo()*.

```
        ……
        unsigned long CurrentNo;
        ……
        DogCascade =0;
        DogData = &CurrentNo;
        if(GetCurrentNo())
        printf(“No MicroDog Found”);
        else
        printf(“Your register number is:%ld”,CurrentNo);
                              ……
……
```

DisableShare()is easy to use. It allows the disabling of a Dog’s sharing ability in the beginning of your program. The following is an example:

```
……
        DogCascade =0;
        if(DisableShare())
        printf(“No Hardware Dog found!”);
        else
        printf(“Disable share success!”);
        ……
```

SetPassword() is used to set new passwords.

```
        ……
        DogCascade=0;
        DogPassword=0;
        NewPassword=12345;
        if (!SetPassword())
                printf("SetPassword succeeded!\n");
        else
                printf("SetPassword failed!\n");
```

```
……
```

SetDogCascade() is used to set Cascade Code for Parallel Dog.

```
……
DogCascade=0;
DogPassword=0;
BYTE newcascade=1;
DogData=&newcascade;

if (!SetDogCascade())
        printf("SetDogCascade succeeded!\n");
else
        printf("SetDogCascade failed!\n")
……
```

3) Project Files

After you insert API calls and modify the source code of your program, you may add the OBJ file for C language to your project. After compiling and linking your program, you have your generated EXE file (executable files) protected now.

Since language modules (OBJ files) differ from one Hardware Dog to another, the program linking to a language module can only operate the module’s corresponding Dog. Therefore, when designing the software protection scheme you need not worry about distinguishing different Hardware Dog. On the other hand, if you purchase Dog with the same Serial Number, you can still identify each Dog by setting separate password for each Hardware Dog or storing separate values on it.

The API functions for other languages are similar to the ones in C language in terms of calling methods. You can refer to the README files and example programs in the directory of Parallel Dog system.

Chapter 6 Using Multiple Modules DLL

You can protect multiple modules and limit using times of modules with multiple module DLL. Moreover, multiple modules provide security protocol for DLL authentication, thus guarantee the security of calling DLL. Adopting the security protocol of DLL authentication, it ensures security of calling DLL.

Function Description

1. HRESULT RCDog_Open(DWORD Password, BYTE Cascade)

Input parameters:

DWORD Password: Read and write password of Hardware Dog

BYTE Cascade: Cascade Code

Purpose: Start Hardware Dog

Description: This function should be executed first. You must execute RCDog_Close()before starting another Hardware Dog. Currently you cannot manage multiple Hardware Dog simultaneously with this function.

2. void RCDog_Close()

Purpose: Close Hardware Dog

Description: Release relevant resource. You must execute this function before starting another Hardware Dog.

3. BOOL RCDog_CheckModule(WORD ModuleNo)

Input parameters:

WORD ModuleNo: Module number

Returned values: TRUE or FALSE

Function: Check if one module is valid or not.

Description: If the module is valid, the function returns TRUE; if not, it returns FALSE (Please note that if this module is valid and the number of using times is descending, the function returns FALSE when it is 0.)

4. HRESULT RCDog_GetModule (WORD ModuleNo, WORD *ret)

Input parameters:

WORD ModuleNo: Module number to be read

Returned value:

WORD *ret: The current number of using times of the module

Purpose: Read the current number of using times of a certain module.

Description: You can set the current number of using times only with DogEdt32.exe.

Note: This function requires that the restrictive number of the corresponding module is 1, or it returns error code.

5. HRESULT RCDog_DecModule (WORD ModuleNo)

Input parameters:

WORD ModuleNo: Module number whose using times will be reduced 1.

Purpose: Reduces 1 from the current using times of the module.

Description: If the current using time is 0, it will return an error code.

Note: This function requires that the corresponding module's decrement number and valid restrictive number is 1, or the function will return an error code.

6. HRESULT RCDog_GetSecureWord(DWORD WordNo, WORD *ret);

Input parameters:

DWORD WordNo: Restrictive number

WordNum = 1	Valid restrictive number
WordNum = 2	Decrement restrictive number

Returned value:

WORD *ret: the corresponding restrictive number

Purpose: read or write a certain restrictive number.

7. HRESULT RCDog_AuthDog(DWORD AriNo, BYTE * Random, DWORD RanLen, BYTE * ret)

Input parameters:

DWORD AriNo: Authentication algorithm used. Currently, MicroDog SDK can support only 128-bit RC6 algorithm. In this case, AriNo must be 1.

BYTE * Random: Input a random number.

DWORD RanLen: the length of Random. RanLen = 16 when AriNo = 1,
Returned value: BYTE * ret: The result of encrypting the random number with the authentication algorithm.

Purpose: Verifies the authentication key.

Description: Authentication key is a character string of random length defined by the developers. You can set its value by using DogEdt32.exe. Before saving the key on Hardware Dog, please convert it to a 16-byte HASH value with MD5 algorithm. This function verifies the authentication key. To guarantee the security of DLL, it is recommended to verify the authentication key before performing other functions of DLL.

8. DWORD RCDog_GetLastDogError()

Purpose: If this operation is not successful, Hardware Dog will return a 5-bit decimal error code.

Description: The old error code is not returned value because MicroDog SDK adopts HRESULT structure for convenient upgrade. If you need the error code returned by Hardware Dog, please call this function again to get the error code returned by Hardware Dog last time.

Multiple Module Memory Design

Some cells of the memory are used to edit and upgrade multiple modules. These are described in the following table:

Address	Description
0 - 127	User's memory.
128 - 129	Valid restrictive number of multiple modules.
130 - 131	Decrement restrictive number of multiple modules.
132 - 163	Current using times of 16 modules.
164 - 179	Authentication key.
180 - 195	Authentication key for remote upgrade.
196- 199	Algorithm descriptors.

Sample Programs

```
/*************************************************************

PROGRAM: sample
PURPOSE: This application demonstrates how to use dynamic-link library
DogMM.dll
Copyright(C) 2001 Rainbow China Co., Ltd. All Rights Reserved

*************************************************************/

#include "md5.h"         //md5 Algorithm head file
#include "RC6.h"            //RC6 Algorithm head file
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

```
HINSTANCE hDog=NULL;// DLL    handle
// The following are type definitions of API pointers.
typedef    HRESULT(* RCDOG_OPEN)(DWORD Password, BYTE Cascade);
typedef    void( * RCDOG_CLOSE)();
typedef    BOOL(* RCDOG_CHECKMODULE)(WORD ModuleNo);
typedef    HRESULT(* RCDOG_GETMODULE)(WORD ModuleNo, WORD *ret);
typedef    HRESULT(* RCDOG_DECMODULE)(WORD ModuleNo);
typedef    HRESULT(* RCDOG_AUTHDOG)(DWORD AriNo, BYTE * Random,
DWORD RanLen, BYTE * ret);
typedef    DWORD(* RCDOG_GETLASTDOGERROR)();
// The following are API declarations.
RCDOG_OPEN RCDog_Open=NULL;
RCDOG_CLOSE RCDog_Close=NULL;
RCDOG_CHECKMODULE RCDog_CheckModule=NULL;
RCDOG_GETMODULE RCDog_GetModule=NULL;
RCDOG_DECMODULE RCDog_DecModule=NULL;
RCDOG_AUTHDOG RCDog_AuthDog=NULL;
RCDOG_GETLASTDOGERROR RCDog_GetLastDogError=NULL;
//Test if DLL is valid.
//The input parameters are character strings and length to create authentication keys.
//The returned value is TRUE, and authentication succeeds.
//FALSE Authentication fails.    (Possible reasons: DLL has not been bound;
authentication character string is incorrect; DLL is replaced by hackers.)
BOOL CheckDll(PCHAR Key, int KeyLen);

    void main()
    {

    ULONG DogPassword;
    BYTE DogCascade;
    HRESULT hresult;
    WORD        ModuleNo;
    BOOL       bValid;
```

```
 WORD       ModuleCount;
char  DogKey[6]={'a','b','c','d','e','f'};
//The developer should change this value according to the string, which is used to
generate Dog Authorization Key.
 int        DogKeyLen=6;//DogKeyLen is the real size of DogKey;
 printf("\nThe sample use multiple module to encrypt the application.\n");
 printf("\nCopyright (C) 2001 Rainbow China Co., Ltd.\n\n");
 hDog=LoadLibrary(TEXT("DogMM.dll"));//Load DLL
 if (hDog!=NULL)
 {
     //API functions represented by DLL
             RCDog_Open=(RCDOG_OPEN)GetProcAddress(hDog,"RCDog_Open
       ");
             RCDog_Close =
                 (RCDOG_CLOSE)GetProcAddress(hDog,"RCDog_Close");
             RCDog_CheckModule=(RCDOG_CHECKMODULE)GetProcAddress
                 (hDog,"RCDog_CheckModule");
             RCDog_GetModule=(RCDOG_GETMODULE)GetProcAddress(hDog
                 ,"RCDog_GetModule");
             RCDog_DecModule=(RCDOG_DECMODULE)GetProcAddress(hDo
                 g,"RCDog_DecModule");
             RCDog_AuthDog=(RCDOG_AUTHDOG)GetProcAddress(hDog,"RC
                 Dog_AuthDog");
             RCDog_GetLastDogError =
                 (RCDOG_GETLASTDOGERROR)GetProcAddress(hDog,"RCDo
                 g_GetLastDogError");

       //Take the first step RCDog_Open
       DogPassword=0;
 //Should be same with the password recorded on Hardware Dog
       DogCascade=0;
 // Should be same with the password recorded on Hardware Dog
       hresult = RCDog_Open(DogPassword,DogCascade);
 if (hresult==S_OK)
 {
        printf("Open dog succeeded!\n");
 //If you need to authenticate DLL, call CheckDll(). This function can
authenticate DLL. The function will return FALSE if the authentication fails.
        if (CheckDll(DogKey,DogKeyLen)== TRUE)
        //Authentication succeeds.
        {
               printf("The Dynamic Library authority succeed!\n");
               for (ModuleNo=0;ModuleNo<16;ModuleNo++)
               {
                      bValid=RCDog_CheckModule(ModuleNo);
                      if (bValid==TRUE)
                      {
                             printf("The %d module is valid!\n",ModuleNo);
                             //To Do
                             //the developer can add the module call here;
                            //Take the following step if there is a limit to the using
                            times.
                            hresult=RCDog_GetModule(ModuleNo,&ModuleCount;
                             if (hresult==S_OK&&ModuleCount>0)
                             {
                                    //To Do
                                    //the developer can add the module call here;
                                  //Take the following step if you need to decrement
                                     the using times.
                                    hresult=RCDog_DecModule(ModuleNo);
                                    if (hresult == S_OK)
                                    {
                                                   printf("The %d module count has been
                                                   decreased 1 !\n",ModuleNo);
                                    }
                                    else
```

```
                                {
                                        printf("The %d module count    decrease
                                        failed !\n",ModuleNo);
                                }
                        }
                        else
                        {
                                printf("The %d module can not be
                                used!\n",ModuleNo);
                        }

                }
                else
                {
                        printf("The %d module is invalid!\n",ModuleNo);
                }
            }
        }
        else//Authentication fails.
        {
            printf("The Dynamic Library authority fail,the Dynamic Library
            must be taken place!\n");
        }
        RCDog_Close();//Finish operations and close Hardware Dog.
    }
    else
    {
        printf("Open dog failed!\n");
    }
    FreeLibrary(hDog);//Release DLL

}
else//DLL does not exist or is not located in the directory of .EXE files
```

```
{
    printf("Can't find DogMM.dll\n");
}
}
// Test if DLL is valid.
//The input parameters are character strings and length to create authentication
   key.
//If the returned value is TRUE, the operation is successful.
//FALSE Authentication fails.    (Possible reasons: DLL has not been bound;
   authentication character string is incorrect; DLL is replaced by hackers.)
BOOL CheckDll(PCHAR Key,int KeyLen)
{
//Get md5 key with the key
//Create the random number random
//Authenticate DLL, and return encrypted random number random1
//Get rc6key with md5 key
//The returned random number is decrypted with rc6key and the new random
number random2,
//Compare random2 with random.    If the two random numbers are same, DLL
authentication succeeds.
MD5    md5object;
char Md5Key[12]={'1','3','2','4','3','5','4','6','5','7','6','8'};
// The secret key developers of md5 cannot modify this password.
UCHAR       sign[16]; // MD5 Signature string
RC6_KEY RC6Key;
int RandomLen=16;
HRESULT hresult;
unsigned char Random[16];
unsigned char Random1[16];
unsigned char Random2[16];

memset(sign,0,16);
```

```
//Create random character strings
srand((unsigned)time(NULL));
for (int i=0;i<8;i++)
{
        *(int *)&Random[2*i]=rand();
}
//Authenticate DLL
hresult=RCDog_AuthDog(1,Random,RandomLen,Random1);
if (hresult!=S_OK)
{
        return FALSE;
}
md5object.SetKey(Md5Key,12);
md5object.Encrypt((PCHAR)Key,KeyLen,sign);//Get md5 signature string
GenerateKey(sign, &RC6Key);//Get decryption key for rc6 algorithm
RC6DecryptData(&RC6Key,Random1,16,Random2);//Decrypt random numbers
for (int j=0;j<RandomLen;j++)
{
        if (Random[j]!=Random2[j])//Compare the two numbers
        {
                return FALSE;
        }
}
return TRUE;

}
```

Chapter 7 Making Optimal Use of MicroDog

In MicroDog SDK, various protection strategies are adopted in the Hardware Dog and in the supporting software. However, owning an MicroDog SDK does not mean that it is unnecessary to scheme software protection. If we liken MicroDog SDK to a good lock, it is meaningless to use a good lock for a broken door because a thief can just break through the door without touching the lock.

In a software product protected by MicroDog SDK, the most vulnerable part lies in its software locks. A software lock is an API call to the Hardware Dog. The time and cost required for a skilled pirate to break your scheme is directly related to the number and complexity of the locks you place in your software. Theoretically, all protection systems can be broken. However, if the cost for breaking the system is too high, no pirates will try it. You may maximize this cost by enhancing sophistication of the software locks or increasing software locks. We need your cooperation when we increase the software locks.

You may decide the protection scale according to specific needs. You should design your protection strategy carefully even if you need only simple protection solutions. You need to keep in mind that although more software locks may improve the software protection, they do affect the memory space as well as the execution speed of the protected software. Please take every factor into consideration when designing your strategy.

The following strategies can apply to both Parallel Dog and USB Dog. USB Dog can be supported on Windows 98/ME/2000/XP and only support Win32 module.

- **Inserting Extra Data** **Security Rating: 1**

You may insert frequent "garbage" data between input values and evaluated values, making the relation between two functions confusing to a hacker. For a global value, you can simply evaluate the returned value in another portion of code:

```
RetCode = DogCheck();
...   //   Inserting extra data
if ( RetCode != 0 )
{
...
}
```

- **Verifying Returned Value Flexibly** **Security Rating: 2**

You can verify one returned valued in different parts of the code to confuse the hacker.

```
   RetCode = ReadDog();
   If ( RetCode != 0 )
   {
        ...
   }
...   // Inserting extra data
  If ( RetCode != 0 )
   {
...
   }
...   // Inserting extra data
   If ( RetCode != 0 )
   {
       ...
    }
```

- **Inputting Values Frequently** **Security Rating:3**

You can input a new value to a returned one frequently, or even make a complicated circle in this way. In the following example, the developer input some scrambling value before using RetCode.

```
for ( i = RetCode; i < RetCode + 1000; i++ )
{
   if ( random() % 3 )
   RetCode = i ^ i + 157;
}
RetCode = DogCheck();
```

- **Inputting Multiple Values** **Security Rating: 3**

You can input multiple values into the returned ones to confront an attempt to track the code. Any returned error value will exit the application abnormally.

```
RetCode = ReadDog();
for ( i = 0; i < 100; i ++ )
{
     a [i] = RetCode ;
  }
for ( i = 0; i < 100; i ++ )
{
     b [i] = a [i] ;
}
```

- **Using Returned Values** **Security Rating: 4**

To use returned values will combine your application and protection solution more closely. In the following example, an error returned value will lead to a wrong result

```
RetCode = DogCheck ();
PI = RetCode + 3.14159;
```

- **Using Constants** **Security Rating: 6**

Input some constants in Hardware Dog such as path name, file name and some other frequently used data. In the following example, the string "AUTO" is placed in 10-14 cells, and the number 1 is placed in cell 15.

```
DogBytes = 5;
DogAddr = 10;
RetCode = ReadDog();
Strcpy ( Path, "C:\" );
Strcat ( Path, DogData );
Strcat ( Path, "EXEC.BAT" );
...
DogBytes = 1;
DogAddr = 15;
RetCode = ReadDog ();
for ( i = 0; i < 100; i = i + *DogData )
{
        ...
}
```

- **Using Converted Result Frequently** **Security Rating: 10-20**

The converted result is a predictable returned value. The result varies to input values. You can convert the data frequently and use the different results.

- **Executing the Commands Randomly** **Security Rating: 10**

A hacker tends to run applications continuously to track the code of an application. Some random data will prevent the hacker from locating the encryption lock easily. You can make the system verify the returned value randomly, input the value randomly or execute the command randomly.

```
if ( Random ( 10 ) = =5 )
```

```
{
     RetCode = DogCheck ();
     ...
}
if ( Random ( 10 ) == 7 )
{
     RetCode = ReadDog ();
     ...
}
```

Some sophisticated hackers may find out the random of the program. You can program your own generator for random data. You can adopt random data such as the system time when initializing your generator.

- **Reading Data Randomly** **Security Rating: 15**

You can read data from Hardware Dog randomly:

```
DogAddr = Random ( 200 );
DogBytes = Random ( 200 – DogAddr );
RetCode = ReadDog();
...
```

- **Writing Data Randomly** **Security Rating: 20**

You can write some extra data into Hardware Dog randomly. Please make sure that the useful data will not be overwritten. You can appoint some cells in memory for the extra data particularly. In the following example, cells from 80 to 90 cannot be used.

```
DogAddr = 80 + Random(20);
DogBytes = 20 – Random ( 100 – DogAddr );
DogData[1] = DogAddr;
DogData[2] = DogBytes;
RetCode = WriteDog();
```

- **Writing and Reading Data Randomly** **Security Rating: 35**

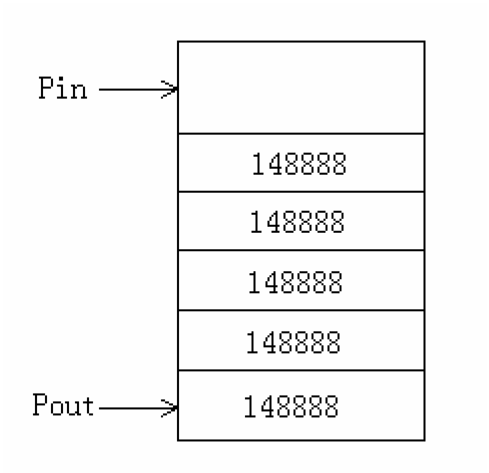
You may insert some other commands between writing data and reading data so that the hacker will not find out the logical relationship easily.

- **Multi-thread Processing** **Security Rating: 20-150**

Multi-thread processing can obstruct an advanced decryption tool. The basic guideline for this technique is to operate Hardware Dog in one processing and to verify the Hardware Dog in another.

- **Queue** **Security Rating: 40-60**

You can make a more complicated queue to manage the returned values and results. The execution of the whole program relies on the queue. You can adopt Multi-thread processing strategy in the queue. The following example shows how to manage the returned values with a queue.



The initialized queue in the above chart contains 6 cells and the initialized value is 148888. When the returned value is not 0, input 205429 into Pin, and the queue will move one cell. It is correct when the valued verified at Pout is 148888, and 205429 is an illegal value.

- **Using Returned Result Directly** **Security Rating: 100-150**

In the beginning of the program, you can use Encrypt () to encrypt important parameters and variables, and use Decrypt () to decrypt them when you need to call those parameters and variables. You can use the returned result directly without verifying the returned value of Decrypt(). The decrypted value is incorrect without Hardware Dog, thus the program cannot run normally.

***Conclusion**:*These protection strategies can help you protect your software effectively, and to some extent they are relied on how you develop your own program. You can adopt the simple strategies flexibly to achieve higher security. Advanced strategies are hard to realize and make it more difficult to call the functions. Usually it takes 50 - 200 millisecond operating MicroDog will cause a. It is strongly recommended to adopt AS technology in your protection strategy. AS technology, in addition to inserting API calls into your program for protection, provides a shell protection tool (for executable files) to add one more level of protection (called "Shell") to your program. The two levels of protection are correlated in shell protection. The API calls in your program will not be functional without the correlated "shell." In other words, if the cracker decrypts the "shell," the API calls used in the program will not work correctly, and the cracking cannot continue.

Perform Other Functions:

1) One Hardware Dog (Parallel Dog or USB Dog) for multiple module software or multiple software packages.

The bytes of the memory of the Hardware Dog can be allocated to different modules or software packages. Alternatively, the same data can apply to different modules or software packages. In the programs where conversion functions are adopted, the last byte in the memory determines the conversion algorithm. Please note that when using one Hardware Dog for multiple modules or software packages, the application protected by Shell can only be set not to verify the password, or passwords are same.

If you do not want to program your own program, you can perform functions RC UMH provides particularly for multiple modules.

2) Setting using times or service period or disable certain modules

You can use some of the cells in the memory as a using time counter. Those cells can be initialized as using times. The program will read or write those cells and check if it is 0 to determine if the application can run normally. And then (or when the application exits) the using times will be reduced 1. The principle of limiting the service period is similar.

You can easily set using times or service period with multiple module functions.

3) Remote upgrade

Operation to the protected application relies on data, reading/writing passwords, converting algorithm on Hardware Dog. All needs for upgrade are satisfied when you provide end users with updated data, read/write password and conversion algorithm, and initialize the flag byte. The code for upgrading is also added in the upgrade program.

You can use the same method to upgrade the demo software, increase using times or service time, or activate some functions or using times.

For more information on multiple module upgrade, please refer to the sample programs located in \win32\RemoteUpgrade.

4) Verifying presence of Hardware Dog during installation

The installation can verify presence of Hardware Dog by calling modules such as WIN32DLL or running installation program of Shell protection. Please install the installation driver first on WINNT/2000/XP.

5) Using high strength DLL to protect applications

The authentication ensures the security of calling DLL. The main difference between high strength DLL and normal DLL is that the high strength DLL has a function of password verification. Please refer to the sample programs located in \win32\win32dll\highdll.

Chapter 8 Distributing Your Software

When you distribute your software protected by MicroDog SDK, you should provide the following items to end-users:

- ◆ Your protected applications and DLL of MicroDog SDK (if called in your applications);
- ◆ The corresponding Hardware Dog (Parallel Dog or USB Dog);
- ◆ Hardware Dog’s driver;
- ◆ Use Direction for Software, including how to install Hardware Dog’s driver;
- ◆ Customer End tools for remote upgrade if multiple module protection is used

The MicroDog Hardware Dog’s driver is located in the installation directory \DRIVER. The applications use the parallel port or USB port through the drivers, thus ensures perfect compatibility. In most cases, all protected applications on WIN9X/ME/NT/2000/XP require MicroDog drivers.

Before running the DOS applications protected by Shell tools on WIN 9X, make sure to copy VGS.VXD located in the same directory of Shell to the directory of SYSTEM, and offer it to end users.

Select one of the following methods to install MicroDog driver.

- **Use installation program of MicroDog drivers directly**

MicroDogInstdrv.exe is located in \DRIVER. The end user must run

MicroDogInstdrv.exe to install MicroDog drivers. MicroDogInstdrv.exe can automatically install the corresponding driver according to the operating system. On WINNT/2000, MicroDogInstdrv.exe will inquire whether or not the 16-bit drivers have been installed. If your software is not 32-bit (WIN9X/NT/2000/XP applications) and needs to run on WINNT/2000/XP, the end users must install 16-bit drivers first.

Rainbow China’s logo will be displayed in the installation program. By adding the parameters **(‘i’ and ‘s’)** to MicroDogInstdrv.exe in command line, namely, “MicroDogInstdrv.exe /i/s”, Rainbow China’s logo will disappear.

- **Make your own installation program to install MicroDog driver**

Your software installation program will install MicroDog drivers by calling RCMicroDogSetup.dll located in the installation directory of \DRIVER. RCMicroDogSetup.dll has the same function as that of MicroDogInstdrv.exe. Please read the Readme file in the directory for more information. The example program of calling RCMicroDogSetup.dll is located in DRIVER\.

Chapter 9 Specifications

Parallel Dog:

Dimension:	55x45x15mm
EEPROM memory capacity:	200 bytes
Minimum working voltage:	2.5V
Maximum average working current:	1mA

(Testing environment: voltage is 5V, reading and writing Hardware Dog continuously.)

Working temperature:	0~55^{o}C
Allowable writing	1,000,000
Allowable reading:	Limitless
Power-off Holding Period:	100 years
Port Pins Used:	1~9, 13, 14, 16

USB Dog:

Dimension:	54x16x8mm
EEPROM memory capacity:	200 bytes
Minimum working voltage:	2.5V
Maximum average working current:	1mA

(Testing environment: voltage is 5V, reading and writing Hardware Dog continuously.)

Working temperature:	0~55^{o}C
Allowable writing	1,000,000
Allowable reading:	Limitless
Power-off Holding Period:	100 years
Communication speed:	1.5 Mbps