Developer's Guide

Dinkey Dongle

Microcosm Limited

1, Eastfield Road
Westbury-on-Trym
Bristol
BS9 4AD
England

E-mail: dinkey@microcosm.co.uk
Web: http://www.microcosm.co.uk

Copyright © 1996-2006 Microcosm Limited for both this publication and the Dinkey Dongle system. All rights are reserved.

No part of this publication nor the computer programs to which it refers may be reproduced, transcribed, transmitted, stored in a retrieval system, or translated into any language, in any form or by any means mechanical, manual, electronic, magnetic, chemical, optical, or otherwise, except so far as described in this manual or accompanying Licence Agreement, without the prior written permission of Microcosm Limited. Successful prosecution for unauthorised reproduction can result in imprisonment and/or fines.

Dinkey Dongle, Dinkey 1, Dinkey 1S, Dinkey 2 and DinkeyNet are trade marks belonging to Microcosm Limited. MSDOS is a trade mark of Microsoft Corporation. IBM and PCDOS are trade marks of International Business Machines.

Intended for SDK version 4.1 (or higher) and setupdrv version 3.4 (or higher).

Manual printed March 2006.

DISCLAIMER OF WARRANTY

Microcosm Limited makes no warranties or representations with respect to the contents or performance of the Dinkey Dongle package or this manual. It particularly disclaims any implied warranty of fitness for any particular purpose.

The Dinkey Dongle system is sold "as is" with all faults. Any claims made by sales literature or salespersons do not constitute warranties. Because of the diversity of hardware, software and conditions under which the system may be used, Microcosm cannot make any warranty of fitness for a particular purpose. The entire risk of using the product must be assumed by the user. Accordingly, the user is recommended to thoroughly test the product before relying on it. In any event, any liability of Microcosm is limited exclusively to refund of the purchase price of the product.

It is the user's responsibility to ensure that they conform to any laws concerning the changes and restrictions they are permitted to make to an end user's computer and its software.

Microcosm Limited reserves the rights to revise and to make changes to the software and/or the hardware and/or this manual without incurring any obligation to notify any person of such changes and revisions.

Contents

Introduction

Overview

A Dinkey Dongle is a small piece of hardware which is part of a sophisticated system designed to protect your software. They are available in models that fit conveniently into either the parallel or USB ports of your computer. Each Dinkey Dongle has a unique serial number which can be locked to your software using the DDAdd program. Then, each time you run your protected program, it will check for the presence of the dongle. If the right dongle is found, the program will be executed, otherwise it will not run.

There are four models of Dinkey Dongle: Dinkey 1, Dinkey 1S, Dinkey 2 and DinkeyNet. All provide a high level of protection.

Dinkey 1 allows you to protect your software. It has the advantage of low cost but you have to individually lock your software to each dongle before you send it out. This range has now been discontinued.

Dinkey 1S is a variant of Dinkey 1 that contains a Software Developer's Serial Number (SDSN) which allows you to bulk copy your protected programs more effectively. This is particularly suited to situations where you may be shipping large numbers of dongles. There is no need to program these dongles.

Dinkey 2 not only allows you to protect your software, but also to control it and store your own secure data. For example, one of the many things that you could do is to put an expiry date on one of your programs. You also have the facility to alter these protection parameters remotely. So, for example, at a later date you could change the expiry date on your program just by issuing a code over the telephone to your customer.

DinkeyNet has all the features of Dinkey 2 but allows you to use one dongle per network instead of one dongle per workstation. In addition, you can limit the number of simultaneous network users of your program. For the remainder of this manual for 'Dinkey 2' please read 'Dinkey 2 and DinkeyNet'.

Every model is available in both parallel and USB forms, except for Dinkey 1 which is just a parallel port dongle.

The software that is supplied with the Dinkey Dongle system is comprised of the following:

- DDAdd - a program that adds protection to your software and programs the dongle.
- DDRemote - a program that generates codes for the remote changing of protection parameters.
- DDChange - a program that accepts the update code that DDRemote generates and modifies the dongle accordingly.
- DDLook - a program that searches all the parallel and USB ports for dongles and displays their contents.
- DDManual - a PDF manual showing how to use the Dinkey Dongle system.
- DINKEY.HLP - Windows help file version of the manual.
- Various runtime modules that contain protection-checking code for the OS you have installed to (in the MODULES directory).
- Sample source code to interrogate the dongle (in the SAMPLES directory).
- Special software for the DinkeyNet dongle (in the NET directory).
- Device drivers for both the parallel and USB dongles for the various different operating systems and an installation program (in the DRIVERS directory).

System Requirements

In order to run DDAdd and DDRemote you need to be running at least Windows 95 or Windows NT4. There are no versions of these programs for Mac or Linux.

Your protected program will execute some of our code. Under Windows this code will run on any IBM PC compatible with at least a 386 processor running under the MSDOS (version 2.0 or later), PCDOS (version 2.0 or later), DRDOS, Novell DOS, Concurrent DOS, MultiUser DOS, Windows (version 3.0 or later), Windows 95/98/Me , Windows 2000/XP/2003, Windows NT (version 3.51 or later) and 64-bit versions of these Operating Systems.

For Mac runtime code to work you will OS X 10.0 or higher, but 10.3 or higher is recommended.

The code for DinkeyNet will run on any network that uses MSDOS (version 4.0 or later), Windows 3.x, Windows 95/98/Me, Windows 2000/XP/2003 and Windows NT (version 3.51 or later) and 64-bit versions of these operating systems.

Please note that the USB dongle will not work under Operating Systems that do not support the USB protocol. e.g. DOS, Win3.11, Windows 95. However, it will work

works Windows NT4. Also, the parallel port dongle is not supported under 64-bit operating systems.

Protecting your program with the Shell or OBJ method will add 25-50K bytes to its length. If you are using a Dinkey 2 or DinkeyNet for protection, and you want to remotely change the user's protection parameters, you will also need to distribute DDChange or one of its variants.

The parallel port dongle requires a driver for all OS except DOS, Win3.11, Windows 95/98/Me.and the USB dongle requires a driver for any of the compatible operating systems it is installed. In this case you must supply the necessary drivers (see Drivers chapter).

Features List

Security

- Innovative compact design offers outstanding protection.
- Strong anti-debug code.
- Advanced encryption techniques used.
- Each one of your protected programs can be automatically encrypted in a unique way.
- Each low-level port access of the dongle is different.

Ease of Use

- Little or no changes to source code required.
- Cross-platform for usb dongle: Windows, Mac, Linux.
- Many programs can be protected by one dongle (up to 17 for Dineky2/DinkeyNet).
- You can protect upgrades to your software so that they will work with the Dinkey Dongles that the users already have. There is no need to ask your customer to send in their dongle back for modification.
- Full sample code in many languages.

Reliability

- Parallel: works with all IBM PC compatibles and all printers. USB: design of USB port guarantees 100% compatibility.
- Dinkey 2 is guaranteed for at least 10,000,000 accesses, and has data retention of at least 200 years. Dinkey 1 has theoretically unlimited use.

Control (Dinkey 2 only)

- Control over which part of your program is used.
- Prevent your software from running after a specified number of days from the date of first use or from a fixed expiry date.
- Limit the number of times your software can be run.
- Store up to 432 bytes in the dongle's data area.

Remote Parameter Changing (Dinkey 2 only)

You can change all the control options remotely by giving the customer a secure code over the telephone. Each code can only be used once but you can produce as many codes as you like. It is a powerful feature that allows great flexibility but is not offered by many dongle manufactures. Options include:

- Setting or removing an expiry date for any of your programs.
- Changing the number of times your program can be run (or allowing your program to run an unlimited number of times).
- Enabling new programs to be protected by the dongle.
- Changing the data area

How This Manual Is Organised

After installing the Dinkey Dongle software you can then start protecting your software. The following is a summary of the next few chapters and how you can use them to help you protect your software.

The **Quick Tour** leads you through the process of protecting a simple program and modifying some of the protection parameters. It is recommended that you read this chapter as it will make you familiar with the Dinkey Dongle software and how it operates. If you feel confident you can omit this chapter and proceed to the next chapters that describe each step in more detail.

Please note that reading this chapter is by no means a replacement for reading the rest of the manual.

You *must* read the chapters on **Protection Method**, **Modifying your Code** (if applicable) and **Adding Protection (DDAdd)** to successfully understand how to protect your software. You must also understand how to install the necessary device drivers onto the customer's machine, especially if you are using the USB dongle. Read the chapter entitled **Driver Installation** for information on this.

The chapter on **Remote Changing of Parameters** describes how you can modify data in the dongle remotely. It only applies to Dinkey 2 and DinkeyNet.

The other chapters are more informative and can be used as a reference book. You need to read the chapters that are relevant to you. In particular, read **DinkeyNet** if you intended to use the DinkeyNet dongle.

Please remember that the security level of your protected application depends on how you implement the Dinkey Dongle system. To achieve the best levels of security you should not just copy the sample code but fully understand how the product works and adapt the sample code using some of the methods suggested in the **Increasing Your Protection** chapter.

Installation

Installing Dinkey Dongle

Windows

We recommend that you install the Dinkey Dongle software onto your hard disk and keep the master disk somewhere safe. Please make a note of your software developer's serial number (SDSN) that is written on the master disk as you will need it in any future correspondence you may have with us.

To install the Dinkey Dongle software you will need the following:

- Windows 95, 98, Me, 2000, XP or Windows NT
- Approx 15 MB available hard disk space

To install the Dinkey Dongle software onto your computer:

- Insert the Microcosm Dinkey Dongle CD into your CD drive.
- The setup program should start automatically (it may take a few moments before you can see anything happening).

If the setup program does not start automatically you can run it manually:

- Click the **Start** button and then choose **Run…**
- When the Run dialog box appears type **d:\setup** (where d is your CDROM drive) and then click the **OK** button.

Alternatively, if we have e-mailed you the software then you should extract all the files to a temporary directory and run **setup** to install.

The setup program will ask you which path you want the software to be installed to. It will then copy the relevant files from the CD to this path and install the necessary device drivers for the Operating System you are running. It will overwrite any previous Dinkey Dongle installation in this directory.

Mac

To install the Dinkey Dongle software on your Mac computer:

1. Uncompress the file in the Mac subdirectory on the masterdisk. You can do this by double-clicking the icon or using StuffIt Expander.
2. Install the software by double-clicking the package. This will install software to the Applications/Dinkey Dongle folder and install the USB drivers.

Linux

To install the Dinkey Dongle software on your Linux computer:

1. Copy across all the files in the Linux subdirectory to your hard disk.
2. Read the drivers chapter (see "Installing the Linux drivers" on page 10-5) to learn how to compile drivers Linux and install those drivers.

Uninstalling Dinkey Dongle

Windows

To uninstall the Dinkey Dongle software from your computer:

1. Click the **Start** button, select **Settings**, and click **Control Panel**.
2. Open the **Add/Remove Programs** control.
3. In the list of programs that appear, select Dinkey Dongle then click the **Add/Remove** button.

Mac

To uninstall the Dinkey Dongle software from your Mac computer:

1. Uninstall the software by deleting the Applications/Dinkey Dongle folder.

1. Uninstall the drivers by removing the ddUsbKeyDriver.kext file from the System | Library | Extensions folder

Linux

To uninstall the Dinkey Dongle software from your Linux machine:

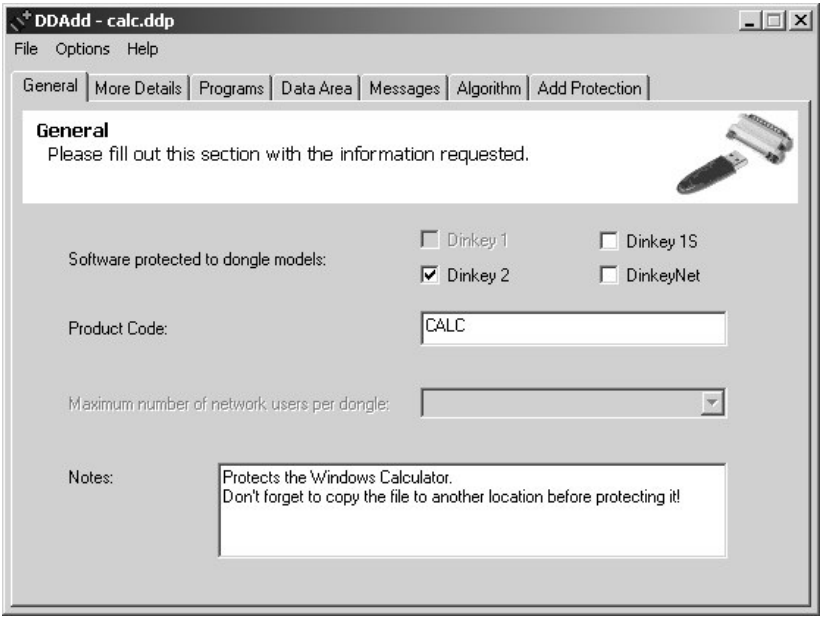
1. Uninstall the software by deleting it from your hard disk.
2. Read the drivers chapter (see "Uninstalling the Linux drivers" on page 10-5) to learn how to uninstall the Linux drivers.

Quick Tour

Protecting a Sample Program

This section will guide you through protecting a simple program. You can choose any program you like for this example, but for simplicity's sake we will choose CALC.EXE, the Windows calculator. This is usually located in the WINDOWS directory (for Windows 95/98/Me) or in WinNT\System32 (for Windows NT/2000/XP.

To protect this file you must run the DDAdd program. Do this now by clicking on Start | Programs | Dinkey Dongle | ddadd. It will display the main DDAdd screen. Now click on File | Open and open the file calc.ddp which is located in the installation directory. Your screen should now look something like this:



DDAdd screen with calc.ddp file loaded

By opening the calc.ddp file we have entered all the protection parameters into DDAdd. If the dongle you have is not a Dinkey 2, please modify the Dongle Model entry so that it is the same as the model of the dongle that you have. Attach this dongle to the parallel port of your machine. If you are unsure which dongle model you have then run DDLook by clicking Start | Programs | Dinkey Dongle | ddlook. This program displays the contents of the dongle(s) attached to the parallel port and USB ports.

If you are using a DinkeyNet dongle then you will also need to 'setup' the DinkeyNet on the network. For full details please read "Using DinkeyNet" on page 11-1.

When you protect CALC it will be modified so you need to make a copy of it before it is protected. You can protect CALC in any directory, but in the calc.ddp file we have chosen the C:\Program Files\Dinkey directory. (If you want to use another directory then you must choose the *Programs* tab and then click the *Modify Selected* button. In the *Full Pathname* edit box, type in the full path of the directory including the drive and the filename calc.exe (or use the browse button). Then click *OK* followed by another *OK)*.

You are now ready to protect CALC. Do this by choosing the Add Protection tab and clicking the *Add Protection Now* button. After a few seconds a message will appear indicating that protection has been added to the CALC program (and to the dongle in the case of a Dinkey 2). Click on *OK* and exit DDAdd.

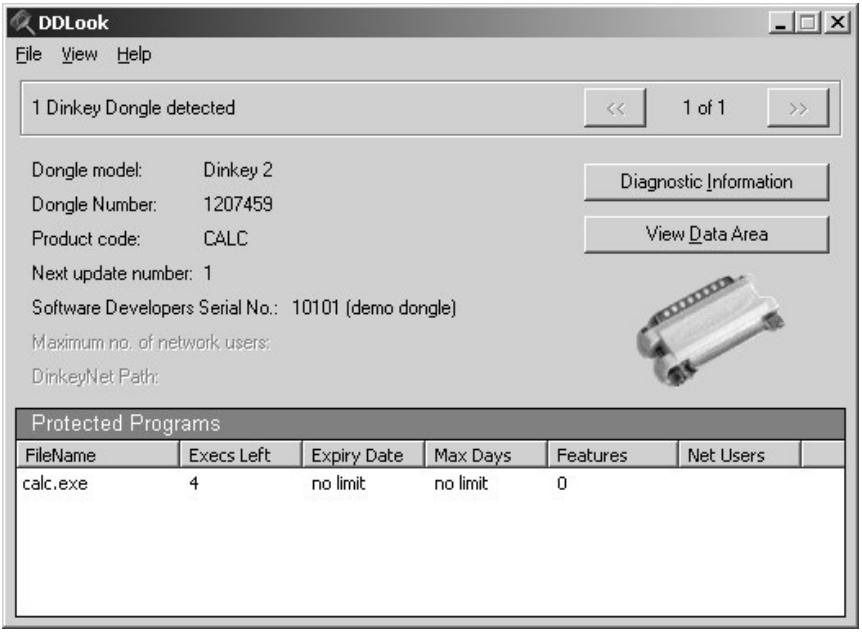
CALC is now protected - see how easy it is! Now try running CALC (the protected one in the C:\Program Files\Dinkey directory!) It will display the message *Successfully interrogated the dongle. Click OK to continue*. When you click on *OK* the CALC program will execute as before. Now exit the CALC program.

Remove the dongle from your machine and try running CALC again. It will display an error number followed by the message *The dongle is not found. Please insert the dongle into a parallel or USB port.* When you press *OK* the program will terminate.

You have added protection to a simple program. To find out how to protect *your* program you need to read "Protection Method" on page 4-1. If you have a Dinkey 2 then you may like to read the next section which shows you how to modify the protection parameters. then you can continue the Quick Tour and learn how to modify the protection parameters.

Modifying Protection Parameters

If the CALC program was locked to a Dinkey 2 dongle then it was initially limited to 5 executions. To find out how many executions you have left please attach the dongle to your machine again and run DDLook. It should display a screen similar to this:



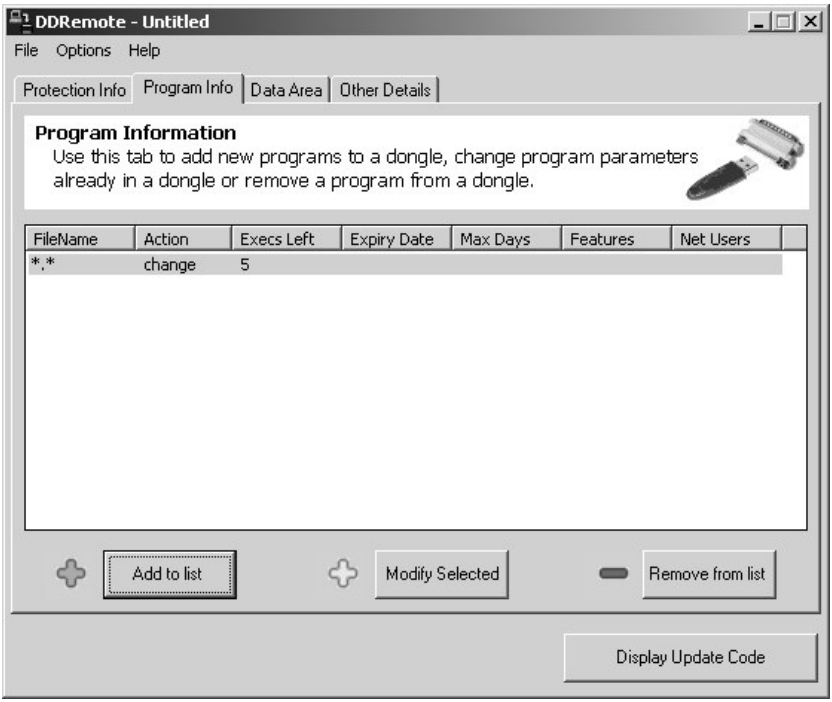
DDLook screen displaying the contents of the protected dongle

The figure under the *Execs Left* title tells you how many more times you can successfully run the CALC program. Please repeatedly run the CALC program until it comes up with the following message: *Dinkey Dongle error no. 024. The product has expired.*

You have now successfully run the CALC program your allotted five times and you cannot run it again. The value of Execs Left in the dongle is 0. However, you can change this value by running the DDRemote and DDChange programs.

Run DDRemote by clicking Start | Programs | Dinkey Dongle | ddremote. You first must enter the product code, dongle number and update number for this dongle. Although you can find this information from DDLook it is also displayed by DDChange, so please run this program (located in the installation directory).

Once you have entered these values please click on the Program Info tab. Click on the *Add to list* button and new dialog box should appear. Enter the value 5 into the *Executions* box and make sure that *Set* is selected. Click OK. Your screen should look similar to the one below:



DDRemote showing Change File Protection Details dialog

Now click on the *Display Update Code* button. Now copy the update code displayed and exit DDRemote. (A copy of the code is written to the file DDCHANGE.TXT and a copy of all the information you have typed in and the update code is written to a file called DDREMOTE.TXT).

Now Alt | Tab back to DDChange and enter the update code. Click on the *Make Changes to Dongle…* button and, if you have entered the code correctly, the following message will appear - *Dinkey Dongle Parameter Changing: parameters changed successfully.* The executions value has now been restored to 5!

You can now run DDLook to prove to yourself that this has actually happened. You now have permission to run CALC five more times before it expires.

In this chapter you have learnt how to use DDAdd, DDLook, DDRemote and DDChange to protect your program and modify the protection parameters in the dongle.

To understand how to use these programs more fully, please read the related chapters *before* you try to protect your own software.

Protection Method

Overview

There are four methods of protecting your software:

- the **Shell Method** for DOS and Windows programs that are in a standard executable file format;
- the **Object Method** for programs that can link in object modules;
- the **DLL Method** for Windows programs that cannot link in object modules, but can call a DLL; and
- the **External Program Method** for programs that cannot use any of the above methods.

Each of these methods is outlined below.

The Shell Method

The Dinkey Dongle software can directly protect Windows (and DOS) programs by putting a protective "shell" around them. The exceptions are:

- Programs that are not .EXE files.
- Visual Studio .NET compiled programs.
- Windows .DLL files.
- Windows 16-bit programs and DOS programs (an older version of DDAdd can do this).

It is possible that the Shell Method will not work properly if your program is not in the standard executable file format. The best thing to do is to try it and see!

The Shell Method requires no preparation - just run DDAdd to protect your program software. The Shell Method is very secure because it encrypts parts of your code and data. For the highest level of security you should use the Shell Method in conjunction with the Object Method (in this case you should use the Object method first).

Shell protection will check the dongle with function 0 (update parameters and start a network user). **Shell2** protection will check the dongle with function 2. See "Calling the DDWIN32 Subroutine" on page 5-1 for more details about functions.

To proceed using this method you need to read "Adding Protection (DDAdd)" on page 6-1.

The Object Method

This method allows you greater flexibility. You can control exactly when and how often the protection is checked and take whatever action you want should the protection not be present. However, if you use this method you have to make some minor changes to your source code. There are three stages to this method:

1. Modify your code so that it checks for the presence of Dinkey Dongle (see "Modifying Your Code" on page 5-1). Checking the dongle is simple and quick, so you can perform it more than once during your program.
2. Link in the appropriate object module to your program.

DDWIN32A.OBJ	for 32-bit Microsoft Visual C++ programs.
DDWIN32B.OBJ	for 32-bit Delphi programs.
DDWIN32C.OBJ	for 32-bit Borland C++ programs.
DDMAC32.O	for Mac OSX programs.
DDLIN32.O	for Linux programs.
DDWIN64A.OBJ	for 64-bit Microsoft Visual C/C++ programs.

   For a complete list of all the object modules available see "Protection Modules" on page 13-1.
3. Run the DDAdd program to lock your software to the dongle (see "Adding Protection (DDAdd)" on page 6-1).

Your software is now protected and ready to use!

To proceed using this method you need to read "Modifying Your Code" on page 5-1 and "Adding Protection (DDAdd)" on page 6-1.

The DLL Method

If you have a Windows program which you cannot link with an object module because, for example, it is an interpreted language like Visual Basic, then we provide a DLL which your program can call. This method has two stages to it:

- Modify your code so that it checks for the presence of the Dinkey Dongle (see "Modifying Your Code" on page 5-1). Checking the dongle is simple and quick, so you can perform it more than once during your program.
- Run the DDAdd program to protect the DLL instead of your program (see "Adding Protection (DDAdd)" on page 6-1). Your program itself is not directly protected but the DLL is. Since your program makes a call to the DLL this gives your program its protection.

We supply 3 DLLs: DD64.DLL (64-bit), DD32.DLL (32-bit) and DD.DLL (16-bit). These DLLs already have the object module linked to them. Therefore when protecting the DLL you should specify Object Method of protection in DDAdd.

To proceed using this method you need to read "Modifying Your Code" on page 5-1 and "Adding Protection (DDAdd)" on page 6-1.

To improve the security of this method you may in addition choose to Shell protect your program and/or use the algorithm when reading and writing data.

The External Program Method

If you cannot use any of the above methods then you have to use the External Program Method. For example, you may be using a very rudimentary programming language that cannot call DLLs but is able to run an EXE program. If you are using a 32-bit DOS extender you generally have to use this method.

The trick with this method is to create a simple dummy EXE file that is protected using one of the above methods and to run this from your program. You should make the program do something that is required for the rest of your program to proceed successfully. For instance, you could pass the current date to it on the command line and get it to create a temporary file that contains this date suitably encrypted. This file could then be read at a later time by your application and action taken accordingly. If the dummy program failed its protection checking, it would have aborted and hence failed to create the temporary file.

A sample DOS dummy program (DDDUM.EXE) and a Windows sample program (DDWDUM.EXE) are provided with the Dinkey Dongle software in the EXTERNAL subdirectory. The chapter "External Program Method" on page 15-1 explains how to use them.

To proceed using this method please read about the method you will need to use to protect your dummy program.

Modifying Your Code

Overview

If you are using the Object or DLL Method you need to make some minor modifications to your code in order to:

- call the Dinkey Dongle subroutine that interrogates the dongle.
- check that the results of this subroutine indicate that it is valid to proceed.

For the Object Method this subroutine is contained in the object module that you will link to your program. For the DLL Method the subroutine is contained in a DLL. What subroutine you call and what module you use depends on the language you are writing in and the Operating System. "Protection Modules" on page 13-1 gives full details.

We supply full working sample code in all the major languages on the master disk. These are installed into the SAMPLES directory. We recommend that you look at the sample for your language and reading the appropriate readme.txt file if it exists.

Please note that the sample code is just a guide so that you can understand how to implement our software. Rather than just blindly copying the sample code it is best to choose the functions that you want and to modify your code using some of the ideas suggested in the chapter "Increasing Your Protection" on page 8-1.

For simplicity, the following text assumes the subroutine is called DDWIN32.

Calling the DDWIN32 Subroutine

Information is exchanged between your program and the DDWIN32 subroutine via a structure which we call the **DDMB** (Dinkey Dongle Memory Block). This structure must be defined in your program, certain fields must be filled in and the address of it must be passed when you call the DDWIN32 routine. The DDWIN32 routine will check for the presence of the dongle and fill in the fields of the DDMB. The structure of the DDMB is

described in detail in "DDMB Structure" on page 12-1. Your program can then check that the values in the DDMB have been filled in correctly.

The first 4 bytes in the DDMB must be set to 'D', 'D', 'M', 'B'.

Bit 0 of the **flags** field can be set to 1 which means that all messages will be suppressed or to 0 which means that all messages will be displayed. The messages we are referring to here are those specified in DDAdd (see "Messages" on page 6-7) and the error code message which can be set in DDAdd (see "Programs

" on page 6-4)

The **function** field must also be set. It can take the following values:

Function = 0. This will do a protection check and update the dongle parameters (e.g. execution count). With a DinkeyNet it will also 'start a network user' and check that the limit on the number of network users has not been exceeded.

Function = 1. This will do a protection check and write to the data area. You need to set the rw_length, rw_offset and rw_data_ptr fields to write to the data area (see below).

Function = 2. This will do a protection check without updating dongle parameters or starting a network user.

Function = 3. This will do a protection check and updates the dongle parameters but does not start a network user.

Function = 4. This will do a protection check and start a network user but will not update the dongle parameters.

Function = 6. DinkeyNet will automatically terminate a network user when the protected program terminates. In some rare cases however, you may want to terminate the network user before your program ends. To do this, call the DDWIN32 routine with function set to 6. This will not do a protection check.

If you are going to check the protection more than once then we recommend that you first use function set to 0 and then use function set to 2 for subsequent protection checks. If function 0 is called more than once then it will decrement an extra execution (and start an extra network user)!

If you want to read/write data from/to the data area in a Dinkey 2 then you must also fill in the fields **rw_length**, **rw_offset** and **rw_data_ptr**. The first four bytes of data pointed to by rw_data_ptr must be 'DDAT'. Your data should immediately follow these bytes. Functions 0, 2, 3 and 4 will read data whereas function 1 will write data. For example: with the function field set to 1 then DDWIN32 will write rw_length bytes of the data at address rw_data_ptr+4 to offset rw_offset in the dongle. If you do not want to read/write data then you should set these fields to 0.

These are the only fields which you need to fill in. You must set the remaining bytes to 0.

Checking the Return Code

You must check the return value from DDWIN32 and then decide on what action to take yourself. A non-zero return code indicates failure and in this case it is recommended that you display the return code along with the **extended error code** (currently six bytes long).

You can also check the other fields in the DDMB to see if they have the correct values. Whilst this is not necessary, it is recommended since it greatly improves the security of the protection. It is also advised that you check these fields in another part of the program, and preferably in different places to make it more difficult for any potential hackers.

For example, the return value of DDWIN32 is also stored in the **r_code** field in the DDMB. This can allow you to independently check the return code elsewhere in the program. For other tips on how to improve the security of the protection see "Increasing Your Protection" on page 8-1.

The details for calling this subroutine vary according to the language your code is written in. Full sample code in all the major programming languages can be found in the SAMPLES directory.

You can also use the algorithm feature to enhance the security of your code.

Note - since version 2.50 the anti-debugging code has been improved to such an extent that if you try to debug or run your program in the development environment it may cause the debugger to crash or not function properly. E.g. It will cause the debugger to crash with the Visual C++, Delphi, C Builder platforms. In this case we recommend that you use one of our debug object modules or DLLs (see "Protection Modules" on page 13-1). In this case you can use the debug object module for the debug build and use the standard object module for the release build.

Some Notes on the DLL Method

In the exceptional case that you are programming in Windows using a language that is unable to allocate memory for the DDMB, you should call a special procedure in the DLL. For 16-bit programs it is called DDDLL1 and for 32-bit programs it is called DD321. "Protection Modules" on page 13-1 has the full details. Essentially, it is the same as the normal procedure, except that you pass it the function number and the flags field instead of a pointer to the DDMB structure.

If you are using the DLL Method then you must specify these functions as imports. This is usually done in the .DEF file, but we also supply a .LIB file that you can use.

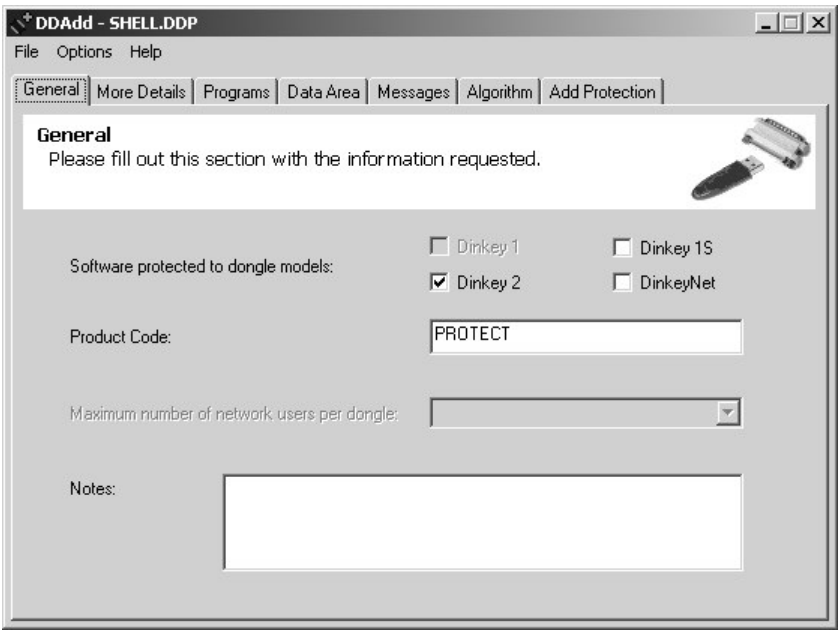
In order to remove potential confusion between your copy of DD.DLL or DD32.DLL and other products' copies of the DLL, we recommend that you rename it to a name of your choosing. (This is difficult if you are using Microsoft Visual C++ compiling 32-bit code and in this case you are not recommended to rename our DLLs. Please look at the sample code in the SAMPLES\C directory for more information on this).

You can also remove confusion by installing the DLL to the installation directory rather than the Windows\System directory.

Adding Protection (DDAdd)

Overview

The next stage is to lock your software to the Dinkey Dongle by running the DDAdd program. When you run DDAdd it will come up with a screen looking like the one below.



DDAdd screen with the shell.dpp file loaded

This program allows you to specify various parameters for the programs that you want protected. If you like, once you have selected the parameters you want, you can save them in a **.ddp file** (Dinkey Dongle Parameter file). This can be loaded the next time you run DDAdd so that you don't have to type in the same information every time you want to protect your program(s). In fact, when you run DDAdd, it will automatically load the last .ddp file that you have saved. The title bar shows you which ddp file (if any) is loaded.

DDAdd can do two things - firstly program the dongle (Dinkey 2 and DinkeyNet only) and secondly modify your program so that at run-time it will look for a dongle that matches the parameters you have specified. Once it has found a matching dongle it will stop searching and return information on that dongle.

Note - Dinkey 1 and Dinkey 1S dongles are not programmed by DDAdd, but each Dinkey 2 and DinkeyNet must be programmed before it is shipped to your customers.

The various protection parameters that can be specified are described in detail in the section below. To find out about the different entries in DDAdd click on protection parameters.

Protection Parameters

General

Dongle Model

Please specify the **dongle model** that you are locking your software to. You can specify more than one dongle model. In this case your protected software will search the parallel and USB ports for any dongle of the type specified. It can also look for a DinkeyNet dongle on the dongle server (this must be setup correctly for this to work properly).

Product Code

This field only applies to Dinkey 2 and DinkeyNet. You must specify a string (spaces are not allowed) up to 8 characters long, ensuring that each of your products has a different product code. The product code (along with the software developer's serial number) uniquely specifies your product.

Maximum Number of Network Users per Dongle

This field only applies if you have a DinkeyNet dongle. First you must decide whether you are going to limit the number of simultaneous network users **per product** or **per program**.

Per program allows you to limit the number of users for each program individually. The total of all these limits must be less than or equal to the maximum value for that DinkeyNet dongle. You can enter these values by clicking on the *Programs* tab.

Per product allows you to limit the total number of users for all your programs. This value must be less than or equal to the maximum for that DinkeyNet dongle. You can enter this value in the box provided.

If you are only protecting one program then it makes no difference whether you select per program or per product.

Notes

This field allows you to store your own notes.

More Details

Type of Protection

Please specify whether you would like to program the dongle (Dinkey 2 and DinkeyNet only) or add protection to your software or both of these options.

DDAdd can both program a dongle and modify your software so that it will search for a dongle with the details specified. When upgrading your software you can specify 'Lock Software only' and as long as the details in DDAdd match with the dongle your customer(s) have then it will work with existing dongles out in the field.

Please note that DDAdd can only add protection to one dongle at a time.

Dongle Settings

As well as having its own dongle number each dongle has another serial number which is unique to your company: the **software developer's serial number** or **SDSN** (except for the Dinkey 1 that does not have this number). The SDSN will be inside each dongle we send you and cannot be modified.

When you protect your software you can do it to an individual dongle, or a range of dongle numbers (with your SDSN) or to any dongle with your SDSN. The advantage of protecting your software to any dongle with your SDSN is that once your software is protected it will work with any dongle that has been correctly programmed. If you

choose to lock your software to a particular dongle then you will have to repeat the process each time you send the software to a customer.

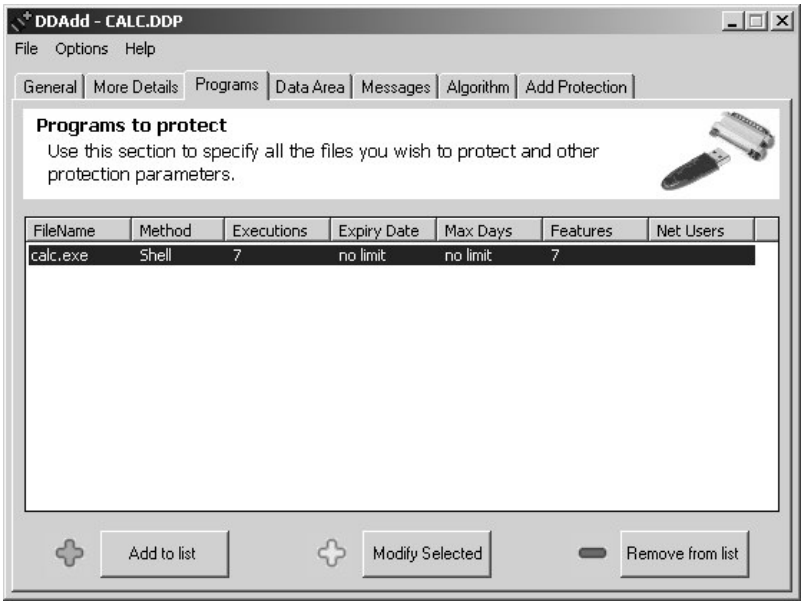
Note: Dinkey 1 and 1S dongles do not need to be programmed, but Dinkey 2 and DinkeyNet dongles must be programmed by DDAdd before they are distributed.

Note: If you decide to *Lock software to a dongle number of the dongle attached to the computer* and it is a Dinkey 2 then you must also select to the program the dongle.

Note: The maximum dongle number is 4,294,967,295. You can also enter *max* in the dongle range maximum.

Programs

Use this tab to specify the files that DDAdd is to protect. If you are using a Dinkey 2 or a DinkeyNet you can also enter various protection parameters for each program. Each filename must be unique as they are stored in the dongle. This is so your protected program can retrieve the correct parameters during a protection check.



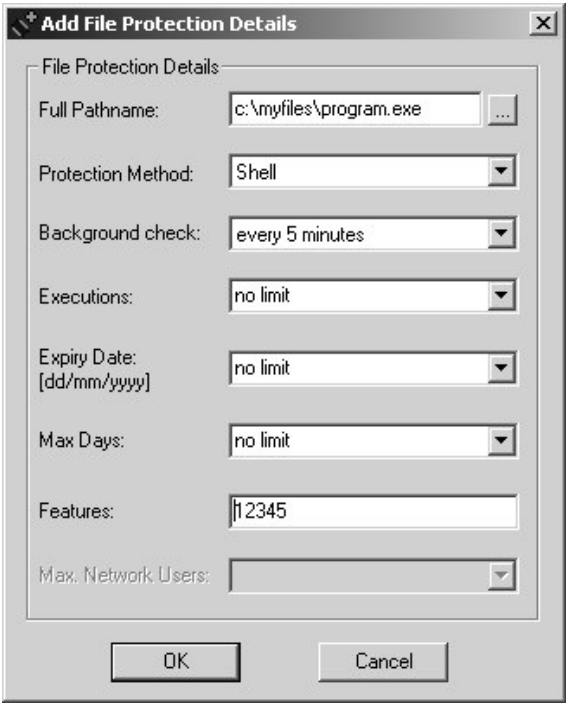
DDAdd File Protection Details

To add new filenames to the list click on the 'Add to list' button. You can change the parameters for a file by double-clicking on that entry or by moving the highlight to that entry and clicking on the 'Modify Selected' button. Pressing the DELETE key or clicking on the 'Remove from List' button will delete the highlighted entry from the list.

Dinkey 1 lets you lock an unlimited number of programs to any one dongle. Dinkey 2 lets you protect up to 17 programs per dongle.

To add files to the list click the 'Add' button and another Dialog Box will open. See the *DDAdd Add File Protections Details dialog* Figure below.

You should enter the full pathname of the program you want to protect or use the browse button. For Mac OSX if the application is a bundle then you should protect the actual program file in the contents\macos directory.



DDAdd Add File Protection Details dialog box

The following paragraphs describe the entries in the Add File Protection Details dialog box.

Method - the method of protection: Shell, Object or Shell2. (see "Protection Method" on page 4-1). If you are using the DLL method then you should protect the DLL (rather than your program) using the Object Method of protection.

The Shell method of protection does a protection check at the beginning of the program with function set to 0 whereas Shell2 does a protection check with function set to 2.

Background Check - this only applies if you have selected the Shell or Shell2 method. This will cause the protected program to check the dongle at repeated intervals for the duration of the program.

With Dinkey 2 and DinkeyNet you have a number of options here that you can use to control your software.

Executions - the number of executions of your program that are allowed. This can be a number between 1 and 65534, or 'no limit'. When calling the DDWIN32 subroutine, function 0 or 3 will decrement the execution count whereas function 1, 2 or 4 will leave it unchanged. By calling function 0 or 3 more than once you could use this parameter to limit the number of times certain parts of your program are executed.

Expiry Date - after midnight on the specified day, calls to the DDWIN32 subroutine will return error -25 (expired). If bit 0 of the 'flags' in the DDMB is 0 then the Expiry Message will be displayed. The latest date that it can be set to is 31/12/2107, although it can be set to 'no limit'.

Max Days - the maximum number of days that your software will run after installation. This can be a number between 1 and 32766 or 'no limit'.

NOTE - 'installation' is the first time that the DDWIN32 subroutine is called from this particular program after it has been protected by DDAdd. If you test your program before you send it out then Dinkey 2 will assume that this is the installation date. In this case you must add the protection to Dinkey 2 again before sending it to your customer.

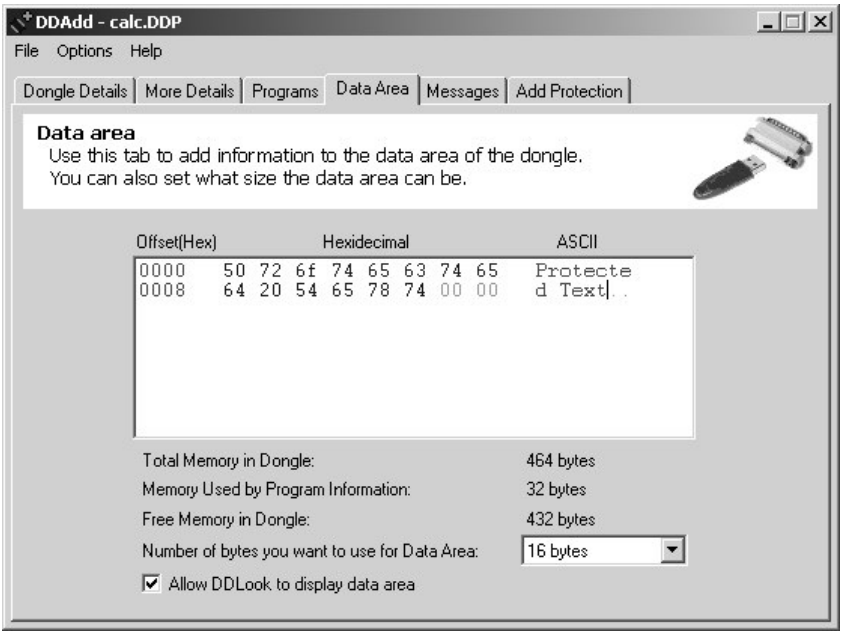
Features - 4 bytes that are for your own use. It can be used, for example, to determine which features of your program are permitted to be used.

Network Users (DinkeyNet only) - this value limits the maximum number of simultaneous network users that can execute the specified program. (i.e. it is the per program value and not the per product value. If per product is selected then this entry is greyed.) You can use any number between 0 and 4095 or ‘no limit’. Please note that the sum of the limits of all the protected programs must be less than or equal to the maximum for that DinkeyNet dongle.

User Data Area

This tab applies only to Dinkey 2 and DinkeyNet. It allows you to specify up to 432 bytes of user data that can be programmed into the dongle.

The memory usage of the dongle is dynamically divided into two parts: that taken up by the parameters for the protected programs and that used by the data area. This information is displayed by the dialog box. It will automatically calculate how much memory is available in the dongle for data. You must still specify, however, how much data you would like to enter. Don't specify more than you actually need. You can increase the size of the data area later using DDRemote if you need to.



DDAdd Data Area dialog box

You can enter data in hexadecimal notation or in ASCII. If you enter in ASCII then the hexadecimal equivalent will be displayed and vice versa.

If you want DDLook to be able to display the data then please tick the box. Otherwise the data will not be displayed.

Messages

The **Secure Signon Message** is displayed on return from DDWIN32 if there was no error. It is fully protected by Dinkey Dongle's encryption and anti-debug code.

The **Unauthorised Use Message** is displayed if a copy of the program is run without the correct Dinkey Dongle present.

The **Product Expiry Message** is displayed (Dinkey 2 and DinkeyNet only) if the Expiry Date has passed or the Execution count has reached 0.

The **User Limit Exceeded Message** (DinkeyNet only) is displayed when the number of network users has exceeded the limit you specified.

Messages can be up to 255 characters long and can be suppressed by setting bit 0 of the flags field to 1 in the DDMB.

Display Error Code if Protection Invalid

If this option is selected then an error code will be displayed by the Dinkey Dongle software if the protection check fails. Remember, however, that if you set bit 0 of the flags field in the DDMB, the display will be suppressed. If this option is not selected then it is strongly recommended that you display the error code and extended error code from within your own code.

Algorithm

This tab applies only to Dinkey 2 and DinkeyNet. It allows you to specify your own user-defined algorithm for the dongle. This greatly improves the security that is offered by the Dinkey Dongle. You can make up an algorithm using four variables and six operators: addition, subtraction, modulo (i.e. remainder after division) and the logical operators: AND, OR and XOR. You are allowed to use up to 6 operations in your algorithm.

The algorithm is displayed as you make up the algorithm. It must start and end with a variable. If you click on the "Generate Source Code…" button then sample code is generated in C/C++, Visual Basic and Delphi.

The next section explains how the algorithm is used when you perform a protection check.

Firstly, before each protection check you must call the function GetAlgVars. This will produce 4 non-zero random variables. If you have specified an algorithm you must always call this function before you call a protection check. What you do next depends on whether you are reading or writing when you do a protection check. There are three cases which are outlined below:

Protection Check with no Reading or Writing Data

1. Call the GetAlgVars function.
2. Call the Protection Check function.
3. Calculate the answer to your algorithm using the variables returned from the GetAlgVars function. Compare this answer with the answer returned in the alg_answer field of the DDMB after the protection check. If the answers are not equal then exit with an error.

Protection Check and Reading Data

1. Call the GetAlgVars function.
2. Call the Protection Check function and read the data.

3. The data returned is encrypted with the answer to your algorithm (using the variables returned from GetAlgVars function). You must decrypt the data by Xor-ing each byte with the answer to your algorithm.

Protection Check and Writing Data

1. Call the GetAlgVars function.
2. Encrypt the data that you want to write by Xor-ing each byte with the answer to your algorithm (using the variables from the GetAlgVars function).
3. Call the protection check function in the normal way

Notes

1. If you have specified an algorithm then you must always call the GetAlgVars function previous to a protection check each time you call the protection check function.
2. It improves the security if each step can be separated with other code.
3. Please look at sample code in the Samples Directory.
4. Once you have set the algorithm in the dongle then it cannot be modified by DDRemote and it cannot be viewed by DDLook. If you want to change the algorithm you must re-program the dongle using DDAdd.
5. The algorithm is not available if you are using 16-bit modules (DD.DLL, DDWIN.OBJ etc…). Some primitive language are not able to implement the algorithm method of protection.

Add Protection

Now that you have entered all the appropriate details you can protect your software by clicking on the *Add Protection Now* button. This programs the dongle (if necessary) and protects each of the files you specified, displaying an error message if it fails. Unless the message tells you otherwise then the file may have been changed so you must replace it by the original file before you try to add the protection again.

At this point you may also want to keep a note of the protection details such as the dongle number, product code and to whom the dongles will be sent. You will need this information at a later date if you use DDRemote or send out an upgrade.

DDAdd Command Line

DDAdd takes parameters on the command line to enable you to automate the generation of update codes. This is the syntax:

DDAdd [<filename>] [/b] [/q]

where:

filename specifies the name of a ddp file that DDAdd should open (see "DDP Structure" on page 12-5 for the structure of the DDP file. There are also some sample DDP structures in the sample code directory).

/b indicates that the DDAdd will be run in "batch mode". i.e. the dongle and/or program will be protected immediately and then DDAdd will close. (Must be used with filename parameter).

/q indicates that DDAdd will be run in "batch mode" but will not display messages (unless there was an error).

Upgrading your Software

When upgrading your software you want to lock your software to a dongle that is already in your customer's machine. You should choose 'Lock Software Only' in the More Details tab. For Dinkey 2 this means that they will use the protection parameters (i.e. executions, max days, expiry date, and features) of the original programs, although the other details that you specified when upgrading (i.e. messages and 'If Protection Invalid') will be used by the new software.

NOTE - if you want to do an upgrade protection using Dinkey 2, the program names of the upgraded software must be the same as those of the original software. If you want to change the program names or include new programs in the protection then you need to use DDRemote as well. With Dinkey 1 and Dinkey 1S these restrictions do not apply.

Remote Changing of Parameters (DDRemote)

Overview

This section applies to users of Dinkey 2 and DinkeyNet only. It explains how you can change the protection parameters of your software that has already been protected. For this you need to use two programs: **DDRemote** and **DDChange**.

Firstly, you run the DDRemote program, and then type in the details that you want changed. DDRemote will issue you with a special update code which you can give to your customers over the telephone or by e-mail. Your customer then runs the DDChange program and types in this code. The required changes will then be made to the dongle. So you can be sure that the customer has actually carried this out, DDChange issues a unique confirmation code that must match up with the one generated by DDRemote.

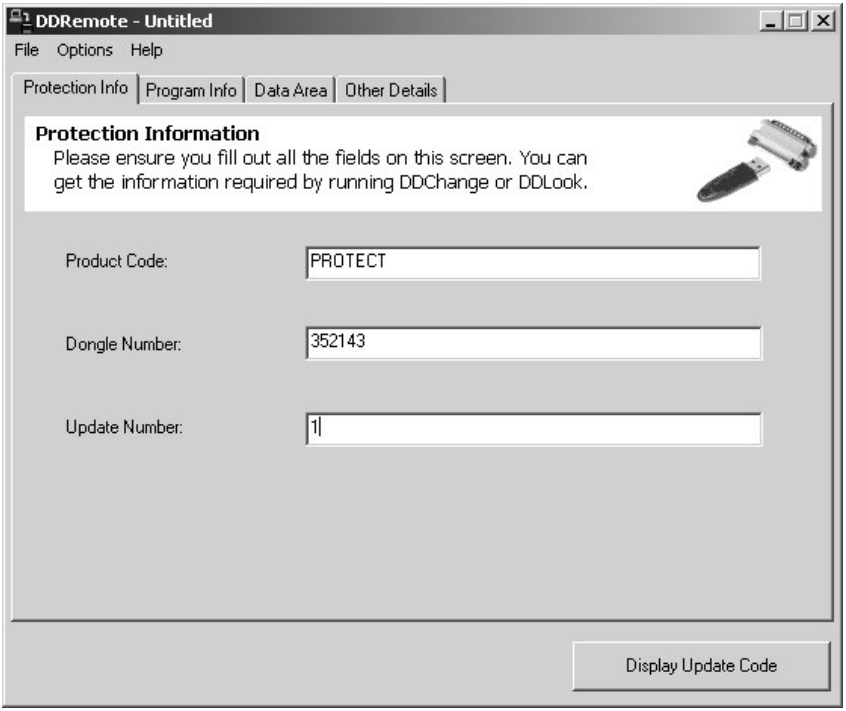
If you like, once you have selected the parameters you want, you can save them in a **.ddr file**. This can be opened the next time you run DDRemote so that you don't have to type in the information next time. This can be particularly useful if you want to make many complicated changes (to the data area for example). You can configure DDRemote to load the last ddr file you had open. The title bar shows you which ddr file (if any) is loaded.

You must, of course, supply your customer with DDChange if you intend them to use this feature.

DDRemote

Protection Info

The first thing that you have to do is specify the product code, the dongle number and the update number for the dongle concerned. You may have kept a record of the dongle number and product code when you first protected your programs for your customer. If not, then you can ask your customer to run DDChange which will display a list of the dongles connected to his machine and the product code, the dongle number and the next update number for each one. They can copy and paste this information.



DDRemote main screen

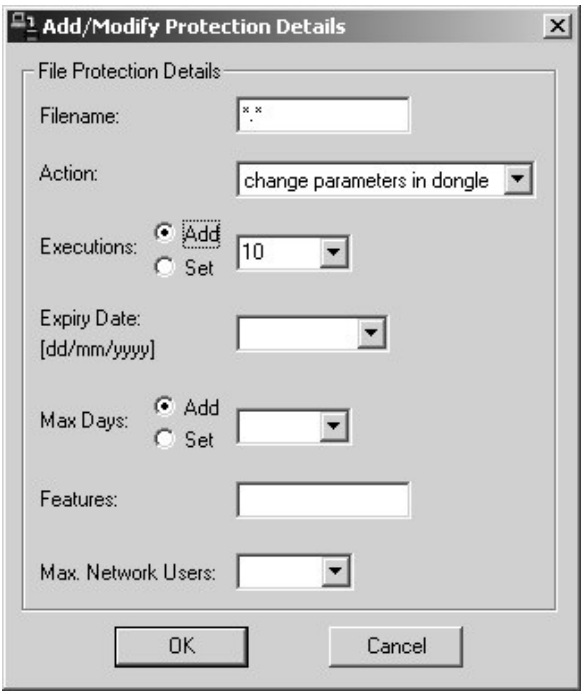
The **update number** starts at one and is incremented by one each time DDChange has successfully run. This is to ensure that your customer can use your code once and once only. In fact, all DDRemote update codes will be unique to that particular dongle and for the particular operation that you have specified. This ensures that nobody else can issue DDRemote codes on your behalf.

Please note that since the dongle is characterised by its SDSN, product code, dongle number and update number these things cannot be changed by DDRemote. Everything else can. This makes it a very powerful tool.

In DDRemote you specify which of your programs you want to change and what you want to change. Only enter information into those parameters that you want to change. If you do not want a certain parameter changed then leave that field blank.

Program Info (Changing File Protection Details)

In the Program Info tab if you click on *Add to list* a new dialog box will appear.



DDRemote Change File Protection Details dialog box

You have the option of changing parameters for all the programs protected by a dongle by specifying *.* for the filename. If you have only protected one program you can also use *.* as it makes the update code shorter.

As well as changing the parameters of programs that are already protected by a Dinkey Dongle you can delete programs from the list in the dongle or add new programs to that list. If you delete a program from the list then that program will always fail the protection check. If you want to add a program to the dongle list then you must protect it

specifying 'Upgrade' protection (using DDAdd) and then run DDRemote to specify the protection details.

Add Executions will add the specified number of executions to the execution count of that program. You can select 'no limit' if you want this restriction to be removed.

Set Executions will cause the execution count of the program to be set to the number specified. You can set this to 'no limit' also.

Expiry Date will set a new expiry date for that program. You can select 'no limit' if you want any existing date restriction to be removed.

Add Max Days will add the specified number of days to the greater of the current date and the expiry date to produce a new expiry date. This can be set to no limit.

Set Max Days will allow the program to run for the specified number of days from the first time it has been run after DDCHANGE has been successfully executed. Any previously set expiry date will be ignored. This can be set to 'no limit'.

Features lets you change features (4-bytes of data).

Max. Network Users will change the limit for the number of simultaneous network users for this program. If the number of simultaneous network users is currently limited 'per product', then it will convert this to 'per program'.

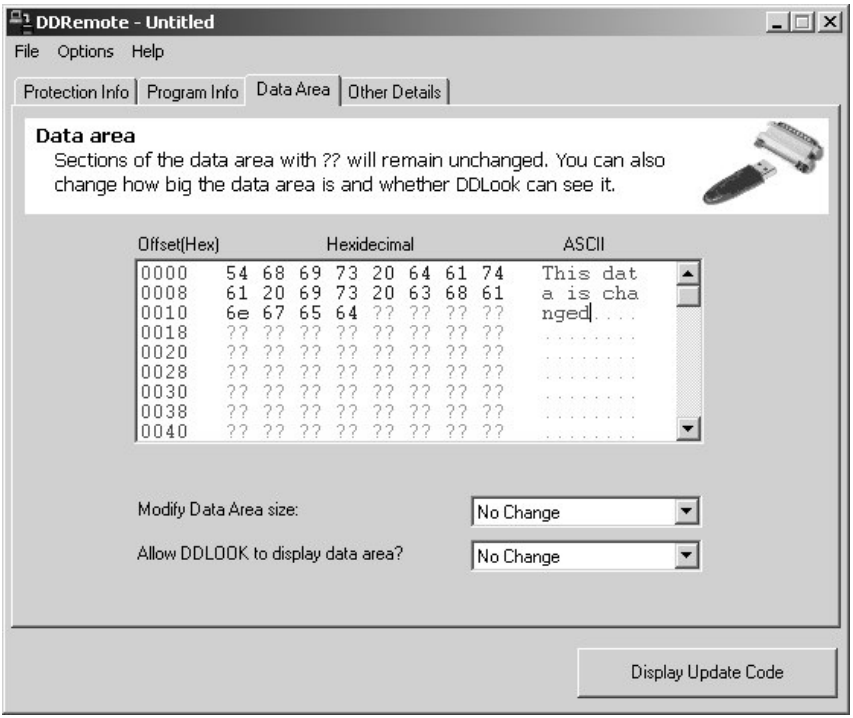
Changing the User Data Area

The *Data Area* tab lets you modify (or create) a data area in the Dinkey Dongle.

The maximum possible data area is displayed. Note that the actual data area available in the dongle may be smaller than the total number of entries listed. A '??' denotes that you do not want to change the value of the data at that offset. Only enter the values that you want to change (in hex or ASCII). If you change values that are outside the limit of the current data area then DDChange will automatically increase the data area size (if possible) and make the necessary changes.

You can extend (if possible) or reduce the size of the data area by changing **Modify Data Size**.

You can change whether DDLook displays the data area by changing **Allow DDLook to display data area**.



DDRemote Change Data Area dialog box

Note - if you do change the data area using DDRemote the update code is likely to be long. You may want to distribute the code as a file (see "DDC" on page 7-6).

Changing the Maximum Number of Network Users

This field (in the *Other Details* tab) allows you to modify the number of simultaneous network users 'per product' for a DinkeyNet. If you had previously limited the network users 'per program' then this will convert it to 'per product'. If you want to change the per program value then change the value in New Protection Parameters.

Changing Last Date Used

DDRemote also lets you reset the 'Last Date Used' in the dongle (in the *Other Details* tab). You would only need to do this if one of your customers had altered the date on his system clock. This could have been an accident or they could have been deliberately trying to cheat the system. In either case the protection check would fail, giving an error 1. It is up to your discretion as to whether you issue this update string to reset the recorded last date used.

Notes

This section can be used to store your own notes about the code that is produced. These notes are written to the log file for your reference.

Generating the Update Code

When you have entered the relevant details you can click the *Display Update Code* button to generate the update code. It also generates two files: DDREMOTE.TXT and DDCHANGE.TXT. DDREMOTE.TXT is a cumulative summary of the changes that you have requested and the update code produced. Please note, that next time DDRemote is run, the data will be appended to this file. DDCHANGE.TXT is a copy of the update code. You could give this file to your customer instead of telling him the code over the telephone, so that he can input it from a file rather than type it in.

DDRemote Command Line

DDRemote takes parameters on the command line to enable you to automate the generation of update codes. This is the syntax:

DDRemote [<filename>] [/b] [/q]

where:

filename specifies the name of a ddr file that DDRemote should open (see "DDR Structure" on page 12-8 for the structure of the DDR file. There are also some sample DDR structures in the sample code directory).

/b indicates that the DDRemote will be run in "batch mode". i.e. the code will be generated immediately and then DDRemote will close. (Must be used with filename parameter).

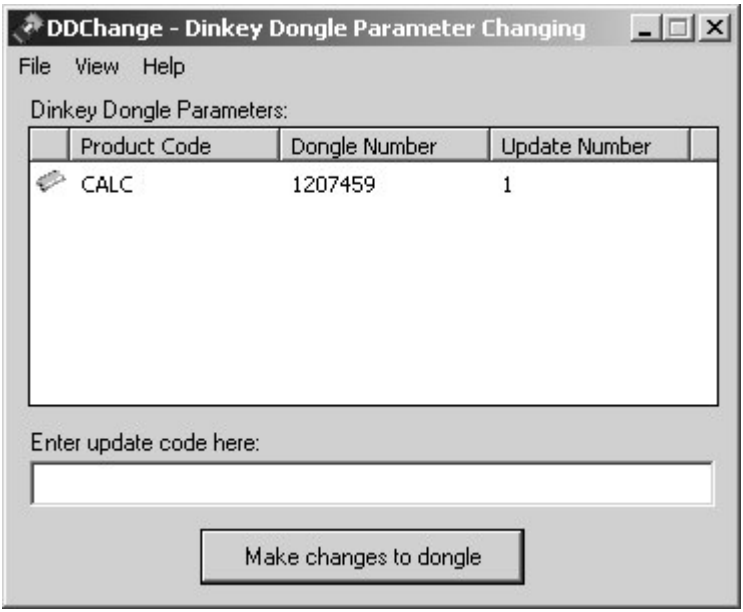
/q indicates that DDRemote will be run in "batch mode" but will not display messages (unless there was an error).

DDChange

There are two forms of DDCHANGE that you can use: The DDCHANGE application and the DDCHANGE dynamic library (ddchange.dll). It is up to you which one you use - whichever is the most convenient.

DDChange Application

When DDChange is run it will display the dongle number, product code and expected update number for each dongle attached to the machine. You can copy & paste this information by right clicking on the appropriate line in the list. It will also allow the user to enter the update code. The update code can be entered with or without spaces and in lower or upper case. A sample screen-shot is given below:



DDChange dialog box

DDChange.dll

If you want to provide your own Windows interface and incorporate DDChange into your program then you should use DDChange.dll. There are two functions that can be called: DDINFO and DDCHANGE.

Calling **DDINFO** gives you information about how many dongles there are attached to the machine and for each of these dongles it tells you the dongle number, the product code and the next update number to be used. You may choose to display this information since it is required by DDRemote.

DDINFO has the following syntax:

return code = DDINFO (ddcb)

DDINFO requires only one parameter - a far pointer to a structure called the Dinkey Dongle Change Block (**DDCB**). The return code is an integer and is 0 if the function is successful.

The DDCB contains a fixed portion followed by a number of 'blocks' - one for each of the dongles you want to find information about. The structure of the DDCB is described in "DDCB Structure" on page 12-5.

You need to set the first 4 characters to 'D', 'D', 'C', 'B', and set num_blocks to the maximum number of Dinkey 2 and DinkeyNet dongles that you want information about. You can restrict the search to dongles with a product code identical to that in prod_code and a software developer's serial number identical to that in sd_sn. When you call DDINFO, it fills in information about each dongle it finds into the 'blocks' until it has done every dongle or num_blocks dongles whichever is the smaller. It also fills in num_blocks with the total number of Dinkey 2 and DinkeyNet dongles connected to the machine that meet the restrictions specified in the DDCB.

The main difficulty is that you don't know in advance how many dongles there are attached to the user's machine. This means that you don't know how much memory to allocate to ddcb (i.e. how many blocks ddcb has). Here are two methods that you can use to solve this problem:

- Restrict the search to dongles with your software developer's serial number only and with your product code only. You know how many dongles that you have issued to this customer with this product (usually one), and you can allocate space accordingly. In general this is the only dongle that you are interested in anyway. This is the simplest method. Sample code can be found in the SAMPLES directory.
- Make two calls to DDINFO. For the first call specify that you want information on no dongles (i.e. set num_blocks to 0 in the ddcb). If this call was successful, num_blocks will now contain the number of dongles in the user's machine. Now you can allocate enough memory to ddcb so that it contains this number of 'blocks'. Call DDINFO a second time and if successful display the information for all the dongles present.

Due to the flexible nature of DDINFO you could produce variations on these two methods to suit your needs.

NOTE - If an error occurs calling DDINFO, it is useful to print not only the return code but also the extended error code returned in ext_err.

The syntax for calling the **DDCHANGE** function is:

return code = DDCHANGE (update_code, ext_err_code, quiet)

update_code is a far pointer to the update code (a null terminated string in upper or lower case with or without spaces). If the string is of the form "/F*filename*", the update code will be read from *filename*.

ext_err_code is a far pointer to some memory where the extended error information will be placed if there is an error. The first 5 bytes must be set to 'D','D','E','R','R', the following 6 bytes are the extended error code.

quiet is an integer that indicates whether DDChange.dll should display messages, error codes and the confirmation code. Set it to 0 to display them or 1 to suppress all display.

The **return code** is an integer. It is 0 on success and the confirmation code will be returned in bytes 6 & 7 of ext_err_code. If it is non zero then an error has occurred and the extended error code will be set if necessary. You should display the return code and the extended error code in this case.

DDCHANGE (DOS)

This MS-DOS program has the following syntax:

DDCHANGE [/Ffilename] [/Q] [/?]

The optional /Ffilename allows DDCHANGE to read the update code from the specified filename (for example DDCHANGE.TXT) instead of the user typing it in, /Q causes all DDCHANGE display to be suppressed. The /? Option shows the syntax for this command.

If DDCHANGE is run without any options, it will display the dongle number, the product code and the expected update number for each dongle attached to the machine and then prompt the user for the update code. It then either gives an error message or reports success and displays a confirmation code.

Increasing Your Protection

Suggested Techniques

This chapter contains techniques to further improve the protection that Dinkey Dongle already offers. It is very important to read this chapter as the security we offer is only as strong as your implementation of our software.

What follows applies to the Object Method (or the DLL Method) only. If you are using the Shell method then you can improve the security by also using the Object method or DLL method.

Any protection system can be beaten provided the hacker puts in enough time and effort. Any supplier that tells you otherwise is simply not telling the truth. The art is to increase the time and expertise that a hacker would need in order to crack the protection. After a certain point, even the most dedicated hacker is going to give up.

Dinkey Dongle combines a number of different techniques to beat the hacker. Our code is very secure and very difficult to debug. However, the weak point is the link between your code and our code. To make your code really secure you can use some of the extra techniques outlined below.

Use the Shell Method as well as the Object Method.

You can combine the Shell Method and the Object Method of protection by running DDAdd twice specifying 'Object' protection on one occasion and 'Shell' or 'Shell2' protection on the other. You should use the Object method of protection first if you decide to do this. In addition to performing a protection check the Shell Method also securely encrypts your file.

Call the dongle from many places throughout your program.

If you only call the dongle once then it is much easier for a hacker to isolate the call. Furthermore, they could remove the dongle after the protection check has been done and transfer it to another machine.

Call the dongle at regular time intervals in your program.

For example you might want to do a protection check every 10 minutes. If you call the dongle only once and you have set an expiry date, the user could execute the program and leave it running indefinitely without the program expiring. The expiry message will only come up if you do a protection check.

Check many of the values in the DDMB.

The weak part of any protection is the call to the subroutine accessing the dongle. If a hacker isolated this call he could simply skip the call, patch the return code to 0 and carry on with the program. If, however, you check many of the values such as the product code, secure signon message, dongle number and the software developer's serial number, then the hacker would have to work out these values too.

Another good value to check is the time returned in the DDMB. By finding out the system time yourself before you make the call you can see whether the protection check has taken too long (this could be caused by a hacker trying to trace through the DDWIN32 subroutine). Another reason for checking this value is that it will be different every time you call the Dinkey Dongle.

If you have a Dinkey 2 or a DinkeyNet dongle you could also read values from the data area.

Check the DDMB values many times, in separate places in your program.

If you can spread out these checks in your program and check them many times in different ways it will make hacking your code much more difficult.

Checksum DD32.DLL (DLL Method only).

If you are calling DD32.DLL or DD.DLL or using the external program method then we recommend that you do a checksum of this file to make sure that it has not been modified by a third party. Note - you will have to calculate the checksum and hard-code this into your software. If you upgrade DD32.DLL or protect it differently then the checksum value will have changed.

Put data from your software into the dongle data area

Take out some of the data (e.g. a variable) from your program and place it in the dongle (Dinkey 2 and DinkeyNet only). Every time you modify this data you will have to write it to the dongle. Every time you use this data you will have to read it from the dongle. (Remember that it may take approx. 0.2-0.5 seconds to read/write to the dongle depending on how much data you read/write. It is best to use this technique on a variable that changes very frequently.)

Using this technique you are effectively placing part of your program in the dongle. This makes it very hard for a hacker to beat! It is best to just use the data after it is read rather than specifically checking it has been changed to the correct value.

Specify your own algorithm

If you specify your own algorithm then the security is greatly increased. This is especially recommended if you are reading or writing data using the DLL method. Try to keep the code that calculates the algorithm separate to the code that performs the protection check. You should also call the GetAlgVars function in a separate place to the protection check.

DDLook

DDLook

This chapter covers an extra utility: **DDLook**. This program will search all the USB ports (and parallel ports for Windows) for Dinkey Dongles and display information regarding their contents.

For Dinkey 1 DDLook will display the dongle model and the dongle number. For Dinkey 1S it will also display the software developer's serial number. For Dinkey 2 it will display all this and the update number, the product code and all the program information (expiry date, executions etc…). It will also display the user data area if this was allowed when the dongle was protected by DDAdd. For DinkeyNet it will additionally display the number of simultaneous network users.

You may want to give DDLook to your customers so that they can view the protection parameters. On the other hand you may not want them to view these parameters! For the data area (in Dinkey 2) we give you the choice of whether DDLook will display it or not. You can specify this in DDAdd and change it (should you need to) using DDRemote.

DDLook is an excellent diagnostic program as it displays an error code plus extended error code if any problem occurs. It can also display some diagnostic information. For these reasons we recommend that you include DDLook on your customer's master disk.

Note for 16-bit and DOS users: You can find a 16-bit version of DDLook in the 16bit subdirectory and a DOS version (called DOSLOOK) in the DOS subdirectory. The syntax for DOSLOOK is:

DOSLOOK [/diag]

where **/diag** will print out some diagnostic information in addition to the normal display.

Driver Installation

Overview

Note: This information applies to setupdrv version 3.4 or higher. If you have a previous version of setupdrv then you will need to look at an earlier release of the documentation.

Both the parallel port and USB Dinkey Dongles may require device drivers to be installed to the Operating System before they can be detected by your software. This means that you must install the necessary device drivers on your customer's computer.

The parallel port dongle only requires for all OS except DOS, Win3.11, Windows 95/98/Me. This means that the OBJ modules and DLLs that we provide will not work on their own. The device driver allows our software to work on these operating systems and means we can supply software that works on any Windows operating system.

The USB dongle requires device drivers for all the operating systems that it is to be installed on. i.e. Windows NT4, Windows 98/Me, Windows 2000/XP/2003 and 64-bit versions of these operating systems. It is not compatible with DOS, Windows 3.x, Windows 95 or Windows NT 3.51. This is because the USB port is not implemented on those operating systems.

With the Dinkey Dongle SDK we supply a program called **setupdrv**, which will detect the operating system running and install the appropriate drivers for you. If dongle drivers already exist then it will upgrade them (if necessary). Setupdrv also allows you to uninstall the device drivers. By default setupdrv will attempt to install drivers for both the parallel port and USB dongles.

We recommend that you call setupdrv to install these device drivers when you install your software. One way to do this is for your installation program to install setupdrv.exe to some directory on the hard disk and then execute setupdrv (please see syntax below). All the device drivers are included in the setupdrv file.

For your information please find below a list of the device drivers and the type of dongle and operating system they are used with:

File	Parallel/USB	Operating System
ddnt.sys	parallel	Windows NT/2000/XP
ddvdd.dll	parallel	Windows NT/2000/XP
usbkey.sys	USB	Windows 98/Me/2000/XP
usbkey.inf	USB	Windows 98/Me/2000/XP
usbkey.vxd	USB	Windows 98/Me
ukeyvdd.dll	USB	Windows 2000/XP
ddusbkey.inf	USB	x64 Windows OS
ddusbkey.sys	USB	x64 Windows OS
usbhub.sys	USB	Windows NT4
usbhcd.sys	USB	Windows NT4
usbkeynt.sys	USB	Windows NT4
u4keyvdd.dll	USB	Windows NT4

Note: you will need administrator access rights to install and uninstall the device drivers under Windows NT/2000/XP/2003.

USB drivers

If you are protecting your software with the USB dongle then you need to install the USB drivers. Because the USB dongle is a plug and play device this means that Windows will detect it each time it is attached and removed from the system.

We recommend that your installation program installs the drivers using setupdrv before the USB device is first attached to the computer. Windows will then recognise the device when it is attached and complete the installation with no user input (or minimum user input for Windows XP).

In practice however, some people will attach a USB dongle to a machine before installing the drivers using setupdrv. If you want to cope with this situation then you should include the following files (uncompressed) on the root directory of your installation media: usbkey.inf, usbkey.sys, usbkey.vxd, ukeyvdd.dll, ddusbkey.inf, ddusbkey.sys (Windows can work out whether to use the 32-bit or 64-bit drivers)

If you do this then if the 'New Hardware' wizard appears, if the user chooses 'Search (recommended)' then Windows will find and install the drivers automatically. If the user chooses 'Specify location' then they must tell Windows to look at your installation media (e.g. CDROM).

Please note that if the user does install the dongle software in this way you may still need to run setupdrv (e.g. if you want to install the drivers for the parallel port dongle). There is no harm in running setupdrv if the drivers have already been installed and so we recommend you always do this.

Windows 98/Me exceptions:

- When you first attach the USB dongle the drivers will be installed. If your software is 16-bit then you will have to reboot before it will be detected by the software correctly.
- Running setupdrv will not upgrade the USB drivers - the user must either: uninstall the drivers, reboot and re-install the new drivers; or choose Control Panel | System | Device Manager, select Security Key, choose 'update drivers' and point Windows to the location of the new drivers.

To Install Drivers

The syntax for installation is:

setupdrv [/par] [/usb] [/q]

If **/par** is specified only the parallel port drivers will be installed. If **/usb** is specified only the USB drivers will be installed. The default is to install drivers for both the parallel port and USB dongles.

If the **/q** option is specified it will suppress the ‘drivers have been successfully installed’ message. Any error messages will still be displayed.

To Uninstall Drivers

To uninstall the Dinkey Dongle drivers the syntax is:

setupdrv /u [/q] [/ufull] [/usb] [/par]

If the **/q** option is specified it will suppress the 'drivers have been successfully uninstalled' message. Any error messages will still be displayed.

The **/ufull** option will fully uninstall the USB drivers under the Windows 2000/XP operating systems. For other operating systems it performs the same operation as the /u option.

If you run **setupdrv /u** it will perform the Microsoft-recommended uninstall. However, for Windows 2000/XP this will still (intentionally) leave some information on the machine. If you use the /ufull option it will remove this information. This can be useful if you are experiencing problems and want to ensure that the machine is entirely 'clean'.

If **/par** is specified only the parallel port drivers will be uninstalled. If **/usb** is specified only the USB drivers will be uninstalled. The default is to uninstall drivers for both the parallel port and USB dongles.

Troubleshooting

Although we have made driver installation as simple as is possible occasionally users get themselves into problems. In that case following this extensive procedure should always solve any problems they are having:

1) Remove any dongles.
2) Run setupdrv /ufull
3) Reboot the machine.
4) Run setupdrv
5) Insert the dongle into the machine (for USB dongles, Windows should now install the drivers)
6) Reboot the machine.

Installing the Mac drivers

In the Driver subdirectory is the file ddUSBKeyDriver.pkg. We recommend that you distribute this file to your customers. To install the drivers just double-click this package. This copies the file ddUSBKeyDriver.kext to the System/Library/Extensions folder.

Uninstalling the Mac drivers

Delete the file ddUSBKeyDriver.kext from the System/Library/Extensions folder.

Installing the Linux drivers

1. From a terminal window uncompress the drivers file using the command:

tar -xvzf drivers-1.1.4.tar.gz (or whatever the drivers file is called)

2. Type 'make' whilst in the directory the tar file has been unpacked to.

3. Type 'make install' to install the usb driver. Note - you must have "superuser" privileges to do this.

Uninstalling the Linux drivers

From the directory that the drivers are located in execute the following commands:

1. type 'make uninstall' to uninstall the drivers.
2. type 'make clean' to remove the drivers.

Using DinkeyNet

Overview

This chapter details the setup procedure for installing DinkeyNet dongles onto your customer's network. The DinkeyNet software is network-independent and uses the same setup procedure for each network. All networks are supported! At least, we have not come across one that Dinkey Dongles do not work with. The only restriction is that the operating system must be MSDOS (4.0 onwards) compatible, including Windows (3.x, 9x, Me, 2000, XP and NT) and that one machine must be running a version of Windows.

DinkeyNet dongles can be connected to any machine that is running any version of Windows. We refer to this machine as a **dongle server**. If you have more than one DinkeyNet dongle, they can be connected to the same or separate dongle servers.

The dongle server needs access to a path that can be accessed from all the workstations that you would like your programs to run from. We refer to this path as the **DinkeyNet path**. The DinkeyNet path can be on any machine on the network, including the dongle server. DinkeyNet will create a sub-directory off this path. The workstations will need full access rights (including read/write/create/delete) to the path. Different dongle servers can use different paths or they can share the same path. (However, if they do share the same path then your program should be protected to a specific dongle rather than a range of dongles.)

Note: You will need to install the appropriate drivers (using Setupdrv) to the dongle server only. You do not need to install the drivers to the workstations.

Note: If the protected program on the workstation crashes then the network user is terminated. If the workstation itself crashes then it may take the server a little while to terminate that network user. If the workstation is re-booted then the network user is lost immediately.

Note: If you use function 0 then the 'execution count' feature will not work properly: it will decrement an execution even when the number of network users is exceeded. For this reason we recommend you use either function 3 or 4 to do a protection check.

Note: If the DinkeyNet path is located on an NT/2000/XP workstation then the Operating system will not allow more than 10 simultaneous network users. If you want to allow more than 10 network users the DinkeyNet path should be located on the Server.

Note: if you use the "debug" object modules or dll then the network user check will not work properly.

Important Note: if you are using version 3.60 (or higher) runtime modules then you must upgrade ddnet to version 3.60 (or higher) for it to function properly.

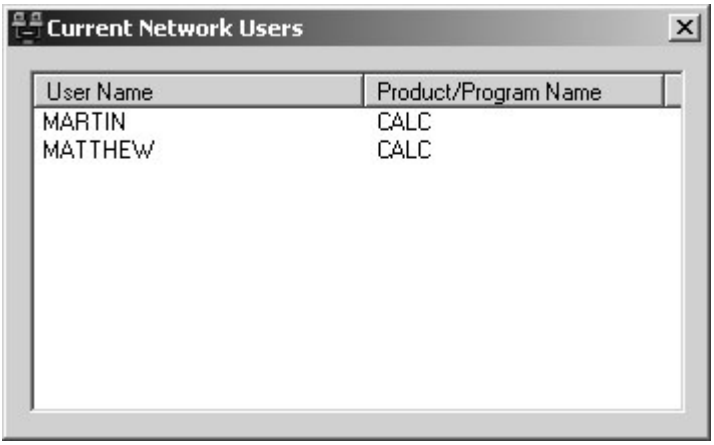
Dongle Server Setup

The **DDNet** program needs to be running on the dongle server for dongle checking to function. It can be run as a Windows program or as an NT Service. In general we recommend it is run as a Windows program because it has greater functionality. However, if your customer want an NT Service (i.e. a special program that can run on a Windows NT/2000/XP server without anyone having to log-on) then they can run it as a service.

If it is run as a Windows program, it should be loaded when the operating system boots up. You can do this by adding DDNet to the Startup Group.

When DDNET runs, it interrogates the dongle(s) and looks in DDNET.INI for a DinkeyNet path. If it cannot find this information it will ask the user for the path. Once this has been established it will store the path in DDNET.INI and not ask for this path again.

Ass DDNet runs in the background we add a DDNet icon to the status bar so you know it is running. If you right-click this icon it will let you terminate the program or view the current network users (it displays the machine name that the user is logged-on to, not the user name).



DDNet showing current network users

The syntax for DDNet is:

DDNET [/Q] [/T] [/U] [/I<DinkeyNet Path>] [/S] [/N] [/V]

If you want to run DDNet without it asking the user for the DinkeyNet path you can use the **/I** switch: DDNet /I<DinkeyNetPath>. Note that there should be no space in-between the /I and the path specified.

Normally this program will run until the computer is shut down. However if you want to terminate DDNet then run DDNet **/T**. It will display a message to confirm that it has stopped. Alternatively you can right click on the DDNet icon in the status bar and choose "Terminate".

If you want to completely uninstall DDNet i.e. stop it and remove all the files it has created: ddnet.ini files and the DinkeyNet Path subdirectories) then use the **/U** switch.

To suppress messages that are not fatal errors use the **/Q** switch.

If you want to suppress the icon displayed in the status bar whilst DDNet is running use the **/N** switch.

If you want DDNet to install itself as an Service (for use with Windows NT/2000/XP) then use the **/S** switch. The service will then start itself and ensure that it is loaded each time Windows NT/2000/XP boots-up. The service works in the same way as DDNet, but has the advantage that you do not have to log-on in order for it to run. However, it has the disadvantage that it will only work if the DinkeyNet path is on the local machine (i.e. the dongle server). Note that if you want to uninstall the service you should use /u /s parameters.

Any run-time error messages produced by the service are stored in the Event Log. Use the Event Manager to view these messages (look in Application events).

If you have already have DDNet Service running but sometimes want to view the network users then you can log-on and run DDNet /v to do this. This will allow the Service to run as normal but also allow you to view the network users. Use the status bar icon to display users and terminate this DDNet Viewer.

Note - if you want to modify the DinkeyNet path the you can edit the ddnet.ini file. However, you need to stop and restart DDNet for it to use this new information.

Workstation Setup

For each workstation that you want to run your program from, you need to have an INI file that contains the DinkeyNet path. The INI file must be called **<PRODCODE>.INI**, where PRODCODE is the product code you gave your software when running DDAdd. For example, if you used the product code FRED then this INI file should be called FRED.INI.

The INI file should contain the section title '**dinkey**' and the entry title '**DinkeyNetPath**'. The entry should contain the full path of the DinkeyNet path. It can be a mapped drive or use the Unified Naming Convention (UNC) e.g. DinkeyNetPath=z:\apath or DinkeyNetPath=\\amachine\apath. For Mac machines the format will be something like: DinkeyNetPath=/volumes/WORKGROUP;MACHINEA/foldername.

The INI file should be in the Windows directory (unless you are protecting a DOS program, it which case it must be in the root directory of the C drive).

The easiest way to create the entry in the INI file is by using the Windows API call WritePrivateProfileString. If you cannot create the INI file by calling WritePrivateProfileString from your setup routine then you can copy our sample: PRODCODE.INI from the NET directory. Your users can then edit it using a text editor and type the path in place of \\server\apath. They must then rename the file to <PRODCODE>.INI and place it in the Windows directory (for Windows) or the root directory of the C drive (for DOS).

Note - On Mac or Linux the <prodcode.ini> file should be located in the same directory as the protected program. If the program is a bundle under Mac OSX then the ini file should be in the Contents\MacOS folder.

Note - When protecting 16-bit programs you can use long filenames in the DinkeyNet path as long as the machine is not running Windows NT.

DinkeyNet Quick Tour

The following section will guide you through protecting a simple program with the DinkeyNet dongle. We will again use the Windows Calculator, CALC.EXE, which is located in either the Windows directory or the Windows\System32 directory.

1) Run DDAdd and open up the calc.ddp file. Modify the dongle model to "DinkeyNet" and specify the maximum number of network users (this can be per product or per program, but in this case we are only protecting one program so it doesn't make any difference).

2) Copy CALC.EXE to the c:\Program Files\dinkey directory. If you want to choose another directory then you must modify the directory specified in DDAdd.

3) In DDAdd click "Add Protection Now". This will program the dongle and protect CALC. If you run CALC now it will give error 33 (no ini file on workstation).

4) Copy the sample prodcode.ini file located in the NET directory to the Windows directory of the workstation machine. Rename this file to calc.ini and modify the DinkeyNet path entry to be the path you have chosen as the DinkeyNet path. Also copy CALC to the workstation. If you run CALC now it will give error 36 (DDNet not yet run).

5) Now install the drivers on the machine you have specified as the dongle server, by running setupdrv. Also attach the dongle to this machine. Copy DDNet to a directory on this machine and run it. It will ask you for the DinkeyNet path for this dongle - please enter exactly the same path as you specified in step 5.

6) Now run CALC from the workstation. The "successfully interrogated dongle" message should appear, followed by the calculator itself. If you get error 36 then it means that you specified a different DinkeyNet path in step 5 to that in step 6.

   Note: If you want to modify the DinkeyNet path you specified in step 5 then you can directly modify the calc.ini file. If you want to modify the DinkeyNet path you specified for DDNet then you can modify the ddnet.ini file in the Windows directory. You must stop ddnet by running ddnet /t and then run it again for the changes to take effect.

7) You must now repeat step 5 for each of the workstations you want to run CALC from.

You can now start CALC from many of the workstations. If the number of simultaneous network users are exceeded then you get error 67 and the "network user limit exceeded" message specified in DDAdd.

Dinkey Dongle Structures

DDMB Structure

The Dinkey Dongle Memory Block (DDMB) is a structure that is used as an interface between your program and the DDWIN32 subroutine. The DDMB is structured as follows:

Field	Length	Type	Description
ddmb	4	char	Always DDMB
func	1	byte	Function No.
r_code	2	int	Return Code from function
ext_err	10	byte	Extended error (only the first 6 bytes at present)
vers	2	byte	Version no. (major, minor)
dongle_sn	4	long	dongle number
sd_sn	2	int	Software Developer's serial number
p_code	9	char	Product Code (ASCIIZ)
p_name	13	char	Original Program Name (ASCIIZ)
execs	2	int	No. of executions left
exp_day	1	byte	expiry day
exp_month	1	byte	expiry month
exp_year	2	int	expiry year
feature	4	long	feature

secure_msg	256	char	Secure Signon Message (ASCIIZ)
update_num	2	int	Number of successful updates for this dongle
flags	2	int	flags field
time	4	byte	GMT time according to the OS
reserved	4	dword	Reserved
rw_offset	2	byte	Offset to read/write data
rw_length	2	byte	Length of data to read/write from data area
rw_data_ptr	4/8*	pointer	Address of data to read/write
net_users	2	int	Maximum number of network users
net_user_data	4	long	Needed if you use function 6
reserved2	1	byte	Reserved
usb	4	long	Tells you if the dongle was detected in the parallel or USB port.
model	4	long	tells you the model of the dongle found.
alg_answer	4	long	the result of your algorithm.
reserved3	4	long	reserved (set to 0)
res_expand	156	byte	Reserved for expansion

*4 bytes for 32-bit code, 8 bytes for 64-bit code

NB If execs is set to no limit, it will be filled in as -1

NB If expiry date is set to no limit, it will be filled in as 31/15/2107

Detailed Description

A more detailed description of each field follows, using the following key:

➜ Field can be initialised before checking protection

➡ Field MUST be initialised before checking protection

➜ Field is filled in after protection has been checked.

ddmb	➡	This field must be filled with the ASCII characters "DDMB"
func	➡	Function number for the protection check. Function 0 will update parameters and start a network user. Function 1 is used to write to the data area. Function 2 will not update any parameters. Function 3 will update parameters only. Function 4 will start a network user only. Any of the functions 0,2,3,4 can be used to read the data area. Function 6 is used to terminate a network user manually ("Calling the DDWIN32 Subroutine" on page 5-1).
r_code	⬅	The value of the return code from a protection check.
ext_err	⬅	Extended error code. This is very useful if you get an error accessing the dongle. We recommend that you display this error if the return code is non-zero. Currently only the first six bytes are used.
Vers	⬅	Dinkey Dongle software version no. (major, minor)
dongle_sn	⬅	The dongle number
sd_sn	⬅	The Software Developer's serial number of the dongle.
P_code	⬅	The Product Code that you specified in DDAdd.
P_name	⬅	Original name of your protected program. This program name is stored in the dongle. It will remain unchanged if you rename your program. You must specify this name if you use DDRemote.
Execs	⬅	Number of executions left
exp_day	⬅	Expiry day (1 to 31)
exp_month	⬅	Expiry month (1 to 12)
exp_year	⬅	Expiry year (1980 to 2107)
feature	⬅	A secure four byte value that can be used for any purpose you like. For example, you could choose to restrict access to certain features of your program according to its value.
Secure_msg	⬅	The Secure Signon Message that was specified in DDAdd. This is displayed if the protection check succeeds (if you are using function 0 and bit 0 of flags1 is 0)

update_num	←	Number of successful updates that have been carried out for this dongle. This number will be one less than that displayed by DDChange (next update number).
Flags	➡	flags field. Set bit 0 to 1 to suppress all messages that may be displayed by the Dinkey Dongle software during a protection check.
dos_time	←	GMT time of protection check according to OS. 4 bytes containing hours, minutes, seconds and hundredths of a second.
Reserved	←	Reserved (must be zero)
rw_offset	→	If you want to read/write the data area then this is the offset in the data area that you will read/write from/to. If not used then this should be set to zero.
Rw_length	→	The length of the data that you are read/writing (excluding the leading DDAT). If you are not read/writing any data then set this to 0.
Rw_data_ptr	→	Address of data that you want to read/write. Data must start with 'DDAT'. If you are not read/writing any data then this should be set to zero. If you are using the special VB functions DDDLLVB or DD32VB then this field is ignored.
net_users	←	Maximum number of network users of your software on a network (-1 is no limit).
net_user_data	←	Reserved. This value must be preserved if you use function 0 followed by function 6
res_expand	←	Reserved for expansion (must be zero-filled)
Reserved2	←	Reserved (must be zero)
usb	←	Set to 1 if the dongle detected was found in the parallel port. Set to 2 if the dongle was found in the USB port.
model	←	Set to 1 = Dinkey 1, 2 = Dinkey 1S, 3 = Dinkey 2, 4 = DinkeyNet
alg_answer		The result of your algorithm using the variables specified by GetAlgVars. Only returned if you specify an algorithm and if you are doing a protection check with no writing or reading.

DDCB Structure

The DDChange Block (DDCB) is a structure that is used as an interface between your program and DDCHANGE.DLL. It is used to pass information about the Dinkey Dongles in a user's machine. It is of variable length and is structured as follows:

Field	Length	Type	Description
ddcb	4	char	Always DDCB
sd_sn	2	int	software developer's serial number
prod_code	9	char	product code (ASCIIZ) (to restrict search)
ext_err	6	byte	extended error code
num_blocks	2	int	maximum number of dongles to search for. After call: the number of dongles found meeting the restrictions

and for each 'block':

dongle_number	4	long	dongle number
product_code	9	char	ASCIIZ string
update_num	2	int	next update number to be used

DDP Structure

DDAdd saves the protection parameters to a ddp file. Here is the structure of that file:

Field	Length	Type	Description
ddp	3	char	Always DDP
Version	3	char	ASCIIZ string - version of DDP file
Dinkey1	4	int	0 = not selected, 1 = selected
Dinkey1S	4	int	0 = not selected, 1 = selected
Dinkey2	4	int	0 = not selected, 1 = selected
Dinkeynet	4	int	0 = not selected, 1 = selected

ProdCode	9	char	Product Code (ASCIIZ)
MaxNetUsers	4	int	Per product net users (-1 = unlimited)
ProtectionType	4	int	0 = software & dongle, 1 = dongle only, 2 = software only
DongleSettings	4	int	0 = lock to SDSN, 1 = dongle in machine, 2 = range specified
RangeMin	4	uint	minimum dongle number in range
RangeMax	4	uint	maximum dongle number in range
NumProgs	4	int	number of programs to protect
FileInfo	17*293	struct	file information for a maximum of 17 files - see structure below.
DataArea	865	byte	data area hex values - in ASCIIZ
DataAreaSize	4	int	size of the data area (in multiples of 16)
LookData	4	int	0 = DDLook will not show the data area 1 = yes it will.
SignonMsg	256	char	Secure Signon Message (ASCIIZ)
UnauthMsg	256	char	Unauthorized Use Message (ASCIIZ)
ExpiryMsg	256	char	Expiry Message (ASCIIZ)
NetLimitMsg	256	char	Network user limit reached message (ASCIIZ)
DisplayErr	4	int	0 = don't display errors, 1 = display error
Algorithm	4	int	encoded form of algorithm
Notes	255	char	user notes

The FileInfo structure is outlined below:

Field	**Length**	**Type**	**Description**
ProgName	257	char	Full pathname of program
Method	4	int	0 = Shell, 1 = Object, 2 = Shell2
Background	4	int	0 = no background, 1 = every 1min, 2 = 5 min, 3 = 10min, 4 = 30min, 5 =1hr
MaxDays	4	int	-1 = no limit

ExpiryDay	4	int	-1 = no limit
ExpiryMonth	4	int	
ExpiryYear	4	int	
Execs	4	int	-1 = no limit
Features	4	uint	
ProgNetUsers	4	int	number of per program net users

DDR Structure

DDRemote saves its parameters to a ddr file. Here is the structure of that file:

Field	Length	Type	Description
ddr	3	char	Always DDR
Version	3	char	ASCIIZ string - version of DDR file
ProdCode	9	char	Product Code (ASCIIZ)
DongleNum	4	uint	Dongle Number
UpdateNum	2	short	Update Number
NumProgs	4	int	number of programs to modify
FileInfo	17*57	struct	Information about protection parameters to change to certain files - see below
LastDayUsed	4	int	0 = no change
LastMonUsed	4	int	
LastYearUsed	4	int	
DataAreaSize	4	int	size of the data area (in multiples of 16) if this is to change, 0 = no change
LookData	4	int	0 = no change, 1 = don't display, 2 = do display
DataArea	865	byte	data area hex values - in ASCIIZ, but ?? signifies 'no change'
MaxNetUsers	4	int	Per Product network users, -2 = no change, -1 = no limit
Reserved	29	byte	set to zero
Notes	255	char	user notes

The FileInfo structure is outlined below:

Field	Length	Type	Description
FileName	13	char	Name of program to change (ASCIIZ)

Action	4	int	0 = change, 1 = delete, 2 = add
MaxDays	4	int	-1 = no limit
AddSetMaxDays	4	int	0 = add, 1 = set, -1 = no limit, -2 = no change
ExpiryDay	4	int	-1 = no limit, 0 = no change
ExpiryMonth	4	int	
ExpiryYear	4	int	
Execs	4	int	
AddSetExecs	4	int	0 = add, 1 = set, -1 = no limit, -2 = no change
Features	4	uint	
ChangeFeatures	4	int	0 = don't change features value 1 = do change features value
ProgNetUsers	4	int	number of per program net users -2 = no change, -1 = no limit

Protection Modules

Reference Guide

A number of modules are supplied with the Dinkey Dongle package. You must use one of them when protecting your code using the Object method. What you use depends upon the language and operating system that you are using. The following table is a summary:

DDDBG64A.OBJ	for x64 programs.
DDDBG64.DLL	for x64 programs
DDWIN32A.OBJ	for 32-bit Microsoft Visual C++ programs.
DDWIN32B.OBJ	for 32-bit Delphi programs.
DDWIN32C.OBJ	for 32-bit Borland C++ programs.
DD32.DLL	for 32-bit Windows programs.
DDDBG32A.OBJ	for 32-bit Microsoft Visual C++ (debugable).
DDDBG32B.OBJ	for 32-bit Delphi programs (debugable).
DDDBG32C.OBJ	for 32-bit Borland C++ (debugable).
DDDBG32.DLL	for 32-bit Windows programs (debugable).
DDINET.DLL	to protect web pages on a server using ASP/PHP.
DDMAC32.O	for Mac OSX programs.
DDLIN32.O	for Linux programs.
DDLIN32X.O	for Linux programs, using XWindows for messages.
Dinkey.fmx	for Filemaker 6 (Windows)
Dinkey7.fmx	for Filemaker 7/8 (Windows)
Dinkey7.plugin	for Filemaker 7/8 (Mac)
Dinkey.x32	Xtra for Authorware and Director

DDJava.dll	module to protect Java pages (Windows)
DDCOM.dll	for Visual Lisp in AutoCad
Dinkey.4dx	module for 4d programs
DinkeyRB.dll	Windows plug-in for Real Basic programs.

The following are in the 16bit subdirectory:

DDWIN.OBJ	for 16-bit Windows programs
DDWINL.OBJ	for large-model 16-bit Windows programs
DD.DLL	for 16-bit Windows programs

The following are in the DOS subdirectory:

DD.OBJ	for DOS programs
DDPROT.OBJ	for programs written with a DOS extender, except:
DDXM.OBJ	for programs using the XM DOS Extender
DDCLAR.BIN	for Clarion programs before version 3
DDCLAR3.OBJ	for Clarion 3 programs

DD32.DLL and all the DDWIN32 object modules are very similar modules for 32-bit Windows applications. Linking DDWIN32 to your program is more secure than using DD32.DLL but DD32.DLL is more convenient to use. Visual Basic and Clarion 4 users must use DD32.DLL.

DDWIN32.OBJ and DD32.DLL are very secure and so if you debug your protected application it can crash once you have done a protection check. For this reason we have produced some "debugable" object modules and DLLs: DDDBG32.OBJ and DDDBG32.DLL that you can use so that you can debug you protected code. Note, in general it is best to use the standard object modules for the release version of your code. This only applies Visual C++, Delphi and C Builder users. (Note - the network users check will not work with the debug modules).

DDDBG64A.OBJ and DDDBG64.DLL are 64-bit versions of these modules for x64 Operating Systems.

DDMAC32.O is the object module for Mac OSX programs. You must all link the IOKit.framework Carbon framework because the object module uses some Carbon functions.

DDLIN32.O is the object module for Linux programs. By default, if you have asked the modules to display messages they are displayed to stdout. We also provide a module, DDLIN32X.O that displays messages using XWindows. You need to compile it including the following switches:

gcc -L/usr/X11R6/lib -lXm -lXt -lX11 "prog.c" ddlin32X.o -o "youroutput"

DD.DLL and DDWIN.OBJ are very similar modules for 16-bit Windows applications. Linking DDWIN.OBJ into your program is more secure than using DD.DLL, but DD.DLL is often more convenient to use.

DD32.DLL is a 32-bit library and should be used to protect a 32-bit program. The associated lib file DD32.LIB is for use with Microsoft Visual C++. Borland users must generate their own LIB file using the Borland IMPLIB program.

In general, for 16-bit Windows programs that can link object modules, DDWIN.OBJ should be used. However, if this causes your data segment to exceed the 64K limit or you have exceeded this limit in any case (i.e. large model) then you should use DDWINL.OBJ.

The name of the subroutine depends on the module used. The following table is a summary.

Module	**Procedure Names**
DDDBG64A.OBJ	ddwin64
DDDBG64.DLL	dd64, dd641, dd64VB
DDWIN32A.OBJ, DDWIN32B.OBJ DDWIN32C.OBJ, DDDBG32A.OBJ DDDBG32B.OBJ, DDDBG32C.OBJ	DDWIN32
DD32.DLL	DD32, DD321, DD32VB
DDMAC32.O	ddmac32
DDLIN32.O, DDLIN32X.O	ddlin32
DDWIN.OBJ	DDWIN
DDWINL.OBJ	DDWINL
DD.DLL	DDDLL, DDDLL1, DDDLLVB
DD.OBJ	_dd, DDBFP, _DDJPI
DDPROT.OBJ	ddprot, real_dd, DDBFP, REAL_DDBFP _DDJPI, _REAL_DDJPI
DDCLAR.BIN	DDCLAR
DDCLAR3.OBJ	_DDJPI

Note: The sample code in the SAMPLES directory has working examples of these functions in many different programming languages.

All the procedures perform the same functions but use different parameter passing conventions. They all (except DDDLL1 and DD321) expect to be passed the address of

a correctly filled in DDMB. DDDLL1 and DD321 take two parameters: Function and Flags, and are designed for programming languages that cannot pass a pointer to a memory block.

The 32-bit Windows procedures (DD32, DD321 and DD32VB) follow the standard calling convention for 32-bit procedures. i.e. stdcall. The 16-bit Windows procedures (DDDLL, DDDLL1, DDWIN, DDWINL and DDDLLVB) follow the standard calling convention for Windows procedures, i.e. they are Far Pascal procedures.

DDDLLVB and DD32VB are special functions for Visual Basic. They are used for reading and writing to the data area in Dinkey2. Please look at the sample code in the SAMPLES\VB directory for instructions on how to use them.

The _dd and ddprot procedures use the Microsoft C parameter passing convention i.e. they assume that the DDMB far address is PUSH-ed onto the stack (segment first then the offset within the segment) prior to issuing a FAR CALL and that the stack is adjusted by the calling program after returning from _dd.

The DDBFP routine uses the parameter passing convention used by most Basic, Fortran and Pascal programs, i.e. the same as for C but the stack is adjusted by DDBFP before returning to the calling program.

The _DDJPI routine uses the TopSpeed C calling convention (also used by Clarion 3) which passes the address of the DDMB in registers instead of on the stack.

The _real_dd, REAL_DDBFP and _REAL_DDJPI routines in the protected mode modules enable you to call the Dinkey Dongle code from real mode sections of your protected mode program. They correspond to the DD.OBJ procedures _dd, DDBFP and _DDJPI respectively.

Error Codes

Overview

All Dinkey Dongle programs return error codes. If you or your customers encounter any error codes then please use the tables below to try to fix the problem. If you have suppressed run-time error messages, you can temporarily turn them on by setting the **DD_ERRCODE** environment variable. To do this set DD_ERRCODE=ON. Setting DD_ERRCODE=OFF will suppress the error messages. Removing this entry will return your system back to its original state.

In DOS you just need to type SET DD_ERRCODE=ON. If you are in Windows 3.11/9x/Me then you need to modify the autoexec.bat and re-boot Windows. Under Windows NT/2000/XP you can set this environment variable by going to Control Panel | System | Environment.

The extended error code is currently six bytes although we have allowed ten bytes space for it in the DDMB. They are only meaningful to Microcosm. These should be displayed whenever an error occurs as it will help Microcosm to identify the problem. DDLook also acts as a diagnostic tool as it will always display the extended error code if there is any problem accessing the dongle. We recommend using the online support for the latest error code information: http://www.microcosm.co.uk/support.

The error codes have been arranged to correspond with those of Microcosm's CopyControl product wherever possible.

Error Codes common to all programs

These error codes are common to DDAdd, DDChange, and the run-time code.

1	The system clock has been tampered with. You remove this error you must reset the last date used in the dongle using DDRemote.

4	Error accessing the dongle. The extended error code has more information.
5	Dongle memory has been corrupted - contact Microcosm.
6 to 15	DPMI errors.
16	Dongle memory has been corrupted - contact Microcosm.
20	This program is not in the list of protected programs in Dinkey 2 / DinkeyNet. It could be that it is the wrong dongle for this program or that you have deleted this program from the program list inside the dongle (using DDRemote).
24	The execution count has reached 0 (product expired).
25	The expiry date has been exceeded (product expired).
26	A dongle is present, but it has the wrong dongle number. This error could also be generated if there was a problem reading the dongle memory. The extended error code gives more information.
27	Not enough free memory (Shell method).
28	No dongle can be found in the machine. This error could also be generated if there is a problem accessing the dongle. The extended error code gives more information.
29	General Error. Extended error code will give more information.
30	Dinkey 2 / DinkeyNet has not been protected yet.
31	Data block does not start with DDAT.
32	Attempting to read/write beyond data area.
33	Cannot find <ProdCode>.ini file on workstation.
34	Cannot open <ProdCode>.ini file.
35	Cannot read <ProdCode>.ini file.
36	Either DDNet has not been run yet or the path in <ProdCode>.ini file is incorrect.
37	DDNet has not yet been run for this dongle. Stop ddnet and start it again.
42	Dinkey 1S is corrupted.
43	ddnt.sys could not be found. Run setupdrv to install this device driver on Windows NT (2000/XP).
45	Error creating file in DinkeyNet path (check access rights).
46	DinkeyNet error. Error communicating between workstation and dongle server. To solve this problem please stop ddnet, reboot machine containing the DinkeyNet path and then restart ddnet.

48	Error writing to file in DinkeyNet path (check access rights).
50	DDNet is not running on dongle server.
51	Error reading file on DinkeyNet path (check access rights).
53	Error deleting file on DinkeyNet path (check access rights).
54	DinkeyNet error. Please check that the system date on the workstation agrees with the date on the dongle server.
55	Error creating file on server path (check access rights).
56	Error using function 6.
62	Error calling the ddnt.sys device driver under Windows NT (2000/XP)
66	Error getting temporary directory.
67	Too many network users.
68-70	Error creating / writing to temporary file.
71	This program has not been protected by DDAdd.
72-73	Error writing to temporary file.
74	Could not create subprocess.
76	Subprocess failed to terminate.
77	Error deleting temporary file.
78-81	Error opening / reading temporary file.
82	No more files left of the form DD3216??.*. Delete all files of this form from the temporary directory and try again.
83	Temporary directory is too long.
84-85	Error opening / reading temporary file.
87	Error calling Windows 2000/XP USB driver.
88	Error calling Windows 98/Me USB driver.
89	This operating system does not support the USB dongle.
90	Could not load drivers for parallel or USB dongle.
91	Error calling NT4 USB drivers.
92	Error accessing USB drivers. For version 4.1 or later you must install the latest USB drivers (run setupdrv).
95	DD software cannot run on 286 processors or earlier. Please use version 2.00 or earlier.
96	GetVersion Windows API call failed. Extended error code has more details.

97	Setupapi.dll could not be loaded. This DLL must be in the Windows System directory.
123	Network file sharing is badly implemented on the network. Please modify this so that the network protection can work properly.
124	'DDMB' not found in DDMB.
125	Cannot load the device driver. Please correctly install the device drivers by running setupdrv.exe.
126	Error calling the device driver.
128	Timeout on protection check..
129	Internal error with plugin.
130	The parameters passed to the plug-in are in an incorrect format.
132	Bad pointer as arguments in GetAlgVars function
133	Did not call the GetAlgVars function before a protection check.
134	Data encryption method is too new for this module.
135	Incompatible ddnet with runtime module - upgrade ddnet.
163	No DOS memory available (protected mode object module).
164	No DOS memory available (protected mode object module).
198	Protection check is performing function that has not been implemented in this version of the code.
199-205	Error communicating with Mac drivers - please check that they have been installed correctly.
109-210	Error communicating with Linux drivers - please check that they have been installed correctly.
215	The <productcode.ini> file is not in the correct format.

Troubleshooting

Parallel - If the error code you are getting asks you to refer to an extended error code then there may be a problem communicating with the dongle. In that case please check whether there any other devices attached to the parallel port. If there are remove them and then check to see if the dongle works.

Dinkey Dongle is designed to be compatible with all printers, bi-directional or otherwise. In some cases however, it may be necessary for the printer to be powered on for the dongle to work properly.

If other parallel port devices are causing problems then we recommend attaching the dongle to the output (printer) port of the device, if one exists. In particular this is what we recommend for ZIP drives.

General - If you are still having problems accessing the dongle then please use our automated online support: http://www.microcosm.co.uk/support. Failing that contact Microcosm stating the error code produced and extended error code (DDLook will display this information even if your product doesn't).

DDChange Error Codes

2	Incompatible versions of DDChange and DDRemote.
17	The update code typed in is different to the one DDRemote generated.
18	Wrong dongle number, product code or update number used in DDRemote.
19	Not enough memory in the dongle to allow you to make these changes. The combined length of the data area and the number of programs protected exceeds the maximum for this dongle.
20	General error. Extended error code gives more details.
21	Tried to extend expiry date beyond limit.
59	Tried to specify to many network users.
60	Dongle is not a DinkeyNet.
68-70	error creating / writing temporary file.
72-73	error writing to temporary file.
74	could not create subprocess.
76	subprocess failed to terminate.
77	error deleting temporary file.
78-81	error opening / reading temporary file
82	No more files left of the form CH3216??.*. Delete all files of this form from the temporary directory and try again.
83	temporary directory is too long.
100	ddcb doesn't start with DDCB (DDChange.dll).
101	error_code doesn't start with DDERR (DDChange.dll).
102	too many command line options (DDChange).

103	cannot open file containing update code.
105	not enough memory.
106	cannot read file containing update code.
107	not enough memory.
108	invalid command line options (DDChange).
109-111	not enough memory.
113-116	update code has been entered in an incorrect format.
117	not enough memory.
131	trying to add a new program to the dongle program list, but this program already exists.

External Program Method

DDDUM program examples

DDDUM.EXE and **DDWDUM.EXE** are example programs for use with the External Program Method (see "The External Program Method" on page 4-4). DDDUM is a simple DOS program and DDWDUM is a simple Windows program. They both create a data file with some data hidden in it. They both have already been linked with a Dinkey Object module, so you only need to run DDAdd to protect it (using the Object Method of protection). Windows programs can also be protected by the External Program Method but you would usually use the DLL Method instead.

You will need to include in your program an instruction to run DDDUM (DDWDUM) and, later in your program, you will need to read the data file produced by it.

DDDUM can be run from DOS programs including those written in DBASE, 1-2-3, TAS, SMART etc. like any other DOS program. You will need to specify some command line parameters.

The DOS command that your program should produce is:

DDDUM <Data_filename> <Key> <Offset>

where <Data_filename> is the name of the data file that you want DDDUM to produce; <Key> is a number (0-65535) that should, preferably, be different every time; <Offset> is the offset from the beginning of the data file (1st position = 0) where DDDUM should put an "encrypted" value. The "encrypted" value is the sum of <Key> and <Offset>. For example:

DDDUM FILE1 2 14

would cause a file called FILE1 to be created with the value 16 (2 + 14) inserted at the 15th byte (i.e. offset 14).

The actual statement you need to insert in your program to execute a DOS program is different in different systems. In SMART, for instance, you prefix the normal DOS command with COMMAND /C. For example:

COMMAND /C DDDUM FILE1 2 14

Check your programming manual for details on how to run a DOS program.

For added security, you could make <Offset> different every time as well as <Key>. The date or time of day can make a convenient basis for a random number. The value for <Offset> must be less than 1000. The values for <Key> and <Offset> should be such that the sum of them is less than 32768.

Later in your program, you should read this data file and check that it contains the correct value at the specified position. If the file does not exist or the value is not (Key + Offset), you know that DDDUM has not run successfully and is, therefore, not a valid copy. The (Key + Offset) value is stored as an ASCIIZ string without quotation marks.

Note that DDDUM is just a sample program. There are many other ways of using this method. Your program would be much more secure if you produced your own equivalent (and perhaps more complicated) program. However, you are free to use DDDUM.EXE if you wish. If you do so, we suggest you rename it to a name of your choosing so that pirates will not realise that you are using DDDUM.

If your program is a Windows program you may find it more convenient to call the dummy Windows program DDWDUM. This program runs in the background. It is a 32-bit program and so it will not work on Windows 3.x. It works along the same principals as DDDUM except the command line syntax is different:

DDWDUM Datafile /Kkey /Ooffset

You will find source code and compiled versions of DDDUM and DDWDUM in the EXTERNAL subdirectory.

Technical Specifications

Dinkey Parallel Port

Dimensions : 54mm x 30mm x 17mm
Power consumption active/idle: 1mA (max) / 10µA
Minimum operating voltage: 2.4 volts
Battery: None
Recovery time: None
Maximum number in series: at least 3
Number of programming cycles: >10,000,000
Data retention: > 200 years
Parallel port requirements: 25 pin 'D' connector
Storage temperature: -55°C to +125°C
Operating temperature: 0°C to +70°C
Weight: 28g

Dinkey USB

Dimensions: 61mm x 12mm x 7mm
Power consumption active/idle: 25mA (max) / 20 µA
Battery: None
Number of programming cycles: >10,000,000
Data retention: >200 years
Storage temperature: -65°C to +150°C
Operating temperature: 0°C to 70°C
Weight: 5g

Glossary of Terms

DD_ERRCODE

An environment variable. If set to ON it will force your program to display error code if an error occurs.

DDAdd

A program that adds protection to your program by locking it to one or more Dinkey Dongles. It will also program a Dinkey2 and DinkeyNet dongle.

DDCB

Dinkey **D**ongle **C**hange **B**lock. Use to pass information to and from the DDINFO function in DDChange

DDChange

1. A program that accepts a code generated by DDRemote and will change the protection parameters stored in a Dinkey 2 or DinkeyNet.
2. A function in DDCHANGE.DLL that accepts the update code and makes the relevant changes.

DDCHANGE.TXT

A text file that contains the last update code that was generated by DDRemote.

DDINFO

A function in DDChange.dll that obtains information on all the dongles connected to a computer.

DDLook

A program that searches all the parallel ports for dongles and displays their contents. DOSLOOK is an equivalent program that is written for DOS.

DDMB

Dinkey **D**ongle **M**emory **B**lock. A structure that is used to pass information to and receive information from the dongle.

DDNet

A program that must be run on the dongle server so that the DinkeyNet can be accessed by other machines on the network.

ddp file

Dinkey **D**ongle **P**arameter file. Stores the information that you specified in DDAdd.

ddr file

DDRemote parameter file. Stores the information that you specified in DDRemote.

DDRemote

A program that generates codes for the remote changing of protection parameters stored in a Dinkey 2 or DinkeyNet. It is used in conjunction with DDChange.

DDREMOTE.TXT

A text file that contains a history of all the update codes and a summary of the changes you requested in DDRemote for all the codes that you have generated.

DinkeyNet Path

A directory path of any machine on the network that information can be sent to and received from when communicating with the network dongle.

DLL Method

A Method of protection. You need to modify your code to call a function in the DLL to communicate with the dongle. You must protect the DLL and not your program directly using the object method of protection.

Dongle Number

Each dongle has a unique dongle number. For a Dinkey 1S or Dinkey 2 this can be changed to a number of your choice (numbers must be specified when you order the dongles).

Dongle Server

A machine on the network that has the DinkeyNet attached to it. It can be any machine on the network. It must have the DDNET program loaded for the dongle to work.

Executions

The number of times your protected program (or a feature of your program) can be executed before an expiry message appears.

External Program Method

A method of protection. You must modify your code to call an external program which will communicate with the dongle.

Features

A four-byte field that can be used to store program-specific information.

Last Date Used

The last date that the dongle was successfully called. This value is stored in the dongle and is used to stop the end user from modifying the system clock to beat the expiry date.

Max Days

The number of days (after it is first run) that your program will run before the expiry message is displayed.

Object Method

A method of protection. The relevant object module must be linked to the program before protection is added. You need to modify your code so that it calls a function in the object module that communicates with the dongle.

PRODCODE.INI

A file that tells the workstation where the DinkeyNet path is so that it can access the DinkeyNet dongle. Each workstation must have one of these files if it wants to communicate with the dongle.

Product Code

A string of upto 8 characters that distinguishes one protected product from another. Dinkey 2 and DinkeyNet only.

Product Expiry Message

The message that is displayed if the expiry date has been reached or the execution count has reached zero.

Protection Parameters

The values that are specified in DDAdd and stored in the dongle. They are Executions, Expiry date, Max Days and Features.

SDSN

Software **D**eveloper's **S**erial **N**umber. This serial number is found in Dinkey 1S, Dinkey 2 and DinkeyNet dongles and distinguishes your dongles from those of another software developers.

Secure Signon Message

The message that is displayed when a protection check is successful.

SETUPDRV

A program that installs (or uninstalls) the necessary drivers for the parallel and USB dongles under Windows.

Shell Method

A method of protection. When this method is used you do not need to modify your program at all.

Unauthorised Use Message

The message that is displayed if the protection check has failed for some reason. e.g. the dongle is not present.

Update Number

A number stored in a Dinkey 2 or DinkeyNet that is incremented by one each time a successful DDRemote code is entered. It stops the user from entering the same code twice.

User Data Area

Up to 432 bytes of memory in the dongle that can be used to store data.

User Limit Exceeded Message

The message that is displayed if the number of simultaneous network users is exceeded.

Index

E

F

G

I

L

M

N

O

P

Q

R

S

T

U

W