



MATRIX **LOCK**

HIGH-QUALITY SOFTWARE PROTECTION

USER MANUAL



User Manual

The data and information contained in these documents can be altered without prior notice. Without the express written permission of TDi GmbH, no part of these documents can be reproduced or transmitted for any purpose whatsoever, regardless of how or by which electronic or mechanical means.
The general terms of trade of TDi GmbH apply. Diverging agreements must be made in writing.

Copyright © TDi GmbH TechnoData - Interware. All rights reserved.

WINDOWS is a registered trademark of Microsoft Corporation.
The WINDOWS-logo is a registered trademark (TM) of Microsoft Corporation.

Software License

The software and the enclosed documentation are copyright-protected. By installing the software, you agree to the conditions of the licensing agreement.

Licensing Agreement

TDi GmbH (TDi for short) gives the buyer the simple, exclusive and non-transferable licensing right to use the software on one individual computer or networked computer system (LAN). Copying and any other form of reproduction of the software in full or in part as well as mixing and linking it with others is prohibited. The buyer is authorised to make one single copy of the software as a back-up. TDi reserves the right to change or improve the software without notice or to replace it by a new development. TDi is not obliged to inform the buyer of changes, improvements or new developments or to make these available to him. A legally binding promise of certain qualities is not given. TDi is not responsible for damage unless it is the result of deliberate action or negligence on the part of TDi or its aids and assistants. TDi accepts no responsibility of any kind for indirect, accompanying or subsequent damage.

Compliance with the CE/FCC Regulations



The Matrix device was tested for compliance with the limits for Class B digital devices and approved.

Operation is subject to the following requirements:

1. The device must not cause radiated noise
2. The device must be able to handle radiated noise including noise which can lead to undesired operation

The products complies with the limits according to EN55022 Class B, EN50081-1, EN50082-1 and EN55024.
Alterations to the product without the express approval of TDi GmbH can mean that the CE/FCC regulations are no longer complied with. In this case, the user can lose the right to utilise the product.

PREFACE	
Dear Customer!	6
Who we are	7
 I INTRODUCTION	
General Description	10
Overview of Matrix properties	11
Introduction to cryptography	12
Security levels	14
The “Anti-Hacker” Lock	15
Hardware Models	16
Cross-Platform	17
Architecture	17
 II INSTALLATION	
Installing the software and drivers for Windows	22
Installing the software and drivers for Linux and Mac OS X	25
 III CREATING YOUR PROTECTED PROGRAM	
A few considerations before you begin to protect your application	28
PREPARING THE MATRIX-DONGLE	30
Management and Storage of Data in the Matrix-dongle	30
AUTOMATIC PROTECTION INCLUDED WITHOUT SOURCE CODE	
CHANGES	36
General description	36
Basic settings	37
Protect a program with limited execution time (Demo)	40
Protect a program with unlimited execution time (Not a Demo)	42
Using Matrix-Crypt from the Command-Line	42
MANUAL PROTECTION INCLUDED IN YOUR	
SOURCE-CODE	43
Methods for integration	43
Example for integration into C/C++	44
Example for integration into Visual Basic	48
Example for integration into Pascal	54
Cryptographic authentication of the Matrix-dongle	59
XTEA - encryption reference	62

NETWORK LICENSE-MANAGEMENT	64
Management of network licenses	64
Management of network licenses in the dongle	65
Settings of the MxNET server program	66
Example for the protection of an application in the network with MxNET	69
What you must take into account in your application	72
Advantages and disadvantages of MxNET license management	72
Advantages and disadvantages of license management with one dongle per workplace	73
IV DELIVERY OF PROTECTED PROGRAMS	
Applications for Windows operating systems	76
What you have to Include in the Delivery to your Customers	77
Applications for Linux and Mac OS X operating systems	79
REMOTE UPDATE	80
General Description	80
Benefits of Matrix Remote Update	80
How does Matrix Remote Update work?	81
Basic settings	81
V GUIDLINES FOR GOOD SOFTWARE PROTECTION	
Is there such a thing as an “uncrackable” program?	86
Tools of the cracker	86
Dongle query when the program is started	87
Frequent queries	87
Protective measures during the running of the program	87
Distributing the queries in the code	87
Use of the Matrix memory	87
Working with the cryptography	88
Use the Matrix encryption to authenticate users “on-the-fly”	88
Programming technique	88
Measures to take after detecting an attack	89
VI API-FUNCTIONS REFERENCE	
Overview of API-Functions	92
Description of API-Functions	94
VII MISCELLANEOUS	
Technical Data	136
Prices and Delivery Terms	137
General Terms of Trade	138

Preface

Dear Customer!

Thank you for using *TechnoData Interware* products. We trust that this manual will prove useful as you integrate the Matrix Software Protection System into your programming environment and applications. Please feel free to contact us if you have any questions, comments or suggestions regarding this manual.

This manual contains general and specific information regarding software protection and the integration of *Matrix* into your applications. You will find the most up-to-date version, examples and readme files for Mac OS X and Linux plus additional tools for optimizing your work with Matrix on our website **www.tdi-matrix.com**. We are always interested to hear your comments and suggestions.

You can contact us via E-Mail: support@tdi-matrix.com

Note: Please always include the version and model number when addressing questions to our hotline!

Who we are

As successful manufacturers of software programs, we know that licensing conditions are often not taken as seriously as they should be by clients. Checks and other sophisticated copy protection systems such as the generation of unit-dependent license numbers have been circumvented again and again.

As we could not find a reasonably priced and sufficiently secure copy protection system on the market, we developed our own solution to the problem of illegal copies: *Matrix*.

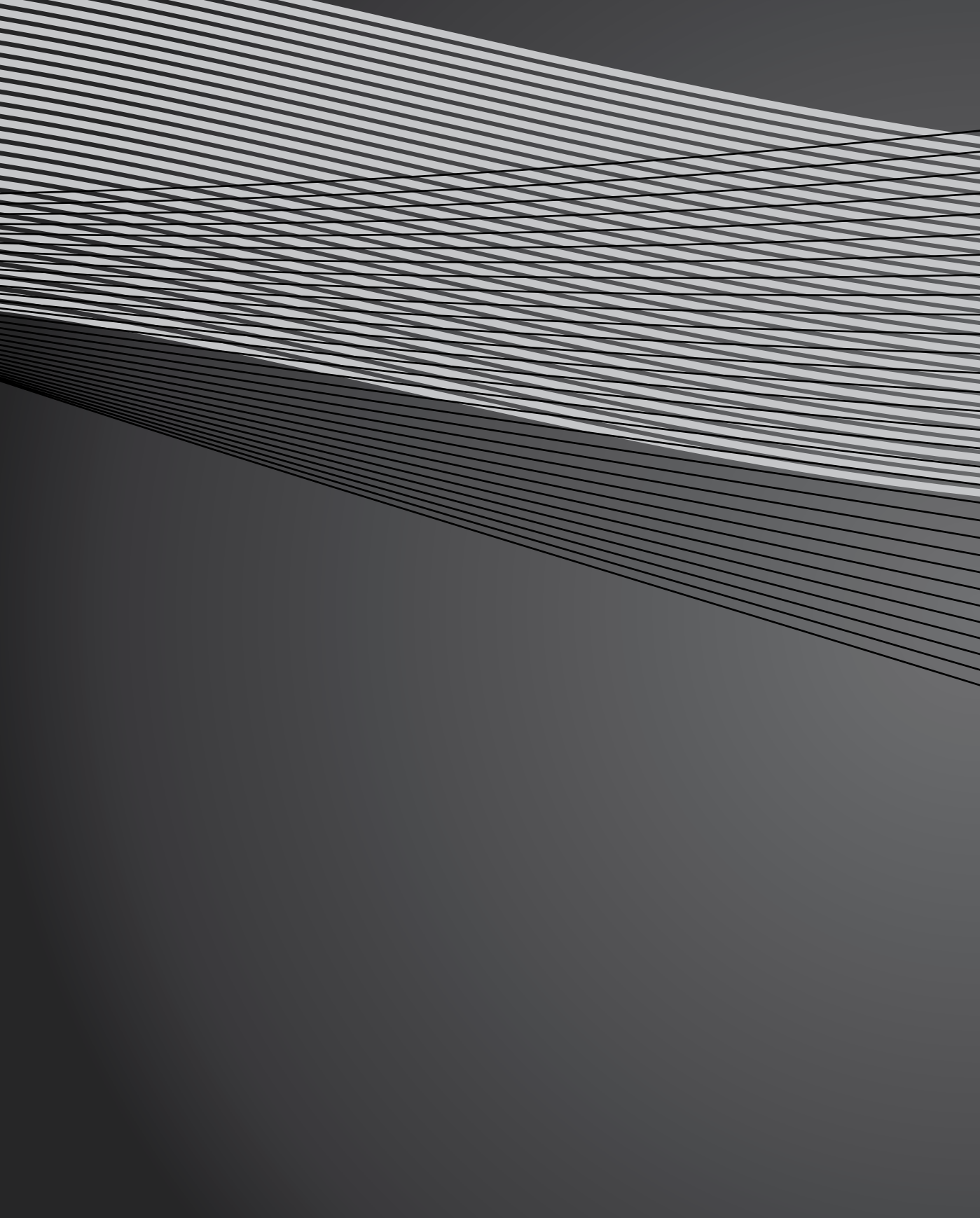
Matrix is already being used in thousands of cases, in many different configurations world wide and successfully protects our own development investments. Since the introduction of the dongle, our clients have placed a remarkable number of follow-up orders, which can only mean that they continue to use it in new products.

Various strategies for circumventing dongles, for example the simulation of the hardware via auxiliary programs or the simulation of communication between the dongle and the protected programs, are doomed to failure due to the coding methods used for *Matrix*.

The values supplied by the dongle cannot be predicted; they are altered at each request by parameters generated at random. We know of no practical instance in which it was possible to crack one of our programs. Certainly we can say that the effort required to defeat the *Matrix*-dongle by potential crackers would be infeasible.

Profit from our experience and successfully protect your development investments with *Matrix*.

Note: Due to our extremely reasonable pricing policy, software in the low and medium price range can now be protected effectively from unauthorized and unremunerated distribution.



The background features a series of horizontal, wavy lines in shades of gray and white, creating a sense of movement and depth. A large, white, stylized number '1' is positioned on the right side of the image.

1

Introduction

Introduction

General Description

Matrix is a reliable security system to protect your software from unauthorized copying. The use of Matrix develops with the ideas and the creativity of the developer. It has a wide range of applications, especially in the field of surveillance, evaluation and distribution support (Pay By Use etc.). **Matrix** can also be used as the basis for IT security and data protection! **Matrix** also permits the hardware-based protection of all kinds of electronic components.

Matrix was developed for both the LPT printer interface and the USB interface of the PC:

- The Matrix-dongles of the ML and the MK series for the printer interface are simply plugged into the LPT interface, where they function perfectly without causing problems for peripherals (printer, scanner etc.) connected downstream.
- The USB Matrix-dongles of the MLU and the MKU series are functionally identical to the LPT dongles and also allow software protection for laptops and PCs which have no parallel printer interface.

During development, particular attention was paid to ensuring **transparent behavior**, a **high degree of security** through the use of a RISC processor, **easy integration** with your software and **high reliability** in practical use.

The Matrix-dongles can also be stacked, meaning that several dongles can be connected to the LPT and/or USB interface at the same time. Even if several dongles are connected to the same interface, it is of course possible to address each dongle individually.

The DLL's included with the delivery ensure easy integration in any programming language whatsoever.

Overview of Matrix properties

The Matrix software protection system offers the following advantages:

- Available for both the LPT and the USB interface
- Creation of demo versions
- Storage of permanent data in the Matrix-dongle
- Use of a single Matrix-dongle to protect several applications
- Management of network licenses with a single Matrix-dongle
- The same hardware is used to protect single-workplace and network programs
- 128-bit encryption and decryption of data through the Matrix-dongle
- Freely selectable 128-bit key for the encryption/decryption
- The 128-bit key cannot be read from the Matrix-dongle
- Cross-Platform (Windows, Linux, Mac) with no proprietary USB drivers
- API support for 16, 32 and 64-bit programs
- Protection of 32-bit EXE files without the need for any programming
- Secure and flexible changes in your customer's dongles via "Remote-Update"
- Multiple Matrix-dongles can be stacked
- Several memory sizes available
- Unequalled value for money
- The same Dongle can be used for software-protection and for Web-Logon

Introduction to cryptography

The earliest encryption methods were very simple. A replacement rule was defined: the message to be encrypted was taken and the letters were replaced by a letter coming elsewhere in the alphabet. This made important messages illegible for unauthorized persons, and they could only be made legible again if the replacement rule was known.

Encryption and decryption

Unencrypted data is known as clear text. In order to protect this data from unauthorized access, it is encrypted, i.e. converted into an unintelligible character string using an encrypting method. Decrypting turns this character string back into the original clear text.

What is cryptography?

Confidential data, i.e. data which is to be transmitted via the Internet, cannot be sent in the form of clear text because it is vulnerable to interception. Encryption ensures that even if the data is intercepted, it cannot be read by unauthorized persons. Methods for making data secure are the realm of the science of cryptography, which also includes crypto-analysis. In crypto-analysis, an attempt is made to recognise structures within the encrypted data and thus determine the appropriate decryption key.

Cryptography uses mathematical methods for the decryption and encryption of data.

Degree of encryption

The degree of encryption is measured in terms of the time and effort required to carry out encryption. Ideally, the encryption method creates an encrypted text which is very hard to decrypt without a suitable decryption method.

According to some theories, it is possible to have a degree of encryption so high that it could not be cracked before the end of the universe, even using all of the computer capacity available at present. However, this does not mean absolute security, and perhaps one of the next computer generations will prove these theories wrong.

No matter how much encryption is used, there is no doubt that conservative, cautious handling of sensitive data will remain an indispensable element of data protection.

Matrix uses a potent method of cryptography: With the help of a 128-bit key, the data to be protected by a XTEA routine is encrypted 32 times in succession. This means that the data is encrypted, the data encrypted in this way is encrypted again and then again and again, 32 times. According to our present knowledge of crypto-analysis, sixfold encryption is considered sufficiently secure at present.

The data is also subjected to the decryption routine 32 times.

How does cryptography work?

A mathematical routine, also known as an encryption algorithm, is used to encrypt and decrypt the data. As an input, this algorithm receives the data to be encrypted and a key, for example a number. The result, i.e. the encrypted text, is calculated from the clear-text input data. The same clear text yields different encrypted texts for each different key used.

The type of algorithm and the degree of secrecy of the encryption key determines the degree of data security.

The encryption key and the encryption algorithm are stored in the Matrix hardware.

Conventional encryption

Conventional encryption is encryption using symmetrical or secret keys. Here the same key is used for the encryption or decryption of data. The XTEA ("eXtended Tiny Encryption Algorithm") algorithm of the Matrix hardware is a conventional encryption system.

Key management and conventional encryption

The advantage of conventional encryption is its speed: this method is very rapid. However, the problem is secrecy and the transmission of the encryption key to the receiver. If the key is intercepted during transmission, it can be used for decrypting the encrypted data at any time and the intended protection is worthless. The weak point with this method is therefore the path of the key or of transmission to the receiver – how can this problem be solved?

The Matrix encryption concept and key management

With Matrix, the problem of key distribution was solved in such a way that encryption or decryption always takes place in the securest part of the whole protection component, i.e. in the Matrix-dongle. This means that the application only transmits data packets to the Matrix-dongle for encryption/decryption but never the key itself. The key, which must be subject to very strict secrecy, is only found in the Matrix-dongle and thus never becomes visible, either at the Application DLL interface or at the LPT/USB interface, and cannot be read from the Matrix-dongle in any way: *"What is not transmitted cannot be intercepted and misused"*.

Security levels

The ML and MK models

The LPT and USB Matrix-dongles are divided up into two model series each, the ML series and the MK series.

Both series offer the same features. However, for an even a higher degree of security, data can only be saved in the Matrix-dongles of the MK series by means of a *MasterKey*-dongle.

The MasterKey-dongle is plugged into the LPT or USB port together with the Matrix-dongle. This dongle is uniquely manufactured once for each customer who orders the MK series. This ensures that the data in the dongle can only be altered by the person who has a valid MasterKey.

The Matrix-dongles are delivered with a customer code called the *UserCode*. This customer code cannot be altered, thus ensuring that each software manufacturer can only program his own dongles. The customer code is assigned once for the first order and remains unchanged for subsequent orders.

For all series, the communication with the PC is encoded. The data exchanged between the PC and the dongle in both directions is encoded before transmission and not decoded until evaluation. Decoding takes place on the PC side and on the dongle side.

The encryption algorithm for communication with the PC changes constantly, thus making the information exchanged worthless for the unauthorized computer hacker.

128-bit encryption and decryption of data

The dongles from ML/MK-60 and MLU/MKU-60 onwards allow the internal encryption and decryption of data via a 128-bit key which cannot be read from the dongle. This key can be defined in whatever way the software developer wishes.

The “Anti-Hacker” Lock

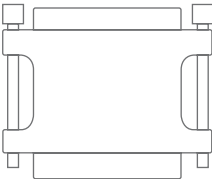
The customer can only read/write data from/to the dongle, if the correct UserCode is specified when calling any dongle function (Read, Write, etc.). If a Hacker tries to deduce the UserCode by calling dongle functions with some wrong UserCodes (e.g. between 1 and nnnnn), the dongle will stop working. In this situation the dongle will lock and will remain locked even if the functions are called with the correct UserCode. Only the software manufacturer can unlock the dongle with their special MasterKey-dongle.

There is no danger of activating the “Anti-Hacker” Lock, if, for example several dongles with different UserCodes are stacked at the LPT and/or USB port and your program tries to find your own dongle in the stack.

Of course your program will try to read the other dongles with your UserCode, but this will not activate the “Anti-Hacker” Lock. This only activates if a hacker tries to read the same dongle a number of times (e.g. in a loop from 1 to nnnn) with constantly wrong and different UserCodes.

Hardware Models

For the management of your software licenses, demo versions etc., we offer several Matrix-dongle models with different memory sizes:



Models for the LPT interface		
ML-12	12 bytes	Storage of a maximum of 3 numerical codes (unsigned 32-Bits)
ML-60	60 bytes	Storage of a maximum of 15 numerical codes (unsigned 32-Bits) 128-Bit encryption and decryption of data
ML-316	316 bytes	Storage of a maximum of 79 numerical codes (unsigned 32-Bits) 128-Bit encryption and decryption of data
MK-Series		Identical to the models of the ML-Series. However, data storage requires an additional security dongle (MasterKey).



Models for the USB interface		
MLU-60	60 Byte	Storage of a maximum of 15 numerical codes (unsigned 32-Bits) 128-Bit encryption and decryption of data
MLU-316	316 Byte	Storage of a maximum of 79 numerical codes (unsigned 32-Bits) 128-Bit encryption and decryption of data
MKU-Series		Identical to the models of the MLU-Series. However, data storage requires an additional security dongle (MasterKey).

All Matrix-dongles can be stacked, which means that a series of several dongles can be plugged into a single PC's USB port or the same LPT port. This does not require any changes to be made to the parameters of the dongle. Stacking is no problem whatsoever and each dongle in the stack can be individually addressed.

Cross-Platform

Matrix-dongles support two USB operating modes:

- With a proprietary USB driver, called “Driver-Mode”
- With no proprietary driver, called “HID-Mode”

These two operating modes are relevant for Windows, given that only the “HID” USB-devices are implemented with no proprietary drivers.

Matrix USB-dongles can be used in all operating systems (Windows, Linux, Mac) with no specific proprietary USB drivers:

- On Linux in both operating modes with no proprietary USB drivers.
- On Windows for the “HID-Mode” with no proprietary USB drivers.
- On Mac OS X in both operating modes with no proprietary USB drivers.

The Matrix USB-dongle can be set to the desired operating mode.

If you set the dongle to “HID-Mode” your customer wouldn’t need to install any driver at all.

Architecture

Communication with your software takes place via the supplied Matrix-API:

- DLL files under Windows
- Shared-Libraries under Linux and Mac OS X.

Matrix protects on Windows 16-, 32- and 64-bit programs. 16-bit programs which have to run on Windows NT/2000/XP/Vista can also be protected by Matrix without any problems whatsoever as the transition from 16 to 32 bits is automatically controlled by the Matrix DLLs.

LPT interface

Your program controls the addressing of the Matrix-dongle connected to the LPT port via a VXD driver under Windows 95/98 and via a SYS driver under Windows NT/2000/XP/Vista.

The use of the VXD driver under Windows 95/98 is optional - addressing can also be carried out directly via the hardware and without a VXD driver.

USB interface

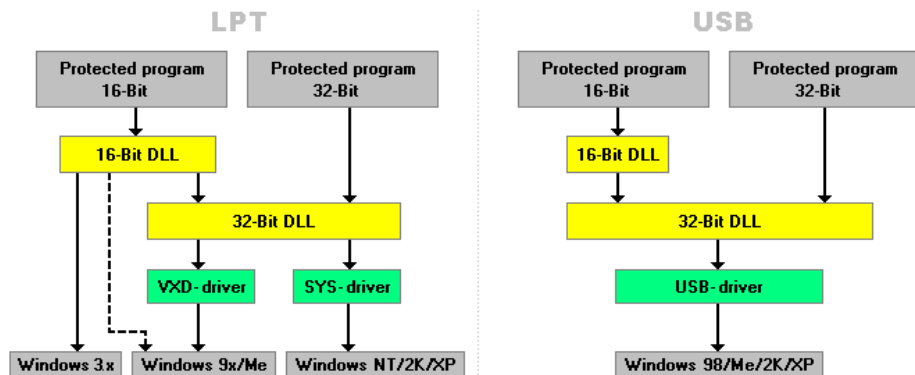
The Matrix USB-dongles have initially been designed to operate with an additional driver which is provided in the installation kit.

Starting from hardware version 5.0, Matrix USB-dongles have a second operating mode, the “HID-Mode”.

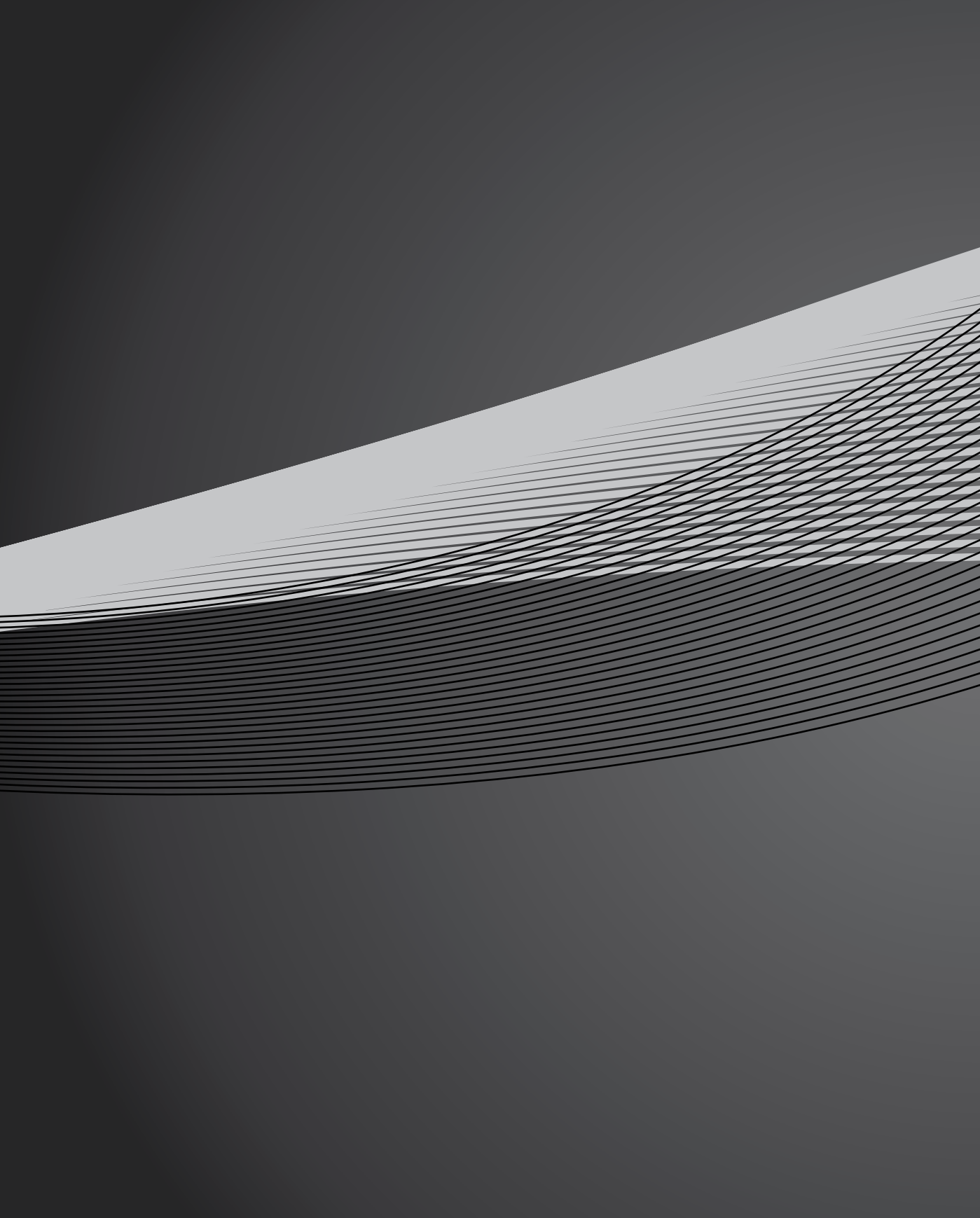
- In “HID-Mode” the Matrix-dongle is recognized automatically by the operating system when the dongle is connected. The advantage in this operating mode is that no driver needs to be delivered or installed.
- In “Driver-Mode” the dongles need an additional driver provided in the installation kit. This driver must be delivered and installed on the target computer. In this operating mode the USB-dongle is slightly faster as in “HID-Mode”.

You have the possibility to choose between the two operating modes using the programming tool provided with the Matrix installation kit.

The following diagram explains how it works and how 16- and 32-bit programs protected via Matrix function under Windows 3.x, 95/98, NT, 2000 and XP/Vista.



Under Windows 95/98 communication with the LPT-Dongle can take place without the VXD-Driver. However, in order to avoid conflicts when accessing the LPT interface, communication via the VXD driver is recommended.





2

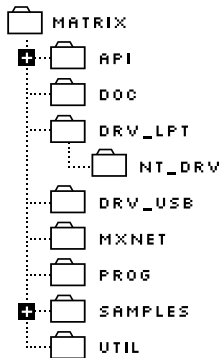
Installation

Installation

Installing the software and drivers for Windows

Start the **setup.exe** program from the diskette/CD.

On completion of installation, you will find the directory **\matrix** with the following structure on your disk:



API Application Programming Interface

This subdirectory contains the 16-, 32- and 64-bit API with the corresponding libraries which you require for integration into your software.

DOC

You can find this user manual in PDF format [here](#).

DRV_LPT Driver LPT

This directory contains the LPT drivers for Win9x/ME and NT/2000/XP/Vista.

NT_DRV

This directory contains the installation program for the Windows NT/2000/XP/Vista LPT driver.

DRV_USB Driver USB

This directory contains the USB driver for Win98/ME/2000/XP/XP-64/Vista and the installation program for the USB driver.

MXNET

Here you can find the MxNet server program for managing network licenses.

PROG

This directory contains the programs which you require in order to program your Matrix-dongles.

SAMPLES

Here you can find examples of integrating of protection in various programming languages.

UTIL

Here you can find the MxCheck program you can use to check the installed driver and API on the system. You can deliver this program to your customer as support tool.

The installation routine automatically copies the Matrix-API (matrix16.dll and matrix32.dll) and the LPT driver (iwport.vxd) for Windows 9.x/ME into the directory **\windows\system** and the LPT driver (iwport.sys) for Windows NT/2000/XP/Vista into the directory **\windows\system32\drivers**.

LPT driver installation under Windows 95/98/ME

Under Windows-9x/ME, communication with the Matrix-dongle on the LPT port takes place via the appropriate VXD driver. As it is not necessary to expressly install and register this driver, it is enough to copy the driver file (iwport.vxd) into the **\windows\system** directory.

LPT driver installation under Windows NT/2000/XP/Vista

Under Windows-NT/2000/XP/Vista, communication with the Matrix-dongle on the LPT port takes place via the appropriate SYS kernel driver which must be installed and registered.

The driver can be installed automatically or manually:

- Automatic installation is carried out by the Matrix-API when the program is started for the first time on condition that the driver file (iwport.sys) is present in the **\windows\system32\drivers** directory and you possess administration rights. The integrated installation function simplifies the installation process considerably. It is therefore sufficient to copy the driver file (iwport.sys) into the **\windows\system32\drivers** directory. The only requirement is that the customer must possess administration rights when the program is started for the first time.
- Manual installation/uninstallation of the driver can be carried out via the corresponding installation program (drv_inst.exe) from the **\nt_drv** directory. For additional information on the drv_inst.exe program, please see the Readme file in the **\nt_drv** directory.

Please bear in mind that only users with the necessary access rights, such as the administrator, can install drivers under Windows NT/2000/XP/Vista.

USB driver installation under Windows 98/ME/2000/XP/Vista

The Matrix USB-dongles supports by default the “Driver-Mode”. In this operating mode it is necessary to install the proprietary USB driver provided in the installation kit.

Starting from hardware version 5.0, the USB-dongles supports additionally the operating in “HID-Mode”. In this operating mode a driver installation is not required anymore. The dongle will therefore use the HID-driver which is already included in the operating system. On Windows 2000/XP/Vista the HID-drivers are included once they have been installed. Windows 98/ME don't initially include the HID-driver and the operating system will ask for the Windows installation CD when a HID device is connected for the first time.

In order to have both operating modes functional, the setup program will also install the proprietary Matrix USB driver.

We will describe next, the detailed Matrix USB driver installation steps:

As the Matrix USB dongle is a Plug&Play unit, when you connect it for the first time the system will ask you to specify a suitable driver.

In the Plug&Play wizard then you have to select the directory in which the USB driver is located (for example the `\drv_usb` directory).

It is enough to install the driver once under Windows 98/ME/2000. Windows then automatically finds and installs the driver, even if the Matrix USB dongle is plugged into another USB connection (for example another HUB slot).

However, under Windows XP/Vista you are requested to reinstall the driver for every available USB slot. This annoying repeated installation of the driver can be avoided if the INF file of the USB driver is pre-installed. For this purpose, the `inf_inst.exe` program is available in the `\drv_usb` directory.

Pre-installation of the INF file tells the Windows hardware manager where the USB driver is located. Even after pre-installation of the INF file, the Plug&Play wizard appears, but it is then no longer necessary to indicate the location of the driver. It is enough to click the **Continue >** button and everything is done automatically.

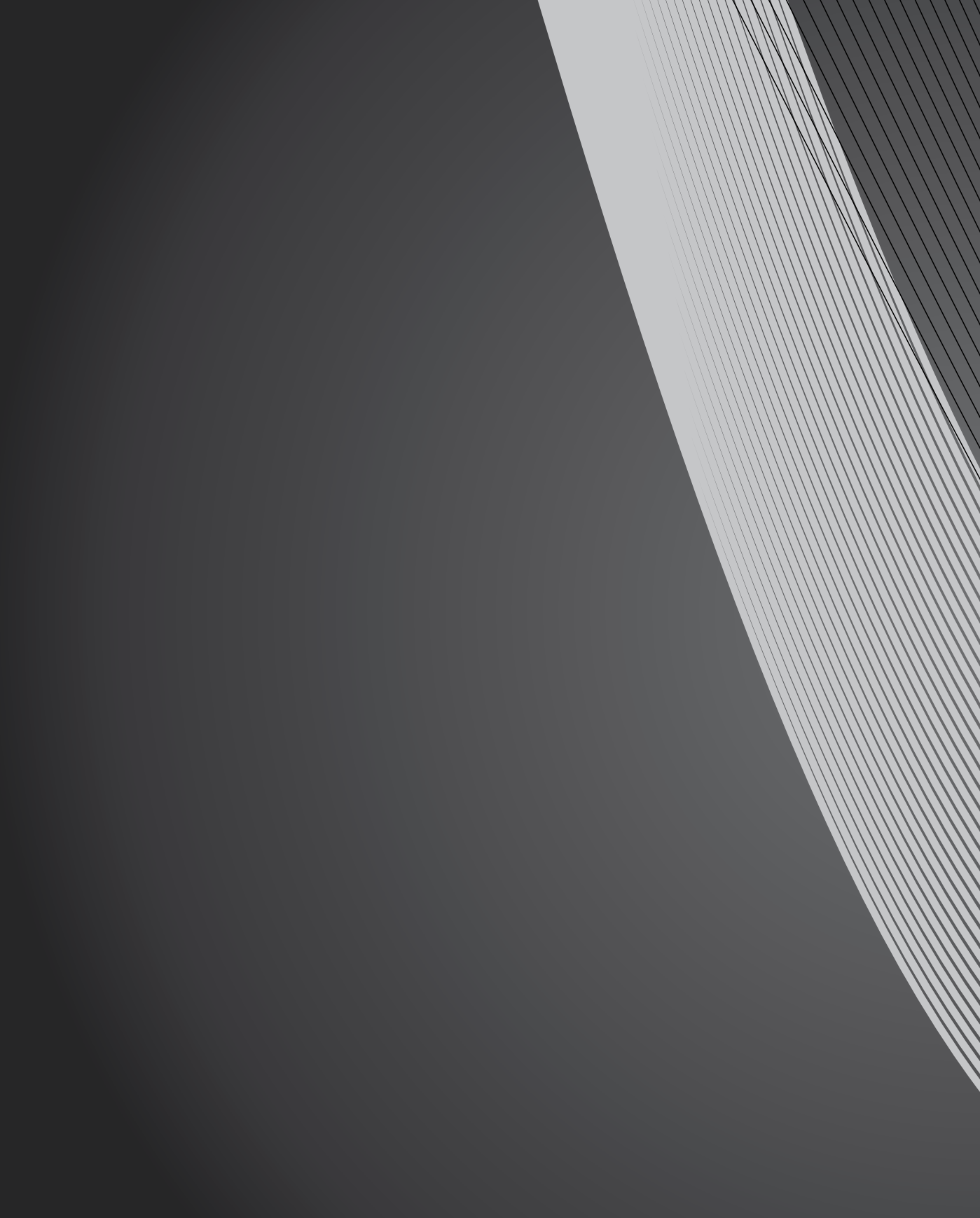
Note: We recommend pre-installing the INF file in all Windows versions.

For additional information on the `inf_inst.exe` program, please read the Readme file in the `\drv_usb` directory.

Installing the software and drivers for Linux and Mac OS X

For Linux and Mac-OSX, please follow the installation steps from the corresponding Readme file.

Note: You will find the newest versions of the API and tools available for download at www.tdi-matrix.com



3

Creating your protected program

Creating your protected program

A few considerations before you begin to protect your application

The *Matrix Security System* provides you with several methods which you can use to integrate protection into your application:

Automatic protection (without source code changes)

Is the easiest and fastest method to protect 32-bit Windows applications without modifying their source code.

Manual protection (in your source code)

Allows the implementation of protection in the source code of your application via API-calls.

You may choose one method or combine both, depending on your requirements. The following comparison table should help you to decide which solution is the right one for your needs.

Automatic protection

No source code needed.

Quick and easy protection without any programming efforts, is also the easiest way to build demo versions of your program, for example with a limited number of runs. But only a fixed set of features is available.

Easily insert mechanisms to prevent your application from being tampered with or cracked.

When the Matrix-dongle is not connected, the protected application will stop.

Allows you to integrate protection in standalone applications. For network applications, you have to use one Matrix-dongle for each workstation.

Manual protection

Source code must be available.

Requires more effort, but offers you the maximum flexibility. Through integration into your source code you are able to decide which parts of your software are active and which are not. You deliver the same software to all customers and save time and administration costs.

You have to include your own methods to avoid “reverse engineering”, or you can also use Automatic Protection on top of your own coding.

You can define your own scheme of how your program should react when the Matrix-dongle is not present: stop your program, disable menu items to reduce its functionality, switch to demo mode or whatever you need.

You can include network license management for the entire network using just one Matrix-dongle.

You can use the power of encryption through hardware to create Access control, Two-Factor-Authentication, Secure Business-to-Business solutions and so on.

Recommendation: We recommend the manual protection through integration into your source code. This method requires more effort, but offers you the maximum flexibility. You can define your own scheme of how your program should react when the Matrix-dongle is not present.

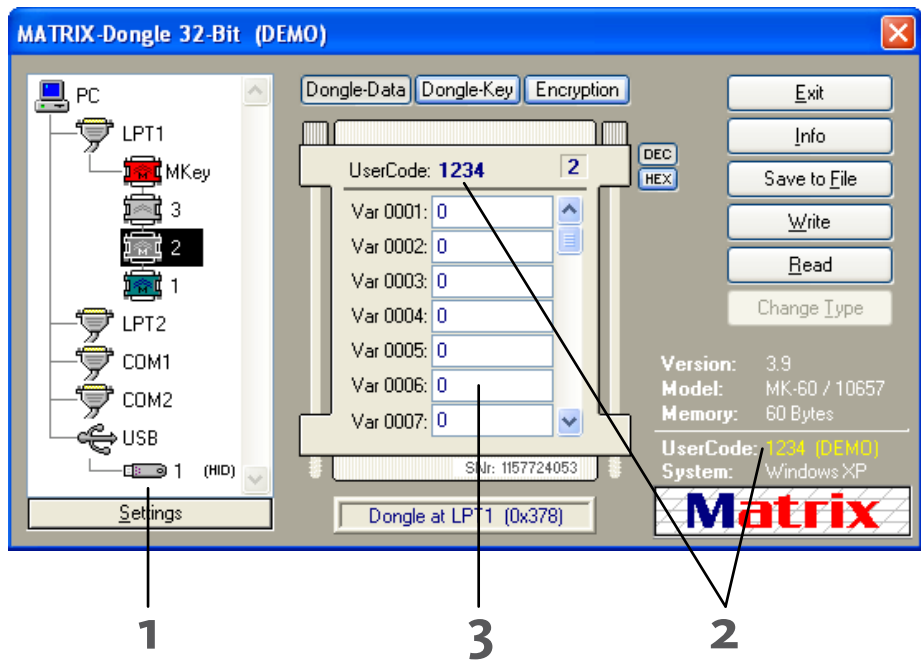
The network functionality can only be implemented through the manual protection method.

Preparing the Matrix-dongle

Management and Storage of Data in the Matrix-dongle

After installation, a convenient program is provided for the management of your data in the Matrix-dongle.

The management program is supplied in a 16-bit and a 32-bit version. Users running Windows 3.x can also program the Matrix-dongles without any problems whatsoever.



This program is extremely easy to operate.

1. The LPT and USB ports available in the PC and the connected Matrix-dongles are shown on the left-hand side. The display can be updated at any time via the "Read" button.
2. Customer no. "UserCode"
The customer no. entitled "UserCode" is assigned to each Matrix user at his first order and cannot be altered. This number limits access to data in just his own dongles.

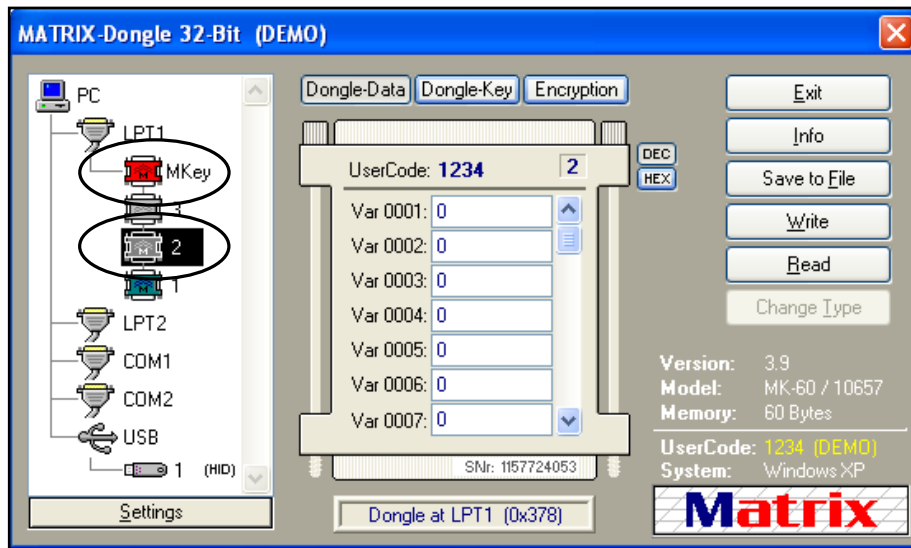
- The dongle fields (Var001-VarXXX) can be used as you wish and can contain numbers in the range 0 - 4294967295 (hex: 0x00 - 0xFFFFFFFF). The number of fields available depends on the memory size. The “DEC/HEX” buttons allow you to switch to decimal or hexadecimal view.
You can save the values with the “Write” button.

Note: The memory size of the dongle divided by 4 gives the maximum number of variables which can be saved.

Example: With a dongle of type ML-60 with 60 bytes, $60 / 4 = 15$ variables are available.

Special Function of the MK series

Dongles of the MK series (“MasterKey” Series) can only be programmed when a MasterKey-dongle is plugged in during the programming process. When the MK series is ordered, the MasterKey-dongle is included in the delivery.

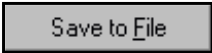


When the MK series is programmed, the MasterKey-dongle must be plugged into an LPT or USB port of the computer. Only then is write access available to the dongles to be programmed.

The presence of the MasterKey-dongle is also displayed on the left-hand side. MK series dongles are marked with an **M** in order to avoid confusion.


 Write

The “*Write*” button saves the data in the dongle highlighted on the left.


 Save to File

The “*Save to file*” button saves the dongle data in an ASCII file. To save typing work, this file can be imported in the *Matrix-Crypt* encryption program.


 Change Type


The “*Change Type*” button will allow you to set the USB-dongle to the desired operating mode: “HID-Mode” or “Driver-Mode”.


 Info

The “*Info*” button gives you the version number of the Matrix API and drivers present in the system.


 Settings

The “*Settings*” button allows the type of access to the LPT ports to be set.


 Dongle-Key Encryption

The “*Dongle Key*” and “*Encryption*” buttons allow you to switch between the different program functions. These are described in more detail on the following pages.

Access to the LPT port

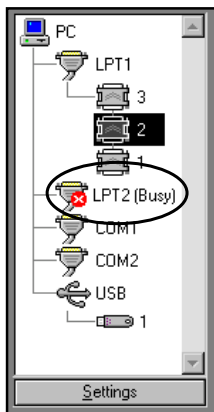
The Matrix driver only occupies the LPT interface if it is free. If the interface is currently occupied by other devices such as printers, scanners, etc., the Matrix driver waits for the interface to be released. The wait time is set to 10 seconds by default.

If the interface becomes free within the waiting time, the Matrix driver acquires the interface with the dongle for the duration of communication and releases it again afterwards.

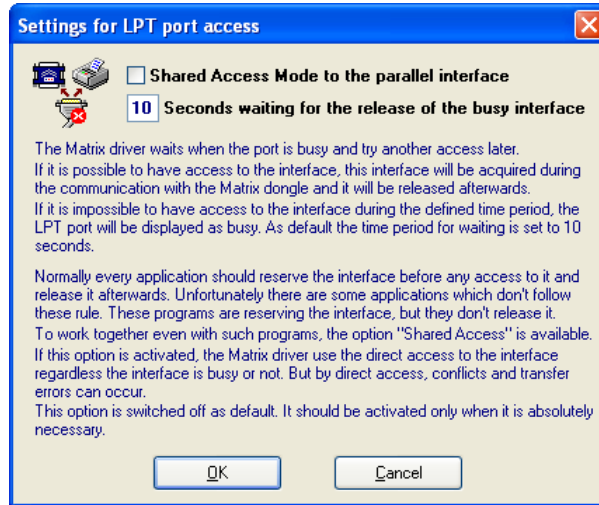
If the interface does not become free within the waiting time, it is marked as occupied (“Busy”).

Before access to an interface, an application should always first acquire it and then release it again. Unfortunately there are some programs which do not observe this rule and acquire the interface but do not release it again. In order to be able to communicate with the dongle in spite of problems of this kind, the driver is able to access the interface even if it is busy (“Shared Access Mode”).

However, this kind of access to the interface can cause access conflicts and transmission failure. The default setting is therefore “off”.



The “Settings” button switches on this option, but it should only be activated if absolutely necessary. If the option “Shared Access Mode” is switched off, i.e. if the driver correctly only acquires the interface when it is free, the waiting time in seconds can be set here.



These settings are saved in the matrix.ini file located in the `\windows\system` directory.

This file can also be delivered along with your protected application so that you can also alter these settings for your customer if necessary.

The **matrix.ini** file has the following structure:

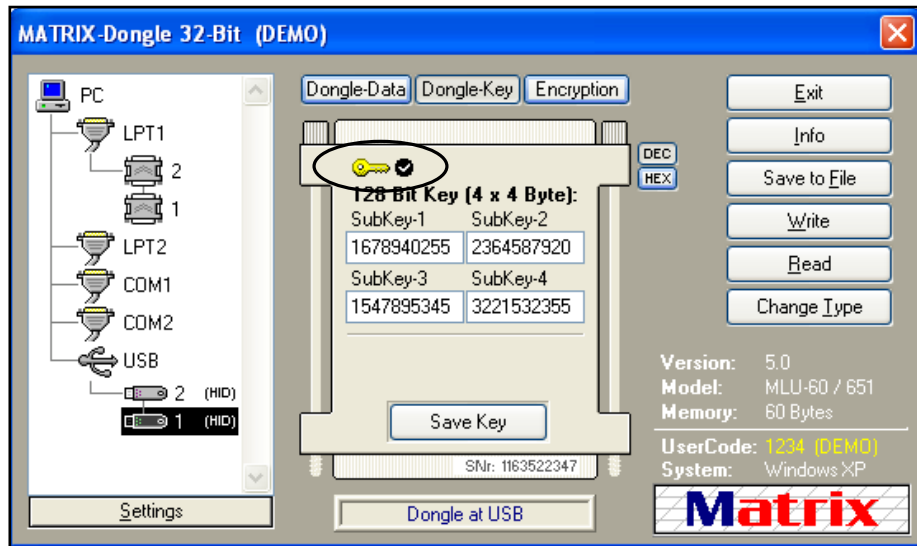
```
[LPT-PORT]
AccessLPT=ON
DirectAccess=OFF
AccessTime=10

[USB-PORT]
AccessUSB=ON
AccessTime=8
```

The settings AccessLPT=ON and AccessUSB=ON can be turned OFF to disable the support and check for the specified interface. It can be helpful to turn off the LPT support on some computers or laptops which have no parallel printer interface.



The Dongle-Key function allows you to save an encryption key consisting of 128 Bits (4x4 bytes) for use with the encrypt/decrypt function of the Matrix-dongle. For security reasons, a key is no longer displayed after being saved.



If a key has already been saved, the yellow key symbol appears above the input fields. Keys are deleted when the “Save Key” button is operated with empty input fields.

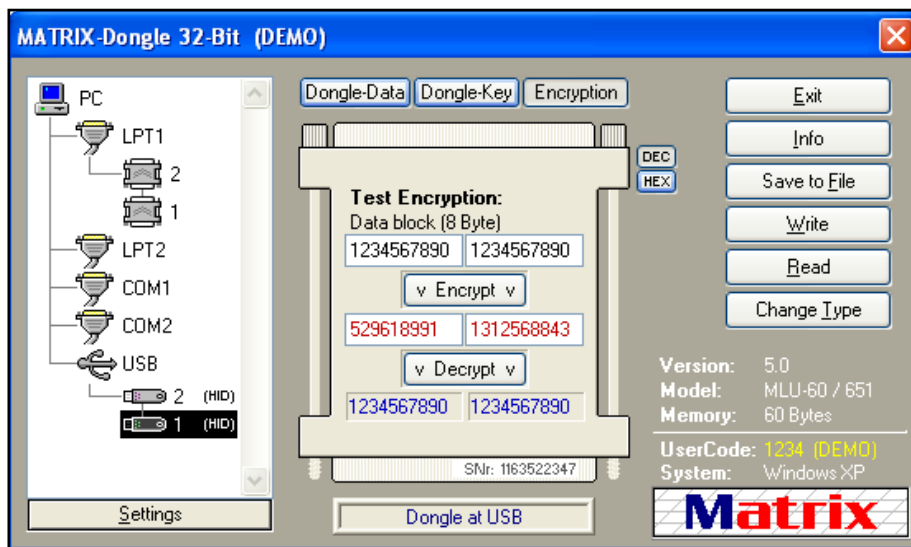
A “Zero-Key” (i.e. with all key bytes = 0), is not interpreted as a valid key.

Encryption/decryption could also be carried out via a “Zero-Key”, but this would not offer any special degree of security.

Note: The key is saved in the currently connected Matrix-dongle and cannot be read from it.



The encrypt/decrypt function of the Matrix-dongles can be tested via the Encrypt function. The XTEA encryption reference table in this manual can be used for testing.



After entry of the 8-byte clear data block, the “Encrypt” button is pressed and the result of encryption is displayed in the centre fields. After the “Decrypt” button is pressed, the clear data must reappear in the bottom data fields.

This function can also be used in order to encrypt constants / program parameters and use the encrypted values in the program to be protected. The contents are not converted back into meaningful values until the program is run. For example encrypted constants, calculation variables etc., can be integrated into your program.

In order to ensure that your program continues to function effectively, the Matrix-dongle must always be connected to the computer while the program is running, as the values can only be decrypted via the Matrix-dongle.

Note: The encrypt/decrypt function is not available for the LPT models ML-12/MK-12. All other LPT models support this function from hardware version 2.1 onwards. All USB models include the described encrypt/decrypt function.

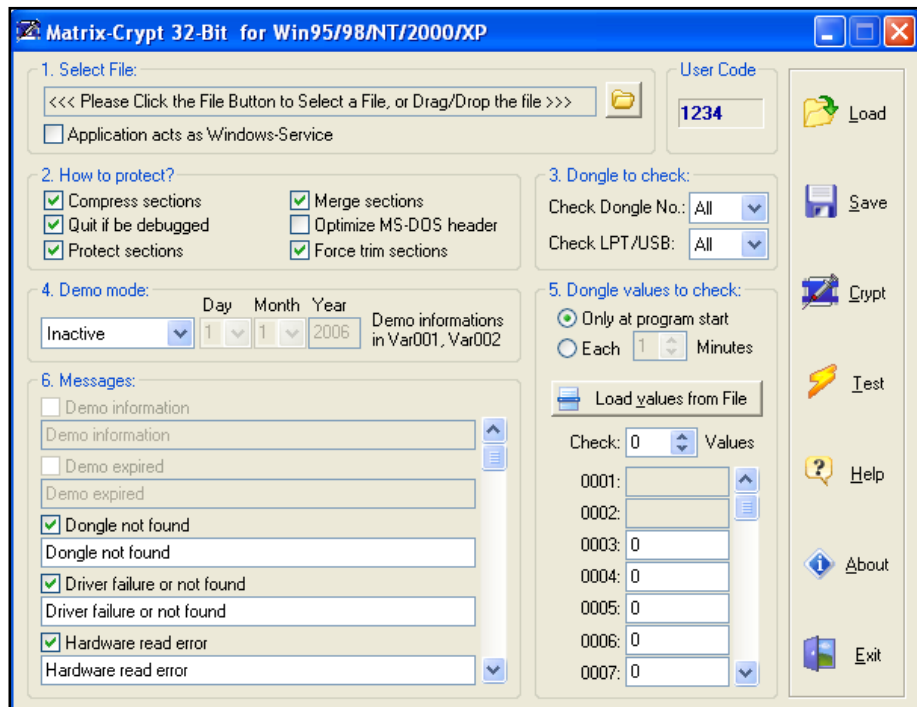
Automatic protection included without source code changes

General description

Matrix-Crypt for Win32 is a professional software protection tool. With Matrix-Crypt you can easily integrate Matrix protection into your Windows 95/98/NT/2000/XP/Vista 32bit EXE and DLL applications. In particular, you don't need to edit any source code or have any programming experience to do this. Everything is fully automatic, so even if you are not a programmer, you can protect any program.

With **Matrix-Crypt** you can:

- Integrate the dongle checks into your software.
- Protect your software from “Reverse Engineering” and analyzing.
- Integrate “Anti Memory Dump” protection into your software.
- Protect your application with “Two Secure Encryption Layers”.
- Make a demo version of your software with execution and date limitations.
- Make your Matrix protected software with unlimited execution.



To protect your program, select via the “File” button the EXE/DLL file which you would like to protect.

If your application is a Windows service, then you must enable the option “Application acts as Windows-Service”, otherwise the encrypted service cannot be handled correctly by the Windows Service Manager. This option is not available for DLLs and will be disabled automatically when you select a DLL file.

To make Matrix-Crypt easier to work with, you can save the entire working area with the selected settings in a project via the “Save” button.

If you want to re-encrypt a program at a later date, you can load the project in question via the “Load” button. This function is for example especially helpful in the mass production of demo versions or of programs with a modular structure and dongle-controlled module release.

Basic settings

The following basic settings are available to you in Matrix-Crypt:

“How to protect”

In this section you can select how you would prefer to protect your program against “reverse engineering”:

Compress sections

This compresses the entire code segment of your program. Using this option, you can decrease the size of your encrypted program.

Quit if debugged

This protects your program against debug (SoftICE). The program ends automatically when an attempt is made to analyse it with the debugger. To ensure highest security for your protected application, this option is always ON and cannot be disabled.

Protect sections

This encrypts the entire code segment of your program. To ensure highest security for your protected application, this option is always ON and cannot be disabled.

Merge sections

When your application includes virtual sections (with no section body), then this section will be merged with the previous one. Using this option, the size of your encrypted program will also be decreased.

Optimize MS-DOS header

This optimizes the header of your program. Using this option, you can decrease the size of your encrypted program.

Force trim sections

In most cases compilers puts at end of section unused bytes (tail, padding bytes). This option removes unused bytes at end of sections which will result in a better compression ratio and size reduction of your encrypted program.

Note: We recommend you, to start with minimum protection options. If everything is working fine, then try the next level of protection, by enabling additional options.

Each additional level of protection adds more checks and increase the protection of your application.

“Dongle to check”*Check Dongle No.*

If you wish, you can also explicitly specify the dongle number when several dongles are stacked. The “All” setting causes the program to automatically search for the dongle in the whole dongle stack (if there are more than one).

Check LPT/USB

This allows you to specify which port your dongle is connected to. The “All” setting causes your program to automatically search for the dongle on all available LPT/USB ports.

“Dongle values to check”*Only at program start*

If you select this option, your protected program will check the dongle once at program start.

Each ___ Minutes

To increase the protection of your program, you can select this option and enter here the time interval between dongle checks. In this case, your protected program will check the dongle every “n” minutes. This option is not available for DLLs.

Check___ Values

The contents of the variables in the dongle from Var003 onwards can also be checked when your program runs. Specify the number of values you want to check in “Check ___ Values”. If the number of comparison variables are not specified, in “Check ___ Values”, only the UserCode is checked. If the existence of a dongle with a correct UserCode has been confirmed, the protected program can be run. We recommend also to check values and not rely solely on the UserCode.

The variables Var0001 and Var0002 are reserved for the creation of demo programs with limited execution time.

“Messages”

Your own messages which you would like to have displayed when the program is run can be stored in the fields of this section.

Protect a program with limited execution time (Demo)

Several settings are available to you for the creation of demo programs protected by the Matrix dongle. The variables Var0001 and Var0002 in the dongle are reserved in the demo modes. Select one of the following types of demo in the section “Demo-Mode”:

Number of Runs

This allows the number of possible program runs to be specified. When you prepare your dongle, you state the permitted number of runs in the variable Var0001 of the dongle via the Matrix programming program. After each run of the protected program, the number of possible runs is reduced by one and written back to the dongle memory. When the contents of the variable reaches 0, the running time of the demo is finished and the program can no longer be run.

Number of Days

This setting can be used to limit the running time of your demo program to a certain number of days. When preparing your dongle, you state the permitted number of days in the variable Var0001 of the dongle via the Matrix programming program. The variable Var0002 must contain the value 0. When the demo program is first run by the user, this type of encoding automatically stores the date in the variable Var0002 of the dongle. This date then marks the start of the demo time with the program user.

The number of days in Var0001 is automatically reduced by one and written back to the dongle memory. When the contents of Var0001 reaches 0, the running time of the demo is finished and the program can no longer be run.

Valid thru

In this demo mode, the running time of the program is permitted up to a certain date. You can specify the date in the fields “Day”, “Month”, “Year”. When you prepare your dongle, you must set the content of the variables Var0001 and Var0002 to 0.

Warning: We don't recommend the Demo-Mode “Valid thru”, because it may be vulnerable to the PC system date manipulation. Use – as often as possible – the Demo-Mode “Number of Runs”.

Note: Also the demo mode “Number of Runs” can be used to limit the run time of your application to a certain number of days.

Following example will show you how to use “Number of Runs” to limit the run time to a certain number of days:

- we consider for this example that the user starts your application an average of 5 times per day.
- when you like to limit the run time of your application to 30 days, this results into a total of $5 \times 30 = 150$ starts for this time period.
- you can use the demo mode “Number of Runs” with 150 starts and your application will run nearly the estimated time period. Carefully you can give a little more and permit for example 160 instead of 150 starts.

Inactive

This mode allows the encoding of programs which were not created for demo purposes and are to remain completely functional.

For further details please see: “Protect a program with unlimited execution time”.

Protect a program with unlimited execution time (Not a demo)

In order to create programs protected by the Matrix-dongle and which are completely functional (not demos), you simply set the demo mode to “Inactive”.

The contents of the variables in the dongle from Varoo3 onwards can also be checked when your demo program runs. If comparison variables are not specified, only the UserCode is checked. If the existence of a dongle with a correct UserCode has been confirmed, the protected program can be run.

After selecting the desired settings, click onto the “Crypt” button in order to integrate the dongle request into your program. When your program is encoded, the original unprotected EXE/DLL file is always stored as a backup under the name programname.exe.old.

*Warning: You should always test your program thoroughly before delivery. The “Test” button is available for this, allowing you to run the protected program.
You should always move the backup file (original unencrypted file) to a safe place!*

Using Matrix-Crypt from the Command-Line

To integrate the encryption of EXE/DLL files into your own build environment, you can use Matrix-Crypt as command line tool:

```
mx-crypt.exe -p ProjectFile.mcp [-e File.exe] [-o OutMessageFile]
```

-p ProjectFile.mcp
settings should be used.

Important: The project file must be specified with full path.

-e File.exe

This parameter is optional. Using this parameter, you can specify which EXE/DLL file should be encrypt instead of the EXE/DLL file specified in the project file.

Important: The EXE/DLL file must be specified with full path.

-o OutMessageFile

This parameter is optional. Using this parameter, you can redirect display messages to the specified file.

Manual protection included in your source code

Methods for integration

Several functions are available to you to integrate the Matrix-dongle into your application. There are two programming methods:

Static

This method can be used for programs that can link in static libraries (e.g. VC++). For the static method, the API functions are located in the 16-bit and the 32-bit LIB.

Dynamic

This method can be used for programs that can call a DLL (e.g. VC++, Delphi, VisualBasic, Access etc.). For the dynamic method, the API functions are located in the 16-bit and the 32-bit DLL.

The following files are available to you for integration:

Dynamic link

16-bit

matrix16.h
matrix16.lib
matrix16.dll

32-bit

matrix32_64.h
matrix32.lib
matrix32.dll

64-bit

matrix32_64.h
matrix64.lib
matrix64.dll

Static link

16-bit

mxst16.h
mxst16.lib

32-bit

matrix32_64.h
mxst32.lib

64-bit

matrix32_64.h
mxst64_vc8o.lib

The following examples demonstrate you how to integrate the API-calls in the most popular programming languages: C/C++, Visual Basic, Pascal/Delphi. The primary purpose of these examples is to be readable and understandable and not to provide the highest level of security.

Please keep this in mind when implementing protection in your application. Make use of the information contained in the chapters “Cryptographic authentication of the Matrix-dongle” and “Guidelines for good software protection”.

In the chapter “API-Functions Reference” you will find a detailed description for each available API function.

More samples for different programming languages are also available in the directory **\matrix\samples**.

Example for integration into C/C++

```
#include "matrix32.h"

long  DataIn[256];    /* Buffer to read the dongle data      */
long  DataOut[256];   /* Buffer for data to be stored      */
long  DataCrypt[2];   /* Buffer for data to be encrypted   */
short RetCode;        /* Return value                      */
long  DNG_Version;    /* Dongle version number            */
long  DNG_SerNr;      /* Unique serial number of the Dongle */
short DNG_Port;       /* LPT/USB-Port                     */
short DNG_LPTADR;     /* Adress of LPT port               */
short DNG_Count;      /* Number of dongles at LPT/USB port */
short DNG_Mem;        /* Memory size of dongle            */
short DNG_MaxVar;     /* Maximum number of data fields     */
short i;

/*-----*/
/* Set the port (LPT or USB) for following calls.      */
/* Possible values for DNG_Port are:                   */
/* 1-3 = LPT1-LPT3, 'U' or 85 = USB                    */
/*-----*/
DNG_Port = 1;    /* 85 = USB */

/*-----*/
/* Init the Matrix-API.                                */
/*-----*/
RetCode = Init_MatrixAPI();
if(RetCode < 0)
{
    printf("Init_MatrixAPI Return-Code: %d", RetCode);
}

/*-----*/
/* Check whether LPT1 is available.                     */
/*-----*/
DNG_LPTADR = GetPortAdr(1);    /* 1 = LPT1 */
if(DNG_LPTADR == 0)
{
    printf("LPT1 is not available!");
    Release_MatrixAPI();
    exit;
}
```

```

/*-----*/
/* Determine the number of dongles at LPT/USB. */
/*-----*/
DNG_Count = Dongle_Count(DNG_Port);
if(DNG_Count == 0)
{
    printf("Matrix-dongles not found at LPT/USB!");
    Release_MatrixAPI();
    exit;
}

/*-----*/
/* The memory size of the last dongle at LPT/USB. */
/*-----*/
DNG_Mem = Dongle_MemSize(DNG_Count, DNG_Port);
if(DNG_Mem == 0)
{
    printf("Memory size could not be read!");
    Release_MatrixAPI();
    exit;
}
DNG_MaxVar = DNG_Mem / 4;          /* Number of data fields */

/*-----*/
/* Read dongle version from last dongle at LPT/USB. */
/*-----*/
DNG_Version = Dongle_Version(DNG_Count, DNG_Port);
if(DNG_Version <= 0)
{
    printf("Version number could not be read!");
    Release_MatrixAPI();
    exit;
}
printf("Dongle's Version-No: %d.%d", HIWORD(DNG_Version),
                                             LOWORD(DNG_Version));

```

... example C/C++
continued

... example C/C++
continued

```

/*-----*/
/* Read the unique serial number of the last dongle at */
/* LPT/USB with UserCode 1234 and display.           */
/*-----*/
DNG_SerNr = Dongle_ReadSerNr(1234, DNG_Count, DNG_Port);
if(DNG_SerNr < 0)
{
    printf("Error while reading serial number!");
    Release_MatrixAPI(); exit;
}
printf("This dongle have the Serial-No: = %ld", DNG_SerNr);

/*-----*/
/* Read 15 data fields from last dongle at LPT/USB   */
/* with UserCode 1234 and display.                   */
/*-----*/
RetCode = Dongle_ReadData(1234, DataIn, 15, DNG_Count,
                           DNG_Port);

if(RetCode < 0)
{
    printf("Error while reading data!");
    Release_MatrixAPI(); exit;
}
for(i=0; i<15; i++)
{
    printf("Value %d = %ld", i+1, DataIn[i]);
}

/*-----*/
/* Save 15 data to LPT/USB in last dongle via UserCode */
/* 1234. The values 101, 102...115 are saved as an    */
/* example.                                           */
/*-----*/
for(i=0; i<15; i++)
{
    DataOut[i] = 101 + i;
}
RetCode = Dongle_WriteData(1234, DataOut, 15, DNG_Count,
                           DNG_Port);

if(RetCode < 0)
{
    printf("Error while writing data!");
    Release_MatrixAPI(); exit;
}

```

```

/*-----*/
/* Encrypt Data over the dongle and display the */
/* encrypted result. */
/* For example the Data Block '1234567890 1234567890' */
/* will be encrypted. */
/* The key which is stored in the dongle will be used */
/* to encrypt the Data Block. */
/*-----*/
DataCrypt[0] = 1234567890;
DataCrypt[1] = 1234567890;
RetCode = Dongle_EncryptData(1234, DataCrypt, DNG_Count,
                             DNG_Port);

if(RetCode < 0)
{
    printf("Error while encrypting data");
    Release_MatrixAPI();
    exit;
}
printf("Encrypted Data = %lu : %lu", DataCrypt[0],
      DataCrypt[1]);

/*-----*/
/* Close the Matrix-API. */
/*-----*/
Release_MatrixAPI();

```

*... example C/C++
continued*

Example for integration into Visual Basic

```

Type DNGINFO
    LPT_Nr As Integer
    LPT_Adr As Integer
    DNG_Cnt As Integer
End Type

Declare Function Init_MatrixAPI Lib "matrix32.dll"
    () As Integer

Declare Function Release_MatrixAPI Lib "matrix32.dll"
    () As Integer

Declare Function PausePrinterActivity Lib "matrix32.dll"
    () As Integer

Declare Function ResumePrinterActivity Lib "matrix32.dll"
    () As Integer

Declare Function GetPortAdr Lib "matrix32.dll"
    (ByVal DNG_LPT As Integer) As Integer

Declare Function GetVersionAPI Lib "matrix32.dll"
    () As Long

Declare Function GetVersionDRV Lib "matrix32.dll"
    () As Long

Declare Function GetVersionDRV_USB Lib "matrix32.dll"
    () As Long

Declare Function Dongle_Find Lib "matrix32.dll"
    () As Integer

Declare Function Dongle_FindEx Lib "matrix32.dll"
    (ByRef xBuffer As DNGINFO) As Long

Declare Function Dongle_Count Lib "matrix32.dll"
    (ByVal DNG_Port As Integer) As Integer

Declare Function Dongle_Version Lib "matrix32.dll"
    (ByVal DNG_Nr As Integer, _
    ByVal DNG_Port As Integer) As Long

```

```
Declare Function Dongle_Model Lib "matrix32.dll"  
    (ByVal DNG_Nr As Integer, _  
     ByVal DNG_Port As Integer) As Integer  
  
Declare Function Dongle_MemSize Lib "matrix32.dll"  
    (ByVal DNG_Nr As Integer, _  
     ByVal DNG_Port As Integer) As Integer  
  
Declare Function Dongle_ReadData Lib "matrix32.dll"  
    (ByVal UserCode As Long, _  
     ByRef DataIn As Long, _  
     ByVal MaxVar As Integer, _  
     ByVal DNG_Nr As Integer, _  
     ByVal DNG_Port As Integer) As Integer  
  
Declare Function Dongle_WriteData Lib "matrix32.dll"  
    (ByVal UserCode As Long, _  
     ByRef DataOut As Long, _  
     ByVal MaxVar As Integer, _  
     ByVal DNG_Nr As Integer, _  
     ByVal DNG_Port As Integer) As Integer  
  
Declare Function Dongle_ReadSerNr Lib "matrix32.dll"  
    (ByVal UserCode As Long, _  
     ByVal DNG_Nr As Integer, _  
     ByVal DNG_Port As Integer) As Long  
  
Declare Function Dongle_EncryptData Lib "matrix32.dll"  
    (ByVal UserCode As Long, _  
     ByRef DataCrypt As Long, _  
     ByVal DNG_Nr As Integer, _  
     ByVal DNG_Port As Integer) As Integer
```

*... Visual Basic
example continued*

... Visual Basic
example continued

```

Dim RetCode      As Integer
Dim xBuffer      As DNGINFO
Dim DataIn(256)  As Long
Dim DataOut(256) As Long
Dim DataCrypt(2) As Long
Dim DNG_Version  As Long
Dim DNG_SerNr    As Long
Dim DNG_Port     As Integer
Dim DNG_LPTADR   As Integer
Dim DNG_Nr       As Integer
Dim DNG_Count    As Integer
Dim DNG_Mem      As Integer
Dim DNG_MaxVar   As Integer
Dim DataBlock(2) As Long
Dim VerMajor     As Integer
Dim VerMinor     As Integer
Const Shift16 = 2 ^ 16

'*****
'* Set the port (LPT or USB) for following calls.          *
'* Possible values for DNG_Port are:                       *
'* 1-3 = LPT1-LPT3, 85 (ASCII 'U') = USB                   *
'*****
DNG_Port = 1      '*** 85 = USB ***

'*****
'* Init the Matrix-API.                                     *
'*****
RetCode = Init_MatrixAPI()
If DNG_LPTADR < 0 Then
    MsgBox "Init_MatrixAPI Return-Code: " & RetCode
End If

'*****
'* Check whether LPT1 port is available.                   *
'*****
DNG_LPTADR = GetPortAdr(1)      '*** 1 = LPT1 ***

If DNG_LPTADR = 0 Then
    MsgBox "LPT1 is not available!"
End If
MsgBox "The adress of LPT1 is: " & Hex(DNG_LPTADR)

```

```

'*****
'* Search for number of dongles at LPT/USB.      *
'*****
DNG_Count = Dongle_Count(DNG_Port)

If DNG_Count = 0 Then
    MsgBox »No dongle available at Port: » & DNG_Port
End If
MsgBox »Found: » & DNG_Count & » dongles at Port: » & DNG_Port

DNG_Nr = DNG_Count

'*****
'* Read memory size of last dongle at LPT/USB and calculate*
'* the maximum number of data fields.                *
'*****
DNG_Mem = Dongle_MemSize(DNG_Nr, DNG_Port)

If DNG_Mem = 0 Then
    MsgBox "Error while determining dongle memory!"
End If
DNG_MaxVar = DNG_Mem / 4
MsgBox "Number of variables of this dongle: " & DNG_MaxVar

'*****
'* Read dongle version from last dongle at LPT/USB.      *
'*****
DNG_Version = Dongle_Version(DNG_Nr, DNG_Port)

If DNG_Version <= 0 Then
    MsgBox "Error while reading dongle version!"
Else
    VerMinor = CInt(DNG_Version And 65535)
    VerMajor = CInt(DNG_Version \ Shift16)
    MsgBox "The dongle have the version number: " & VerMajor _
        & "." VerMinor
End If

```

... Visual Basic
example continued

... Visual Basic
example continued

```

'*****
'* Read the unique serial number of the last dongle at      *
'* LPT/USB with UserCode 1234 and display.                  *
'*****
DNG_SerNr = Dongle_ReadSerNr(1234, DNG_Nr, DNG_Port)

If DNG_SerNr < 0 Then
    MsgBox »Error while reading serial number!«
Else
    MsgBox »This dongle have the Serial number: » & DNG_SerNr
End If

'*****
'* Read 15 data fields from last dongle at LPT1 via        *
'* UserCode 1234 and display.                                *
'*****
RetCode = Dongle_ReadData(1234, DataIn(0), 15, DNG_Nr, _
                        DNG_Port)    If RetCode < 0 Then
    MsgBox "Error while reading the dongle data!"
End If
For i = 0 To 14
    MsgBox "Content of variable: " & i + 1 & ": " & DataIn(i)
Next i

'*****
'* Save 15 data in last dongle at LPT/USB via UserCode      *
'* 1234. The values 101,102...115 are saved as an example.   *
'*****
For i = 0 To 14
    DataOut(i) = 101 + i
Next i
RetCode = Dongle_WriteData(1234, DataOut(0), 15, DNG_Nr, _
                        DNG_Port)

If RetCode < 0 Then
    MsgBox "Error while writing dongle data!"
Else
    MsgBox "The dongle data have been written successfully!"
End If

```

```

'*****
'* Encrypt Data over the dongle with the key stored in the *
'* dongle and display the encrypted result. *
'*****
DataCrypt(0) = 1234567890
DataCrypt(1) = 1234567890
RetCode = Dongle_EncryptData(1234, DataCrypt, DNG_Nr, _
                                DNG_Port)

If RetCode < 0 Then
    MsgBox »Error while encrypting data!«
Else
    MsgBox »Encrypted Data: » & DataCrypt(0) & » » _
        & DataCrypt(1)
End If

'*****
'* Close the Matrix-API. *
'*****
Release_MatrixAPI()

```

... Visual Basic
example continued

Example for integration into Pascal

```

program demo;

{$N+}

uses WinCrt, matrix16;

var
  DataIn      : array[1..256] of longint;
  DataOut     : array[1..256] of longint;
  DataCrypt   : array[1..2] of longint;
  xBuffer     : array[1..3] of DNGINFO;
  RetCode     : Integer;
  DNG_Version : longint;
  DNG_SerNr   : longint;
  DNG_Port    : Integer;
  DNG_LPTADR  : Integer;
  DNG_Count   : Integer;
  DNG_Nr      : Integer;
  DNG_Mem     : Integer;
  DNG_MaxVar  : Integer;
  i           : Integer;
  VarMajor    : Integer;
  VarMinor    : Integer;

begin
  {*****}
  { * Set the Port (LPT or USB) for following calls.          * }
  { * Possible values for DNG_Port are:                        * }
  { * 1-3 = LPT1-LPT3, 85 (ASCII 'U') = USB                    * }
  {*****}
  DNG_Port := 1;      {*** 85 = USB ***}

  {*****}
  { * Init Matrix-API.                                         * }
  {*****}
  RetCode := Init_MatrixAPI();

  if RetCode < 0 then
  begin
    write('*** Init_MatrixAPI returned with error code: ');
    writeln(RetCode);
  end;

```

```

{*****}
{* Check whether LPT1 is available.          *}
{*****}
DNG_LPTADR := GetPortAdr(1);    {*** 1 = LPT ***}

if DNG_LPTADR = 0 then
begin
    writeln('*** LPT1 not available!');
    Release_MatrixAPI();
    exit;
end;

{*****}
{* Search for number of dongles at LPT/USB.    *}
{*****}
DNG_Count := Dongle_Count(DNG_Port);

if DNG_Count = 0 then
begin
    writeln('Matrix-dongles not found at LPT/USB!');
    Release_MatrixAPI();
    exit;
end;

DNG_Nr := DNG_Count;

{*****}
{* Read memory size of last dongle at LPT/USB and      *}
{* calculate maximum number of data fields.            *}
{*****}
DNG_Mem := Dongle_MemSize(DNG_Nr, DNG_Port);

if DNG_Mem = 0 then
begin
    writeln('Error while reading memory size!');
    Release_MatrixAPI();
    exit;
end;

DNG_MaxVar := DNG_Mem div 4;    {* Number of data fields *}

```

... Pascal

example continued

... Pascal
example continued

```
{*****}
{* Read dongle version from last dongle at LPT/USB.      *}
{*****}
DNG_Version := Dongle_Version(DNG_Nr, DNG_Port);

if DNG_Version = 0 then
begin
    writeln('Error while reading version number!');
    Release_MatrixAPI();
    exit;
end;

VerMinor := (DNG_Version and 65535);
VerMajor := (DNG_Version shr 16);
write('This dongle have the version number: ');
write(VerMajor);
write('.');
writeln(VerMinor);

{*****}
{* Read the unique serial number of last dongle at LPT/USB*}
{* with UserCode 1234 and display.                        *}
{*****}
DNG_SerNr := Dongle_ReadSerNr(1234, DNG_Nr, DNG_Port);

if DNG_SerNr < 0 then
begin
    writeln('Error while reading serial number!');
    Release_MatrixAPI();
    exit;
end;

write('This dongle have the Serial-No.: ');
writeln(DNG_SerNr);
```

```

{*****}
{* Read 15 data fields from last dongle at LPT/USB      *}
{* via UserCode 1234 and display.                      *}
{*****}
RetCode := Dongle_ReadData(1234, @DataIn, 15, DNG_Nr,
                           DNG_Port);

if RetCode < 0 then
begin
    writeln('Error while reading data!');
    Release_MatrixAPI();
    exit;
end;

for i := 1 to 15 do begin
    write('DataIn[i] = ');
    writeln(DataIn[i]);
end;

{*****}
{* Save 15 data in last dongle at LPT/USB via UserCode *}
{* 1234. The values 101,102...115 are saved as an example.*}
{*****}
for i := 1 to 15 do begin
    DataOut[i] := i;
end;
RetCode := Dongle_WriteData(1234, @DataOut, 15, DNG_Nr,
                            DNG_Port);

if RetCode < 0 then
begin
    writeln('Error while writing data!');
    Release_MatrixAPI();
    exit;
end;

writeln('The dongle data have been written successfully!');

```

... Pascal

example continued

... Pascal
example continued

```
{*****}
{* Encrypt Data over the dongle and display the encrypted *}
{* result. *}
{* For example the Data Bolck '1234567890 1234567890' will *}
{* be encrypted with the Key stored in the dongle. *}
{*****}
DataCrypt[1] := 1234567890;
DataCrypt[2] := 1234567890;

RetCode := Dongle_EncryptData(1234, @DataCrypt, DNG_Nr,
                               DNG_Port);

if RetCode < 0 then
begin
    writeln('Error while encrypting data!');
    Release_MatrixAPI();
    exit;
end;

write('Encrypted Data: ');
write(DataCrypt[1]); write(' : '); writeln(DataCrypt[2]);

{*****}
{* Close the Matrix-API. *}
{*****}
Release_MatrixAPI();
```

Cryptographic authentication of the Matrix-dongle

A hacker will usually try to emulate the Matrix driver/API. In order to repel such attacks, an authentication of the Matrix-dongle is necessary. This procedure guarantees that the acknowledgements actually come from the Matrix-dongle and not from a simulating driver.

The safe authentication of the Matrix-dongle is based on the following logic: In the protected application (EXE), a 64-Bit random number (in practice two assembled 32-Bit random numbers) is produced. This random number is then encrypted in two ways with the same key.

1. The first way is via the encryption function `MxApp_Encrypt` contained in your application. The 64-Bit random number will be encrypted with this function using a 128-Bit key. The 128-Bit key used inside the application must be the same as the one stored in the Matrix-dongle.
2. The second way is via the Matrix-dongle. The same 64-Bit random number is sent to the Matrix-dongle through the function `Dongle_EncryptData`. The random number will be encrypted inside of the Matrix-dongle with the 128-Bit key stored in the dongle and be returned by the function.

The authenticity is determined by a comparison of the two encrypted results. The key does not show outside the dongle or the application. Therefore it can not be intercepted and abused.

Example

The following program code will demonstrate the authentication of the Matrix-dongle.

```
unsigned long Key[4]={11111,22222,33333,44444};
unsigned long Data_App[2];
unsigned long Data_Dng[2];

/*-----*/
/* 1. 'Clear data' generated with random numbers      */
/*   Initialize two buffers with the same 'Clear data' */
/*-----*/
Data_App[0] = RandomNo_1;
Data_App[1] = RandomNo_2;

Data_Dng[0] = RandomNo_1;
Data_Dng[1] = RandomNo_2;
```

... example continued

```

/*-----*/
/* 2. Encryption of 'Clear data' in the application */
/*-----*/
MxApp_Encrypt(Data_App, Key);

/*-----*/
/* 3. Encryption of 'Clear data' across the Matrix-dongle */
/*-----*/
Dongle_EncryptData(UserCode, Data_Dng, DongleNr, PortNr);

/*-----*/
/* 4. Comparison of encryption results */
/*-----*/
if(Data_App[0] == Data_Dng[0] && Data_App[1] == Data_Dng[1])
{
    //--- Success: both encryption results are identical ---
}

```

Note: The XTEA-128 encryption function MxApp_Encrypt should be integrated in your application. This function is available for C/C++, VB and Delphi in the files “mxtea.h”, “mxtea.bas”, “mxtea.pas” located in the directory \matrix\samples\...

Recommendation

To increase the safety against code attacks the following points should be considered:

- To generate clear data we recommend the use of random numbers. This ensures that there are always different encrypted results which you receive from the Matrix-dongle.
This method should prevent hackers from determining what's really happening.
- To simplify this example the key variables have fixed values. You can increase the code security against “reverse engineering” by assembling the key at run-time with an algorithm.
- Fill the key variables in your application only during encryption and delete the key variables after use.
- To simplify this example the results were checked with a simple comparison. But to increase the code security, you should make the comparison using bit and byte operations.
- Separate the comparison operations from the encoding operations in the code location.
- You should encrypt several sequences of random numbers. Compare the encryption results in other order (mixed) and at different code locations in your program code.

XTEA - Encryption reference

In the following you will find five data blocks with a length of 8 bytes each which were encrypted using the 128-bit example key.

Some languages, such as Visual Basic, do not have native support for the 32-bit unsigned integers (4 bytes) used in the Clear and Encrypted Data blocks.

In this case when you display huge unsigned integers they will be converted to signed integers, meaning you will get negative values.

For example:

the unsigned byte with the value 128 converted to a signed byte is -128

the unsigned byte with the value 129 converted to a signed byte is -127

This is mathematically correct because the unsigned byte range is from 0 to 255 and the range for a signed byte is from -128 to +127. But it can be very confusing.

In such languages, it is recommended to use hexadecimal values to avoid confusion between signed and unsigned. The Hexadecimal range for example for a byte is from 0x00 to 0xFF regardless of the type of the variable (signed or unsigned).

As an aid for the implementation of the Encryption into your software, the following encryption reference are listed in both formats: Unsigned Decimal and Hexadecimal.

Example Key 128-Bit (4 x 4 Bytes):

1111122222	1111133333	2222244444	2222255555	(dec)
423A612E	423A8C95	8474C25C	8474EDC3	(hex)

Clear data (2 x 4 Byte):**Encrypted data:**

1.	1234567890 499602D2	1234567890 499602D2	4264218190 FE2ACE4E	2765200066 A4D19AC2	(dec) (hex)
2.	2233445566 851FACBE	1122445577 42E72909	0128004498 07A13192	2776431824 A57CFCD0	(dec) (hex)
3.	1468205773 57830ACD	0934113682 37AD7192	2273514099 87831273	1600112827 5F5FC8BB	(dec) (hex)
4.	3942612857 E AFF7F79	2557492693 98703DD5	3416279985 CBA04BB1	0347058491 14AFB13B	(dec) (hex)
5.	2670314019 9F29C223	1725837151 66DE2F5F	1447417338 5645D5FA	0381438979 16BC4C03	(dec) (hex)

Network License Management

Management of network licenses

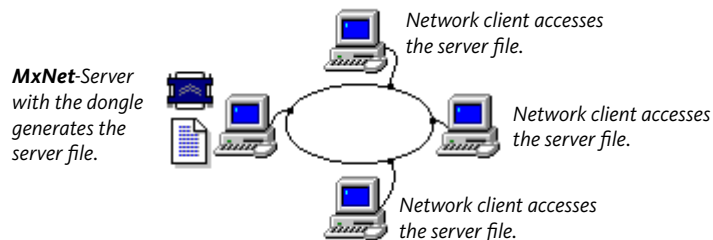
For the protection of software in a network, it is possible to define a certain number of licenses which can be used simultaneously by the user. This function, known as “License Management”, can be implemented in two different ways:

1. Use of one Matrix-dongle at each workplace
2. Use of one Matrix-dongle for the entire network

Version 1) with one dongle per workplace is no different to the protection methods of a program for a single workplace as a locally connected dongle is used. For this reason, this method will not be described in more detail here.

However, version 2) with one dongle for the entire network is based on a slightly different concept than the solution for individual workplaces. This method, known as “MxNet”, allows you to use one dongle in a network which can be connected to any desired workplace. The management of the licenses is carried out via a server file which is generated by the MxNet server program. All clients present in the network can access the server file just as they can a dongle via the Matrix-API.

The mode of operation in the network with the MxNet technology is explained in the following diagram:



The network protection via MxNet does not use network protocols and can thus be used in any desired network system. The server program MxNet which generates the server file runs on the MxNet server (the PC with the dongle). This file is updated at certain defined intervals and is stored in encoded form.

The encoding algorithm changes each time the file is updated and thus provides maximum protection against unauthorised manipulation.

The management of the simultaneously running copies (licenses) of the protected application in the network is carried out via “User-Slots”. A User-Slot is an entry in the server file which is effected by each network client. The User-Slot is occupied by each client for as long as the protected application is running and then released when the application is terminated.

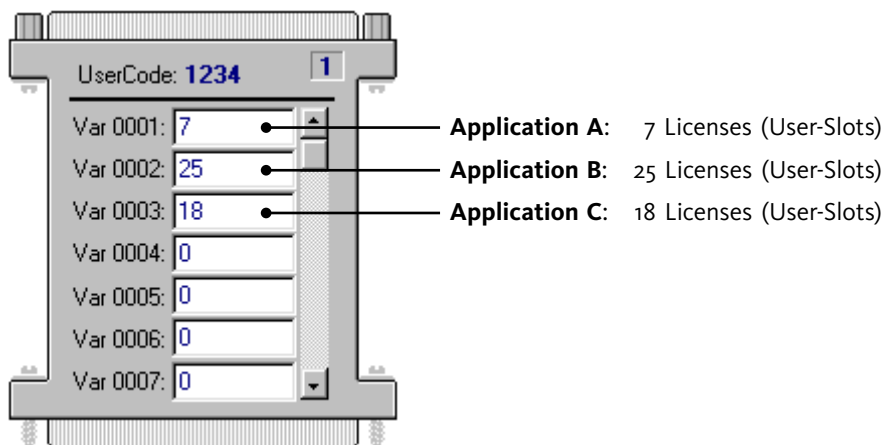
The number of permissible licenses per application is stored in the Matrix-dongle and makes this information available to the MxNet server program.

The management of the licenses with the Matrix-dongle offers maximum flexibility and allows problem-free protection with only one dongle for several applications or applications with a modular structure.

The following section will explain the management of the licenses for three different programs with one dongle only.

Management of network licenses in the dongle

In the dongle, one variable is used for each application. Thus, the number of licenses for each application is stored in these variables. The following illustration shows 7 licenses for the application A, 25 for application B and 18 for application C.



Thus, these application-specific variables, known as “Application-Slots”, contain the number of possible User-Slots managed in the server file. In our example, therefore, for the protected applications a maximum of 7 users for the Application-Slot of application A (Var0001), 25 users for the Application-Slot of application B (Var0002) and 18 users for the Application-Slot of application C (Var0003) can be logged on in the server file. The maximum number of User-Slots per Application-Slot is 32500. If the Application-Slot (entry in the dongle) is larger than 32500, this number is automatically reduced to 32500 in the server file.

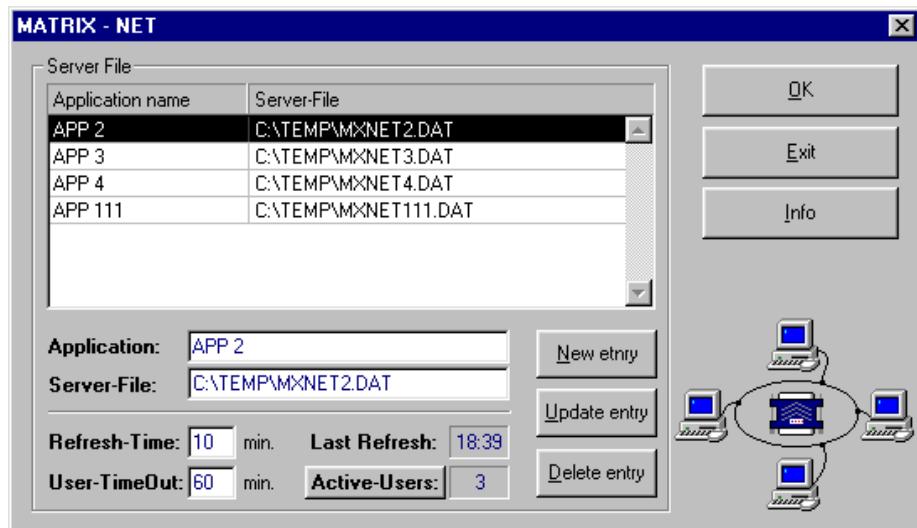
Settings of the MxNET server program

The Matrix-NET program is the MxNet server application and must run on the MxNet server PC (computer with the dongle). This application generates and refreshes the MxNet server file.



After starting the Matrix-NET program a dongle symbol is displayed in the Task bar. Clicking on this icon will activate the Matrix-NET dialog.

The setting options are described in the following section.



If necessary, it is possible to generate a special server file for each application. The server file is entered via the input fields “Application” and “Server File”.

Each entry must consist of “Name of Application” and “Name of Server File”. The server file should always be entered with the absolute path.

Refresh-Time

In this field, the time interval for the refresh of the server file can be set. The last refresh to be carried out is displayed in “Last Refresh”. The refresh period should usually be selected so as to be between 5 and 10 minutes. This specification is global, i.e. it is valid for all server files.

User-TimeOut

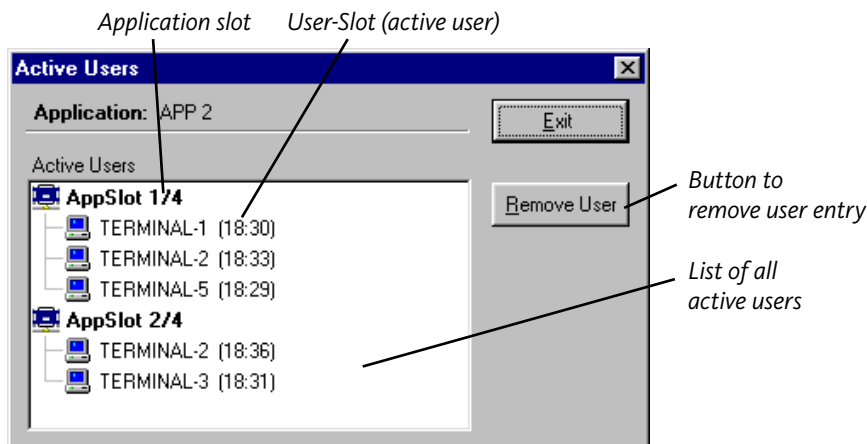
The User-TimeOut can be set in this field. The User-TimeOut is the time limit after which the user is automatically removed from the server file. In the case of abnormal termination of the application (crash) on a client, this function ensures the release of the User-Slot in the server file, as this would otherwise remain occupied. In addition, a user entry can also be removed manually in this way. The User-TimeOut is global, i.e. the TimeOut-Limit set is valid for all server files.

Active-Users

This field is updated constantly and shows the total number of active users for the selected application. The “Active Users” button allows you to display a detailed list of the active users. This list displays the application slots with active user slots.

The application slots are shown in the format Application-Slot number / Dongle number.

AppSlot 1/4 for example means application slot 1 from dongle 4.



This example clearly shows that four or more Matrix-dongles are connected to the MxNet-Server, whereby the application slots 1 and 2 of the 4th dongle are occupied with user entries.

Every occupied User-Slot shows the name of the terminal on which the application is running as well as the time of the last update of the User-Slot by the application.

If your application is terminated abnormally on any terminal, you can either remove the User-Slot from the list manually or wait for the TimeOut. When the TimeOut is reached, the User-Slot is removed automatically from the MxNet program the next time the server file is refreshed.

Note: It is advisable to remove inactive “User-Slots”. from the list as long as the application is active. However, even if a “User-Slot” is deleted by mistake, there is no danger to the functioning of the application.

Important!

It is very important that the system time of the PC is synchronized in the network. Otherwise the “File-TimeOut / User-TimeOut” can not be correctly computed. The maximum allowed deviation of the system time between clients and server, may not exceed the number of minutes, which was selected in the MxNet server program in “Refresh-Time”. The following command can be used to synchronize the system time of the clients with the system time of the server. This command can be implemented in the boot-up procedure of each client to make it an automatic function.

```
NET TIME \\<computername> /SET /YES
```

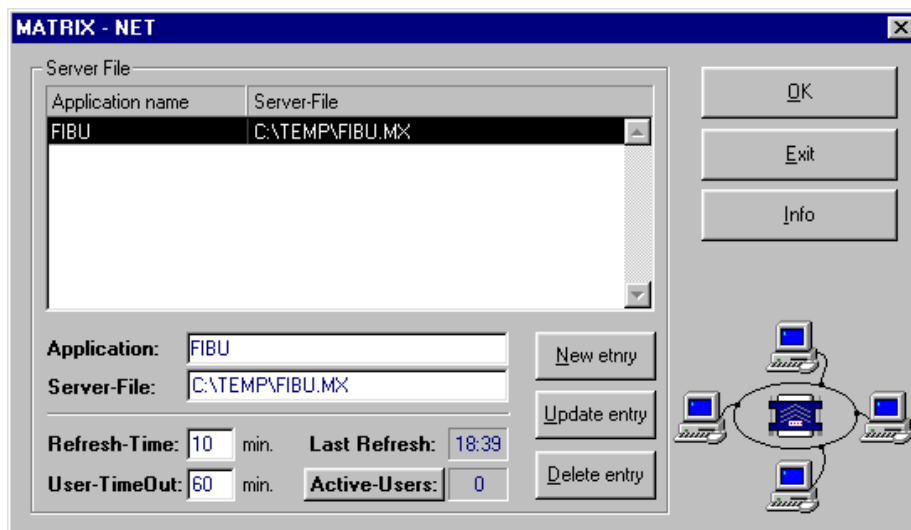
Example for the protection of an application in the network with MxNet.

We shall now describe by way of an example how a program can be protected quickly in a network.

Connect the dongle to a (any) computer in the network and install the software delivered with the unit. Start the Matrix-NET program on the computer with the dongle (MxNet32.exe). The program allows you to generate a server file for each program to be protected. However, it is also possible to manage several applications via the one server file.

In our example, the application to be protected is an accounting program which is to be used by a maximum of five users simultaneously.

Enter a name for the program to be protected in the entry field "Application" of the Matrix-NET program. In this example, the name "FIBU" is used. The name can be anything you like and it need not match the actual application name. The network licenses will be managed via the server file C:\TEMP\FIBU.MX in this example.

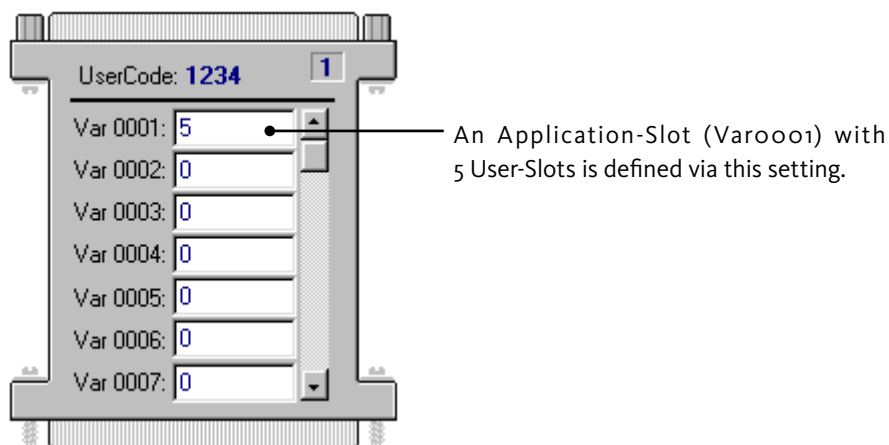


The name of the server file must always be entered with the absolute path and correspond to the naming conventions of the operating system.

The interval at which automatic refresh of the server file is carried out will be set to 10 minutes in this example.

The interval for the User-TimeOut, i.e. the time after which the user is to be released automatically, is set to 60 minutes here in case the user does not answer (crashing).

Now program the dongle with the appropriate number of simultaneously permissible program users; thus, in our example, “5” is entered and stored in the first dongle variable.



Now all that is left to do is to integrate the activation, the LogIn and the LogOut to the network into the program to be protected via the API functions included (see “API-Functions Reference”). With the call-up parameters of the functions, please be sure to use the correct name of the server file and the correct Application-Slot, i.e. that used for this application.

Example for calling up functions

In order to make the example simpler, we shall assume that only one dongle is connected.

The variable DNG_PORT is ignored (can have any value) for network calls.

```
/*-- Start the Matrix-API --*/
Init_MatrixAPI();

DNG_PORT = 1;  // this variable is ignored with network calls
DNG_NR    = 1;
AppSlot   = 1;

/*-- Activate the Network access in the Matrix-API --*/
SetConfig_MatrixNet(1, »F:\\TEMP\\FIBU.MX«);

/*-- LogIn: The program acquires a User-Slot --*/
ret = LogIn_MatrixNet(UserCode, AppSlot, DNG_NR);
if(ret < 0)
{
    printf(»All 5 Users are active! No more User-Slots free.«);
    exit;
}
.....
.....

/*-- LogOut: The User-Slot will be released --*/
Logout_MatrixNet(UserCode, AppSlot, DNG_NR);

/*-- Release the Matrix-API --*/
Release_MatrixAPI();
exit;
```


What you must take into account in your application

- Your protected application must always occupy a user slot during start-up. If no user slot is free, the application should be terminated with an appropriate message.
- Your application must refresh the User-Slot from time to time so that the TimeOut limit is not reached.
- Do not select an excessively small time interval for “Refresh-Time” and “User-TimeOut” so that the system is subjected to the least possible strain.
- In the case of abnormal termination (crashing) of your application, the user slot remains occupied.

This can then be removed manually or automatically by the MxNet server program when the TimeOut limit is reached (see “Settings of the MxNET server program”).

It is not necessary to delete the User-Slot before the abnormally terminated application is restarted.

This finds the existing User-Slot and refreshes it.

- When terminating your application on a client, you should always release the User-Slot.
- It is not necessary to install the LPT/USB drivers on the network client when you disable the LPT/USB support. Copy the file matrix.ini to the \win-dows\system directory of each network client and set the entries AccessLPT and AccessUSB to OFF.

Advantages and disadvantages of MxNET license management

Advantages

- The MxNet method provides reasonably priced management of the network licenses for several applications with just one dongle.
- As MxNet works independently of network protocols, it can easily be used in any desired network system.
- All functions of the Matrix-API with the exception of those for writing data to the dongle (Dongle_WriteData and Dongle_WriteDataEx) can be used for single-work-place applications and for network management.
- Only two additional API functions are used for the management of the network licenses (User-Slots). This makes it very easy to integrate the protection into your application.

Disadvantages

- The MxNet method is an asynchronous process. This means that the polling of the dongle information and the polling of the server file take place in a time displaced way. Thus, data cannot be stored in the dongle via MxNet.

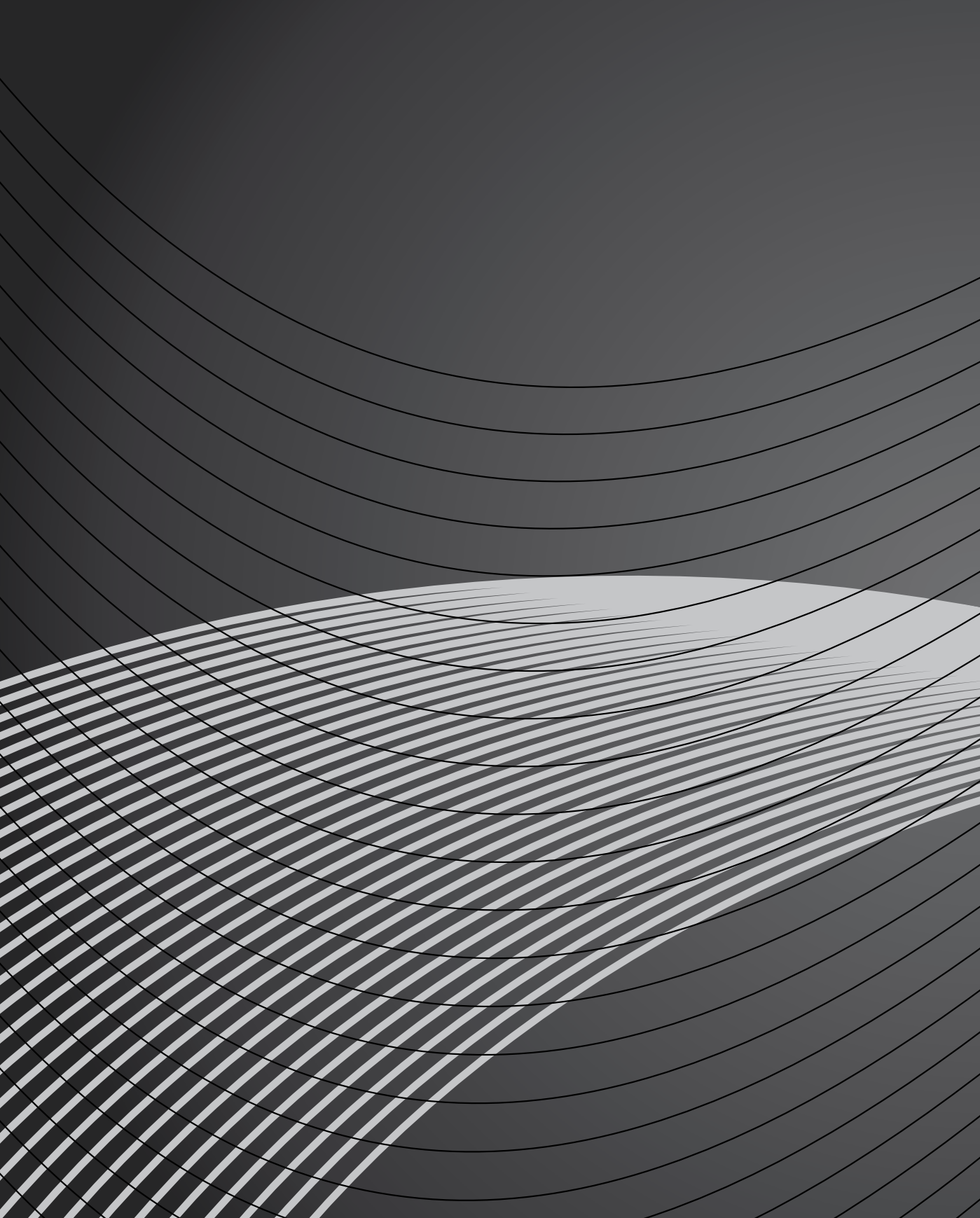
Advantages and disadvantages of license management with one dongle per workplace

Advantages

- This method is more effective if you would like to approve the use of your application as an in-house solution in the local LAN network only, so that external workplaces (remote) cannot use the application without the necessary dongle.
- Depending on the requirements, the dongles can contain differing information.

Disadvantages

- This method is more costly with a large number of workplaces.



4

Delivery of protected
programs

Delivery of protected programs

Applications for Windows operating systems

At run-time, the software protected via Matrix requires the following files, which you need to include with the software you deliver:

Matrix-API

- matrix16.dll – DLL for 16-bit applications
- matrix32.dll – DLL for 32-bit applications
- matrix64.dll – DLL for 64-bit applications
- matrix.ini – Configuration file to change settings for your customer if necessary

LPT-Drivers

- iwport.vxd – LPT driver for Windows 95/98/ME
- iwport.sys – LPT driver for Windows NT/2000/XP/Vista
- drv_inst.exe – Installation program for the Windows NT/2000/XP/Vista LPT driver

USB-Driver (required only in “Driver-Mode”)

- iwusb.sys – USB driver for Windows 98/ME/2000/XP/Vista
- iwusb_x64.sys – USB driver for Windows XP-64
- iwusb.inf – USB driver information file
- inf_inst.exe – Installation program for the Windows 98/ME/2000/XP/Vista USB driver

Network-Program (required only for network license management)

- mxnet32.exe – MxNET server program

Support-Tool

- mxcheck.exe – Tool to check the drivers and components installed on the system

16-bit applications

16-bit applications under Windows 3.x only require matrix16.dll and no driver.
16-bit applications under Windows 95/98, ME, NT, 2000 and XP/Vista require the files matrix16.dll, matrix32.dll and the corresponding driver.

32-bit applications

32-bit applications only require the file matrix32.dll and the corresponding driver.

64-bit applications

64-bit applications only require the file matrix64.dll and the corresponding driver.

What you have to include in the Delivery to your Customers

API

It is necessary to copy the DLL files into the directory `\windows\system` for your customer.

If you have a 32-bit application, it is enough to deliver the `matrix32.dll` file to your customer. If you have integrated the Matrix-API statically or have only protected your application via Matrix-Crypt, the DLL is not necessary as the API is already integrated into your application.

LPT-Driver

It is necessary for you to copy the LPT VXD driver `iwport.vxd` for Windows 95/98/ME into the directory `\windows\system` for your customer.

It is necessary to install the LPT SYS driver `iwport.sys` for Windows NT/2000/XP/Vista. This driver can be installed automatically via the Matrix-API or manually via the `drv_inst.exe` program and the corresponding driver file `iwport.sy_`.

If you prefer automatic installation, it is enough to copy the driver `iwport.sys` into the `\windows\system32\drivers` directory. The driver is then installed automatically when you start your protected application.

If you prefer manual installation, please deliver the installation program `drv_inst.exe` and the corresponding driver file `iwport.sy_` to your customer. Copy these files into a directory of your choice (for example into a subdirectory of your application) via your setup routine. You can then call up the installation program `drv_inst.exe` from your setup routine.

For more details, please read the section “LPT driver installation under Windows NT/2000/XP/Vista” in the “Installation” chapter and the Readme file from the `\nt_drv` directory.

Note: Please bear in mind that only users with the necessary access rights, such as the administrator, can install drivers under Windows NT/2000/XP/Vista in both cases.

USB-Driver

If you deliver the dongles configured for “HID-Mode”, there is no need to install any driver on the user’s computer.

If you deliver the dongles configured for the “Driver-Mode”, you should consider the following:

You should deliver the USB driver (iwusb.sys, iwusb_x64, iwusb.inf) for Windows 98/ME/2000/XP/Vista and the corresponding installation program inf_inst.exe to your customer and copy it into a directory of your choice via your setup routine (for example into a subdirectory of your application).

You can call up the installation program inf_inst.exe from your setup routine.

For details please read the section “USB driver installation under Windows 98/ME/2000/XP/Vista” in the “Installation” chapter and the Readme file from the \drv_usb directory.

Network application

The MxNET server program is only required if you handle license management for your network application via MxNet. This application should be copied to the \windows\system directory.

The MxNet server program can also be registered as a Windows service, so that the program is started automatically during the boot-up of Windows. The advantage of a service over an Autostart entry is that the service is also started if there is no User-Login in Windows.

You can directly register MxNet as a service from your setup routine by starting MxNet with the corresponding parameters. The following call-up parameters are available:

mxnet32.exe	-i	(Install MxNET service)
mxnet32.exe	-r	(Uninstall MxNET service)

Configuration and Tools

Optionally, you can deliver the matrix.ini file instead. With corresponding entries in this file, you have the ability to switch off the LPT or USB support if necessary (AccessLPT=ON/OFF, AccessUSB=ON/OFF). For instance, if LPT support is switched off, it is no longer necessary to install the LPT drivers. This is, for example, practical if only USB dongles can be used on computers or notebooks without an LPT interface.

This file should be copied into the \windows\system directory. If this file is not present, the Matrix-API uses default settings (all ON).

The mxcheck.exe program for checking the APIs and drivers installed in the system can be supplied to your customers as a support tool.

Applications for Linux and Mac OS X operating systems

For Linux and Mac OS X applications, please read the corresponding Readme_redist file.

Note: You will find the newest versions of the API and tools available for download at www.tdi-matrix.com

Remote Update

General description

Matrix Remote Update allows you to update the memory contents of your customer's dongles without the customer having to return the Matrix-dongle to you. This is useful if, for example, additional modules within your protected software need to be activated.

Matrix Remote Update consists of a tool named "MxGenDat" which allows you to create an encrypted EXE file, named MxModUpd.exe. This small update program is then sent to the customer (for example via Email) who then runs it. The customer need not be online to do this.

The update program communicates with the Matrix-dongle(s) attached to the customer's computer and carries out the specified checks and changes to the dongle's memory. It is possible to vary the level of authentication used by the update program to meet your software's and customer base needs.

The update program is able to change also MK/MKU dongles, which are alterable otherwise only with a MasterKey dongle.

MxGenDat is an efficient alternative to online updates, which are more complicated to implement and are often not accepted by customers.

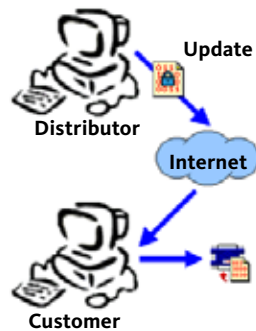
Benefits of Matrix Remote Update

- Additional modules within your protected software can be activated
- The customer's User Rights can be modified
- The License Conditions can be changed (for example to pay per use)
- The customer can add authentication of other software products to the same dongle
- Saves time and money, no shipping of Matrix-dongles back and forth
- The update program can be custom configured for your unique Matrix-dongle
- Your customer can use the new features of your software, without any time delay
- Substantially reduce your support activity
- Allows the update of MK/MKU dongles

How does Matrix Remote Update work?

Matrix Remote Update is very simple to use. Only two steps are necessary to the update Matrix-dongle:

1. The distributor create the customer specific update program and sends it to the customer.
2. The customer receives the update program and runs it.



Basic settings

The following basic settings are available in MxGenDat:

MATRIX Remote Update - Create encrypted File for Modul-Update (DEMO)

Update operations | Messages | DEC | HEX

General parameters for the dongle identification

Model: ML/MLU | 60 | Bytes | UserCode: 1234 (DEMO)

Serial-Nr: 0 | No Check

☐ Use for encryption the same 128-Bit Key as the one stored in Dongle

Key: 0 | 0 | 0 | 0

NOTE: Using this Key, an additional encryption will proceed. Make sure that this Key correspond to the Key of the Dongle to be updated.

Current Value	Update operation	New Value	<>
Var 0001: 2237	Check Only	0	^
Var 0002: 4389	Check & Change	4390	⌵
Var 0003: 0	Change always	17	
Var 0004: 0	Not used	0	
Var 0005: 0	Not used	0	
Var 0006: 0	Not used	0	
Var 0007: 0	Not used	0	⌵

Exit

Load settings

Save settings

Info

Help

▼ Create encrypted Update-File ▼

0110 0011 → 0110 0011

“Update operations” Tab

Model

Enter the model of the Matrix-dongle that your customer has.

Serial No. (optional)

Enter the serial number of the Matrix-dongle for which the Update program should be created. If no serial number is entered (No check) the update program will search for and update only the first Matrix-dongle it finds with the correct UserCode.

Activating the serial number check will increase the level of authentication by linking checks/changes to a particular Matrix-dongle.

Note: Be careful to enter the correct serial number!

Use for encryption the same 128-Bit Key as is stored in the Matrix-dongle (optional)

By activating this option you can further increase the level of authentication. The Update EXE will also be encrypted with this 128-Bit key. Decryption will take place via the customer's Matrix-Dongle. This will only be possible when the key inside the update program is the same as that in the Matrix-dongle.

Assuming each customer has a different key, use of this option will prevent customers using updates intended for others.

Note: Be careful to enter the correct 128-Bit Key!

Variable list

Current Value:	The value currently in the Matrix-dongle
Update Operation:	Not Used / Check Only / Check & Change / Change always
New Value:	The value to which the Matrix-dongle is to be set

This set of fields determines the actual changes to be made to the memory variables stored in the Matrix-dongle:

< *Change always* >

Is the simplest update form which will store the “New Value” in the variable of the Matrix-dongle, regardless of it’s previous content.

< *Check @ Change* >

Is the more complex variant which will first check that the “Current Value” equals that stored in the variable of the Matrix-dongle. If it does, it will be changed to the “New Value”.

< *Check Only* >

Will just check the variable against the “Current Value” without changing it.

< *Not Used* >

Will do nothing.

“Messages” Tab

The various messages displayed by the program can be modified here to suit your particular requirements such as a different language.

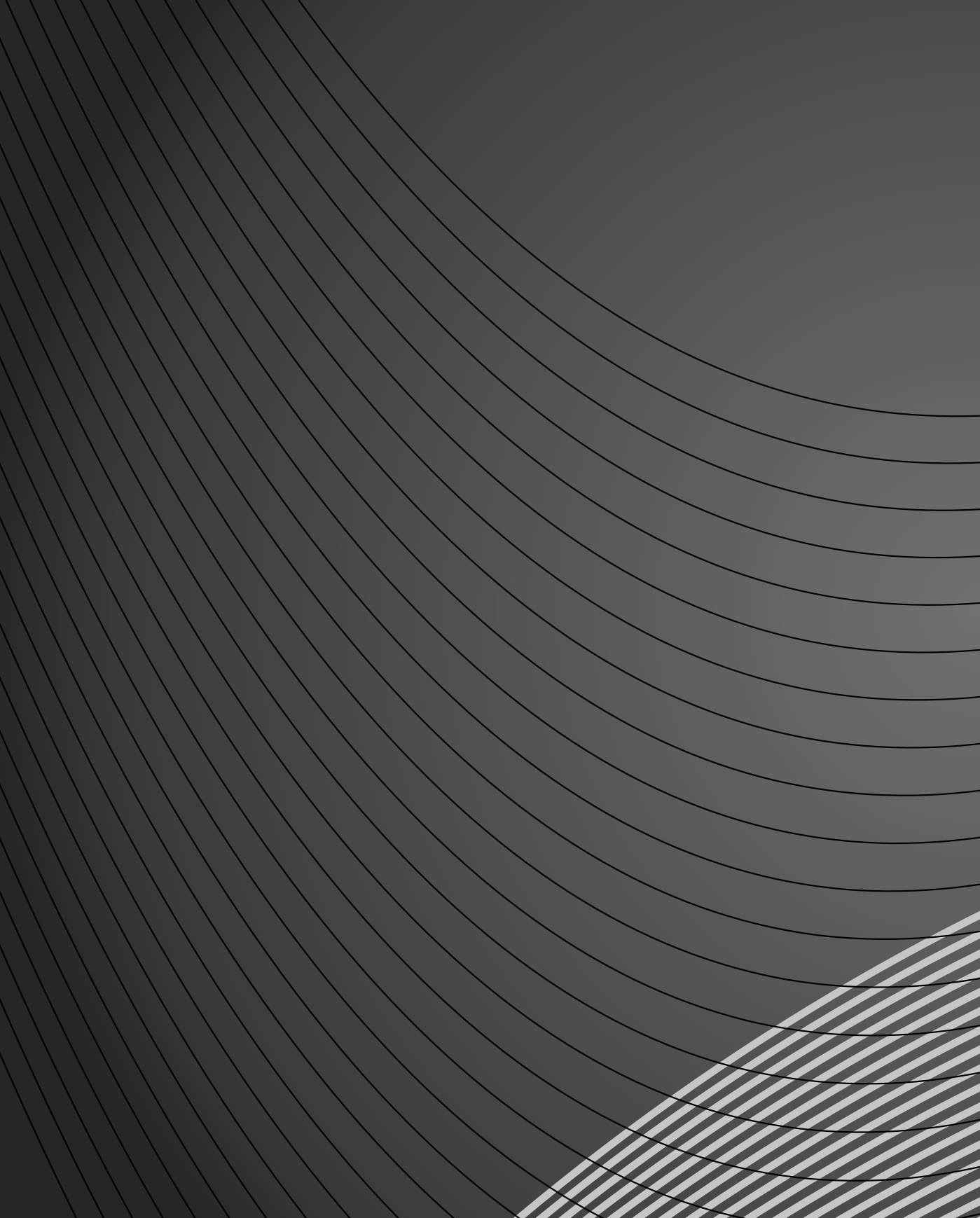
Save / Load settings

To facilitate the management of different update files, you can save your settings as a project. Saved projects can be reloaded for further use.

v Create encrypted Update-File v

Generate the update program MxModUpd.exe, which you send to your customer.

Note: If you’d like to generate an update program for MK/MKU dongles, the MasterKey dongle must be plugged in at the time you generate the update program. Consequently, the update program will be able to update the MK/MKU dongles without using a MasterKey dongle.



5

Guidlines for good
software protection

Guidelines for good software protection

In order to effectively protect a program from bootleg copies, it is always necessary to know the limits of the protective measures involved. In the case of a qualified software protection such as Matrix, hardware security is so thorough that there are not many opportunities for crackers to manipulate it. The weak point is thus the integration of the protection into your programs. A cracker will start here and attempt to separate the intended unity of protective hardware and the software to be protected. The goal must thus be to integrate Matrix into your programs in an inseparable or virtually inseparable way.

Of course, there can be no patent solution for an absolutely safe protection. Ultimately, though, imagination and creativity can help you to reliably protect your investment and make things difficult for bootleggers.

Please note: Rapid integration into your program can never be as reliable as a solution which is carefully thought out but perhaps takes longer to implement.

In order to achieve good protection for your application, you should take the following points into account and include them when you develop your protection.

Is there such a thing as an “uncrackable” program?

This question must be answered with “no”, as any type of protection can be detected and eliminated. Any company which offers protective products and promises to guarantee the perfect protection for your software can be considered as not very reliable. However, the measures put at your disposal by Matrix allow you to increase the work required to analyse and eliminate the protection as much as you desire.

Tools of the cracker

A cracker will usually use a debug program, in the simplest case for example DEBUG.EXE which is supplied with the operating system. However, you should always assume that he will use more complicated tools with which your program can be executed step by step. Via analysis of the assembler code, he will attempt to find the Matrix-dongle calls and change them in such a way that the program being cracked also functions if no dongle is connected.

If you have, for example, distributed the function calls mentioned above in your program, the work for the cracker will of course be more difficult than if only one function call is made when the program is started.

Dongle query when the program is started

A simple Matrix query should always be carried out when the program is started. This ensures that the Matrix-dongle is present. This first step is only to ensure the dongle is correctly recognised. However, this step is not relevant to the actual protection of the software because it is relatively easy to circumvent. In the further course of the application though, it can be assumed that the Matrix-dongle is present.

Frequent queries

Protection which is only active when starting the program is thus relatively easy to detect and circumvent. Also, the user can easily move the dongle to another computer and start a second (third, fourth...) copy of your program. Any intentional prevention of multiple use would be doomed to failure from the start. For this reason, you should carry out a Matrix-dongle query as often as possible in your program.

Protective measures during the running of the program

If possible, the dongle queries should also be included in the lowest levels of the program. Widespread distribution of the protective measures is considerably superior to any easily detectable individual measure.

Distributing the queries in the code

Good protection is achieved simply by distributing Matrix calls through the entire program. Program sequences for calling the dongle should not always be programmed in one piece, as otherwise (almost) identical code would be created. This would make the dongle queries easier to localise for the attacker. You can improve the protection by distributing the procedures with dongle queries throughout the whole of the source code (if possible).

Use of the Matrix memory

You should also make use of the memory available in the Matrix-dongle in the queries. You can for example store customer codes, serial numbers or program variables in the dongle's memory and evaluate these while the program is running.

Using these simple techniques, you can protect your program in such a way that an attempt at cracking the program is practically hopeless without the correct Matrix-dongle as data, parameters or parts of codes which are necessary for running the program are only to be found in the dongle. As the cracker cannot guess this information, he must have the correct dongle for his attempt at cracking the program.

Store a few random numbers in the unused memory of the dongle and read them out and check their presence or value later in the program.

Working with the cryptography

Make use of the options which the encrypt and decrypt function of the Matrix-dongles provide in order to encrypt/decrypt your program data. Using a 128-bit XTEA key which you define yourself and which cannot be read from the dongle, you can decrypt important fixed program data while the program is running and convert them back into meaningful values. Encryption can for example be carried out via the Matrix programming software and these encrypted values, such as constants, calculation variables etc., can be integrated into your program. In order to ensure that your program continues to function effectively, the Matrix-dongle must always be connected to the computer while the program is running, as the values can only be decrypted via the Matrix-dongle. A pirate copy without a dongle would thus be useless.

Use the Matrix encryption to authenticate users “on-the-fly”

Implement flexible authentication and e-commerce solutions using the Matrix hardware as a “Crypto Engine”. You can create random encrypted sequences at run-time to verify user passwords “on-the-fly” without storing this passwords or any real user authentication codes in the memory of the Matrix-dongle.

A hacker cannot simulate or replace this authentication sequence, because there are continuously changing running conditions.

Programming technique

Do not waste time concealing the queries in your programming language via “untidy” programming, for many compilers optimise the code in such a way that “tidy” assembler code is generated in this case.

Do not use secret switches in the program which deactivate the queries for the purposes of internal testing. These would allow the entire security system to be switched off by making one simple change to the program code.

Please ensure that the program version you deliver is prepared without debug information.

Do not use simple integer comparisons

After reading a variable from the memory of the Matrix-dongle, use complex expressions instead of simple integer comparison to check it's value.

For example, you want to check if the variable has the value 225. Instead of using a simple comparison such as *if(MyVar == 225)....*, apply a more complex statement:

*if((MyVar+90)/7 == Sqrt(MyVar)*3)....*

Using a number of complex floating-point operations will create very complicated machine code structures. A hacker will have much more trouble understanding this code and what's going on.

Measures to take after detecting an attack

If you have detected an attack on, or change in, your program via one of the methods described above or via consistency checks, there are a number of methods for making the program unusable. The measures taken here should be “decoupled” from the detection of the attack as far as possible to hamper further progress in cracking your program. The measures described in the following sections can be used for this purpose.

Belated reaction

Postponement of the countermeasures. After a certain time (ranging from seconds to days if the system date is utilised), the program can no longer be started.

Concealment

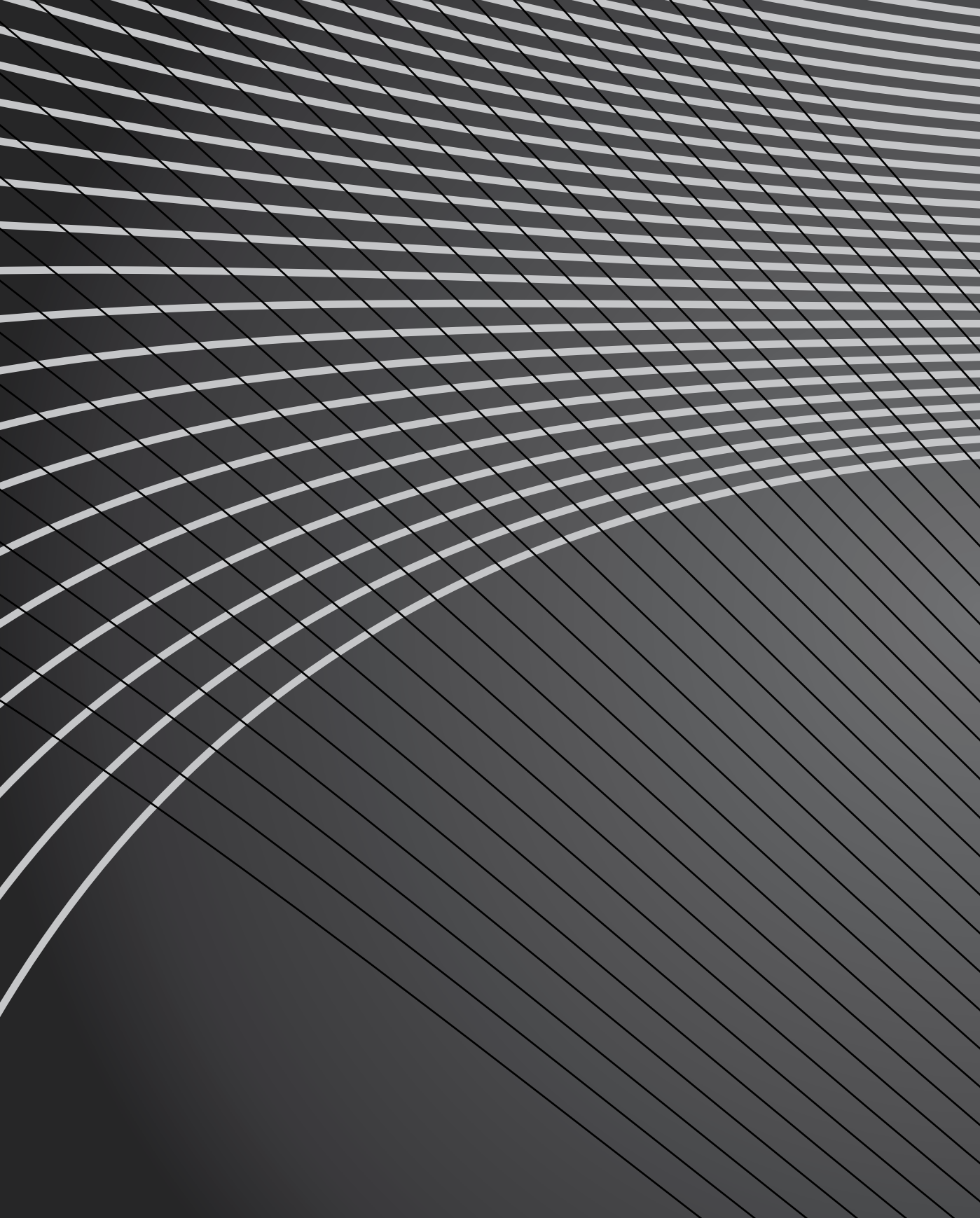
Concealing the relationship between cause and effect. One small change causes many other values to change, of which in turn only one initiates the actual measure.

Distortion of results

The program should not be exited immediately after detection of an attack, but should work on and provide absurd results if necessary.

Restrictions of the program functions

Countermeasures can also be limited to the switching off of individual program functions such as Print, Save etc. The cracker doesn't notice until later that his efforts were in vain.



6

API-Functions Reference

API-Functions Reference

Overview of API-Functions

Function	Description	available for			Page
		LPT	USB	NET	
Init_MatrixAPI	Starts the Matrix API	✓	✓	✓	94
Release_MatrixAPI	Closes the Matrix API	✓	✓	✓	94
GetVersionAPI	Returns the version number of the Matrix-API	✓	✓	✓	95
GetVersionDRV	Returns the version number of the LPT driver	✓			95
GetVersionDRV_USB	Returns the version number of the USB driver		✓		96
SetW95Access	Communication with the dongle under Wingx with or without VXD driver	✓			97
GetPortAdr	Returns the address of the LPT port	✓			97
PausePrinterActivity	Stops the Windows print-spooler (Win 9.x/ME/NT/2000/XP/Vista only)	✓			98
ResumePrinterActivity	Releases the Windows print-spooler (Win 9.x/ME/NT/2000/XP/Vista only)	✓			98
Dongle_Find	Searches the dongle and returns the LPT port at which it was found	✓	✓	✓	99
Dongle_FindEx	Searches all LPT ports and dongles and stores this information in a data buffer	✓		✓	100
Dongle_Count	Returns the number of dongles available at the specified LPT or USB interface	✓	✓	✓	101
Dongle_MemSize	Returns the memory size of the dongle in Bytes	✓	✓	✓	103
Dongle_Model	Reads the model number of the hardware from the dongle	✓	✓	✓	104
Dongle_Version	Reads the version number of the dongle	✓	✓	✓	106
Dongle_ReadData	Reads the data from the dongle beginning from the 1st memory field	✓	✓	✓	108

Function	Description	available for			Page
		LPT	USB	NET	
Dongle_ReadDataEx	Reads the data from the Matrix-dongle beginning from the specified memory field	✓	✓	✓	110
Dongle_WriteData	Writes data into the Matrix-dongle beginning from the first memory field	✓	✓		112
Dongle_WriteDataEx	Writes data into the Matrix-dongle beginning from the specified memory field	✓	✓		114
Dongle_ReadSerNr	Reads the unique serial number which is assigned to each Matrix-dongle	✓	✓	✓	116
Dongle_WriteKey	Saves the 128-bit XTEA key in the dongle	✓	✓		118
Dongle_GetKeyFlag	Checks whether a 128-bit XTEA key different to zero is available in the dongle	✓	✓		120
Dongle_EncryptData	Encrypts a data block with a size of 8 bytes through the Matrix-dongle	✓	✓		122
Dongle_DecryptData	Sets the dongle to the desired operating USB mode "HID-Mode" or "Driver-Mode"	✓	✓		124
Dongle_SetDriverFlag	Sets the dongle to the desired operating USB mode "HID-Mode" or "Driver-Mode"		✓		126
Dongle_GetDriverFlag	Reads the current USB operating mode of the dongle "HID-Mode" or "Driver-Mode"		✓		128
SetConfig_MatrixNet	Activates or deactivates network access			✓	129
GetConfig_MatrixNet	Returns from Server-File parameters configured with the MxNET server program			✓	130
LogIn_MatrixNet	Logs on the network client and acquires/ refreshes the user slot in the server file			✓	131
LogOut_MatrixNet	Logs off the network client and releases the UserSlot in the server file again			✓	132

Description of API-Functions

These functions and the corresponding calling parameters are described in the following.

Init_MatrixAPI - Starts the Matrix API.

Syntax

```
short Init_MatrixAPI( );
```

You have to initialize the Matrix-API before using the dongle functions.

Returns

- 0 The API was started successfully.
- 1 The 32-Bit API (DLL) was not found. This return value can only occur in the 16-Bit API.
- 10 The API is currently occupied by another application. This return value can only occur if the option “Shared access mode to the parallel interface” is activated.
- 20 An error occurred during access to the LPT driver.
- 21 An old LPT driver version was found and the API cannot use this LPT driver.
Install the new LPT driver.
- 22 An old USB driver version was found and the API cannot use this USB driver.
Install the new USB driver.

Note: If use only USB dongles and don't install the LPT drivers, then you can ignore the error codes reported from LPT.

Release_MatrixAPI - Close the Matrix API.

Syntax

```
short Release_MatrixAPI( );
```

You have to close the Matrix-API after using the dongle functions, to free all allocated resources.

Returns

This function has no return value.

GetVersionAPI - Returns the version number of the Matrix-API.

Syntax

```
long GetVersionAPI( );
```

Supports

LPT, USB, NET

Returns

The version number of the Matrix API is returned. The two higher bytes are the higher version number (HIWORD = MajorNumber). The two lower bytes are the lower version number (LOWORD = MinorNumber)

GetVersionDRV - Returns the version number of the LPT driver.

Syntax

```
long GetVersionDRV( );
```

Supports

LPT

Returns

The version number of the LPT driver is returned. The two higher bytes are the higher version number (HIWORD = MajorNumber). The two lower bytes are the lower version number (LOWORD = MinorNumber)

GetVersionDRV_USB - Returns the version number of the USB driver.

Syntax

```
long GetVersionDRV_USB( );
```

Supports

USB

Returns

The version number of the USB driver is returned. The two higher bytes are the higher version number (HIWORD = MajorNumber). The two lower bytes are the lower version number (LOWORD = MinorNumber)

Example for the determination of the version number

The version number of the Matrix-API is determined in this example:

C/C++

```
long API_VerNr;

API_VerNr = GetVersionAPI();
printf(>API-Version: %d.%d<, HIWORD(API_VerNr), LOWORD(API_
VerNr));
```

VisualBasic

```
Dim API_VerNr As Long
Dim VerMajor As Integer
Dim VerMinor As Integer
Const Shift16 = 2 ^ 16

API_VerNr = GetVersionAPI()
VerMinor = CInt(API_VerNr And 65535)
VerMajor = CInt(API_VerNr \ Shift16)
MsgBox >API-Versions: > & VerMajor & >.< VerMinor
```

Wing5/98 with or without VXD driver.

Syntax

```
void SetW95Access( short Mode );
```

Supports

LPT

Parameter

Mode Can have the value 1 or 2 or the predefined values:
IW_DRIVER / IW_NODRIVER

- 1 Communication takes place via the VXD driver.
- 2 Communication takes place without the VXD driver.

Returns

This function has no return value.

GetPortAdr - Returns the address of the LPT port.

Syntax

```
short GetPortAdr( short LptNr );
```

Supports

LPT

Parameter

LptNr This is the number of the LPTport, for example 1 for LPT1, 2 for LPT2 or 3 for LPT3. The LPT number must be between 1 and 3.

Return

The address of the LPT port is returned, for example 378 (Hex) for LPT1 or 0 if the specified LPT port is not available. This function allows the number of available LPT ports in the PC in question to be determined.

Example

The number of available LPT ports is determined in this example:

```
for(LptNr=1; LptNr<=3; LptNr++)  
{  
    if(GetPortAdr(LptNr) > 0) MaxLPT = LptNr;  
}
```

PausePrinterActivity - Stops the Windows print-spooler.
(under Wing.x / Win-NT / 2000 only)

Syntax

```
short PausePrinterActivity( );
```

Supports

LPT

Returns

- o The Print-Spooler was stopped.
- 14 It was not possible to stop the print spooler.

Note: With this function the Windows print-spooler can be stopped. During printjobs the LPT-port can be released for dongle communication. After dongle communication has finished the print spooler should be released with the function ResumePrinterActivity to continue printing.

ResumePrinterActivity - Releases the Windows print-spooler again.
(under Wing.x / Win-NT / 2000 only)

Syntax

```
short ResumePrinterActivity( );
```

Supports

LPT

Returns

- o The Print-Spooler was released.
- 14 It was not possible to release the print spooler.

Note: With this function the Windows print-spooler could be released again. Print jobs you have stopped could be continued.

Dongle_Find	- Searches for the dongle and returns the LPT/USB interface at which it was found.
--------------------	--

Syntax

```
short Dongle_Find( );
```

Supports

LPT, USB, NET

Returns

The LPT port at which the dongle was found is returned: **1** for LPT1, **2** for LPT 2 or **3** for LPT3 or **'U'** (ASCII 85) for USB.

- 0 No dongle was found at any of the LPT or USB interfaces.
- 1 No LPT-Ports available.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Note: This function only supplies the first port at which a dongle is found, regardless of whether other dongles are available at other LPT or USB interfaces.

Use the function `Dongle_FindEx` or `Dongle_Count` if you would like to check all LPT ports for the existence of one or several dongles.

As the function `Dongle_FindEx` does not support the USB port, the `Dongle_Count` function can be used for USB dongles.

Dongle_FindEx - Searches for all LPT ports and dongles and stores this information in a data buffer.

Syntax

```
short Dongle_FindEx( DNGINFO *xBuffer );
```

Supports

LPT, NET

Parameter

xBuffer This is a pointer to a data buffer in which the information is stored after reading. The data buffer is an array with a maximum size of 3 and has the following type description:

```
typedef struct {
    int LPT_Nr;    /* Number of LPT port */
    int LPT_Adr;   /* Address of LPT port */
    int DNG_Cnt;   /* Number of dongles at LPT port */
} DNGINFO;
```

Returns

The number of available LPT ports is returned, or:

- 0 No LPT-Ports available.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Note: As the function Dongle_FindEx does not support the USB port, the Dongle_Count function can be used for USB dongles.

Dongle_Count	- Returns the number of dongles available at the specified LPT or USB interface.
---------------------	--

Syntax

```
short Dongle_Count( short PortNr );
```

Supports

LPT, USB, NET

Parameter

PortNr With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. Example: 1 for LPT1, 2 for LPT2 or 3 for LPT3.
With USB, this parameter must include the value 'U' (ASCII 85).
This variable is ignored with network calls.

Returns

The number of dongles available at the specified LPT port is returned. The maximum number is 99 for LPT and 127 for USB.

- 0 No dongle was found at the specified interface.
- 1 A communication error occurred, or the specified interface is not available.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Important note: If several dongles are plugged in to one LPT port, these are counted backwards from the PC to the printer. For example, with three dongles plugged in, the 3rd would be immediately at the PC and the 1st at the printer.

Example

The following program code will demonstrate how to use the function `Dongle_Count` to detect on which port (LPT or USB) the Matrix-dongle is connected.

```

short DNG_Port;
short DNG_Nr;
long  DNG_SerNr;
short Gefunden;
short i;

Gefunden = 0;

for(i=0; i<=3; i++)
{
    if(i== 0)
        DNG_Port = 85;    // USB
    else
        DNG_Port = i;     // 1-3 = LPT1-LPT3

    /*--- Get the count of dongles on 'DNG_Port' ---*/
    DNG_Count = Dongle_Count(DNG_Port);
    if(DNG_Count <= 0) continue;

    /*--- Search the dongle with correct UserCode, if ---*/
    /*--- there are more than one dongles present. ---*/
    for(DNG_Nr = 1; DNG_Nr <= DNG_Count; DNG_Nr++)
    {
        DNG_SerNr = Dongle_ReadSerNr(UserCode, DNG_Nr, DNG_Port);

        /*--- Dongle mit falschem UserCode, weiter suchen ---*/
        if(DNG_SerNr == -2) continue;

        /*--- Dongle mit richtigem UserCode gefunden ---*/
        if(DNG_SerNr > 0)
        {
            Gefunden = 1;
            break;
        }
    }
    if(Gefunden == 1) break;
}

/*-----*/
/* We know now the dongle number 'DNG_Nr' and the port      */
/* 'DNG_Port' which we have to use in following dongle calls */
/*-----*/

```

Dongle_MemSize - Returns the memory size of the dongle in Bytes.

Syntax

```
short Dongle_MemSize( short DongleNr,  
                      short PortNr );
```

Supports

LPT, USB, NET

Parameter

- | | |
|----------|---|
| DongleNr | This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function.
A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection. |
| PortNr | With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3.
With USB, this parameter must include the value 'U' (ASCII 85).
This variable is ignored with network calls. |

Returns

The memory size in bytes is returned. For example, if the ML-60 is connected, the return value is **60**. With the ML-12, the return value is **12** etc.

- o The memory size could not be read.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Note: The maximum number of data fields in the dongle can also be determined via this return value.

Example

Determine memory size and the maximum number of data fields for the first Dongle at LPT2.

```
Memory = Dongle_MemSize(1, 2)
MaxValues = Memory / 4
```

In a Matrix-dongle with a memory size of 60 bytes, the number of data fields is 15.

Dongle_Model	- Reads the model number of the hardware from the dongle.
---------------------	---

Syntax

```
long Dongle_Model( short DongleNr,
                  short PortNr );
```

Supports

LPT, USB, NET

Parameter

DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85). This variable is ignored with network calls.

Returns

The model number of the Matrix hardware is returned, for example 257, 657 etc.

- o The model number could not be read.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Note: You should always give the version or model number when addressing questions to our hotline.

Dongle_Version - Reads the version number of the dongle.

Syntax

```
long Dongle_Version( short DongleNr,  
                    short PortNr );
```

Supports

LPT, USB, NET

Parameter

DongleNr	<p>This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function.</p> <p>A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.</p>
PortNr	<p>With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85).</p> <p>This variable is ignored with network calls.</p>

Returns

The version number of the Matrix-dongle is returned. The return value is of type 'LONG'

-> HIWORD = MajorNumber, LOWORD = MinorNumber.

- o The version number could not be read.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Note: You should always give the version or model number when addressing questions to our hotline.

Dongle_ReadData - Reads the data from the Matrix-dongle beginning from the first memory field.

Syntax

```
short Dongle_ReadData( long   UserCode,
                        long   *Data,
                        short  Count,
                        short  DongleNr,
                        short  PortNr );
```

Supports

LPT, USB, NET

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
Data	This is a pointer to a data buffer in which the values read are stored. This data buffer must have at least the number of elements specified in the 'Count' parameter.
Count	This is the number of values to be read from the dongle. The maximum number of values can be specified here (see Dongle_Mem-Size)
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85). This variable is ignored with network calls.

Returns

The return value is the number of data fields which were successfully read from the dongle.

- o No data read from the dongle.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Important note: The reading process is only carried out if the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle. Do not call Dongle_ReadData continuous (e.g. in a loop) with different wrong "UserCodes"! This will activate the "Anti-Hacker" protection. In this case the dongle is locked and it will not work anymore even if the function is called with the right "UserCode". The dongle can be unlocked in this case, only with a MasterKey-dongle. (See also Dongle_ReadDataEx).

Dongle_ReadDataEx - Reads the data from the Matrix-dongle
beginning from the specified memory field.

Syntax

```
short Dongle_ReadDataEx( long   UserCode,
                          long   *Data,
                          short   Fpos,
                          short   Count,
                          short   DongleNr,
                          short   PortNr );
```

Supports

LPT, USB, NET

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
Data	This is a pointer to a data buffer in which the values read are stored. This data buffer must have at least the number of elements specified in the 'Count' parameter.
Fpos	This is the field number where to start reading.
Count	This is the number of values to be read from the dongle. The maximum number of values can be specified here (see Dongle_MemSize).
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85). This variable is ignored with network calls.

Returns

The return value is the number of data fields which were successfully read from the dongle.

- o No data read from the dongle.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Important note: The reading process is only carried out if the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle. Do not call Dongle_ReadDataEx continuous (e.g. in a loop) with different wrong "UserCodes"! This will activate the "Anti-Hacker" protection. In this case the dongle is locked and it will not work anymore even if the function is called with the right "UserCode". The dongle can be unlocked in this case, only with a MasterKey-dongle.

Dongle_WriteData - Writes data into the Matrix-dongle beginning from the first memory field.

Syntax

```
short Dongle_WriteData( long   UserCode,
                        long   *Data,
                        short   Count,
                        short   DongleNr,
                        short   PortNr );
```

Supports

LPT, USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
Data	This is a pointer to a data buffer with the values to be stored in the dongle. This data buffer must have at least the number of elements specified in the 'Count' parameter.
Count	This is the number of values which you would like to store in the dongle. The maximum number of values can be specified here (see Dongle_MemSize).
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85).

Returns

The return value is the number of values which were successfully stored in the dongle.

- o No data was stored in the dongle.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 3 The necessary MasterKey-dongle was not found (only by the MK-Series).
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: The writing process is only carried out when the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle. Do not call Dongle_WriteData continuous (e.g. in a loop) with different wrong "UserCodes"! This will activate the "Anti-Hacker" protection. In this case the dongle is locked and it will not work anymore even if the function is called with the right "UserCode". The dongle can be unlocked in this case, only with an MasterKey-dongle. (See also Dongle_WriteDataEx).

Dongle_WriteDataEx - Writes data into the Matrix-dongle
beginning from the specified memory field.

Syntax

```
short Dongle_WriteDataEx( long   UserCode,
                          long   *Data,
                          short  Fpos,
                          short  Count,
                          short  DongleNr,
                          short  PortNr );
```

Supports

LPT, USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
Data	This is a pointer to a data buffer with the values to be stored in the dongle. This data buffer must have at least the number of elements specified in the 'Count' parameter.
Fpos	This is the field number where to start writing.
Count	This is the number of values which you would like to store in the dongle. The maximum number of values can be specified here (see Dongle_MemSize).
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85).

Returns

The return value is the number of values which were successfully stored in the dongle.

- o No data was stored in the dongle.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 3 The necessary MasterKey-dongle was not found (only by the MK-Series).
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: The writing process is only carried out when the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle. Do not call Dongle_WriteDataEx continuous (e.g. in a loop) with different wrong "UserCodes"! This will activate the "Anti-Hacker" protection. In this case the dongle is locked and it will not work anymore even if the function is called with the right "UserCode". The dongle can be unlocked in this case, only with an MasterKey-dongle.

Dongle_ReadSerNr - Reads the unique serial number
which is assigned to each Matrix-dongle.

Syntax

```
long Dongle_ReadSerNr( long  UserCode,  
                        short DongleNr,  
                        short PortNr );
```

Supports

LPT, USB, NET

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
DongleNr	<p>This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function.</p> <p>A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.</p>
PortNr	<p>With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3.</p> <p>With USB, this parameter must include the value 'U' (ASCII 85).</p> <p>This variable is ignored with network calls.</p>

Returns

The return value is the serial number which were successfully read from the dongle.

- o Serial number is not present in the dongle.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 7 The dongle does not support this function. (Serial-Number is only supported from hardware version 2.3 onwards. The models ML-12/MK-12 do not support Serial-Number at all).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Important note: The reading process is only carried out if the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle.

Dongle_WriteKey - Saves the 128-bit XTEA key in the dongle.**Syntax**

```
short Dongle_WriteKey( long   UserCode,  
                        long   *KeyData,  
                        short  DongleNr,  
                        short  PortNr );
```

Supports

LPT, USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
KeyData	This is a pointer to a data buffer with the values to be stored in the dongle.
DongleNr	<p>This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function.</p> <p>A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.</p>
PortNr	<p>With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3.</p> <p>With USB, this parameter must include the value 'U' (ASCII 85).</p>

Returns

If saving is successful, the function has to return a value higher than 0.

- 0 No data was stored in the dongle.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 3 The necessary MasterKey-dongle was not found (only by the MK-Series).
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 7 The dongle does not support this function. (Cryptography is only supported from hardware version 2.1 onwards. The models ML-12/MK-12 do not support cryptography at all).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: The writing process is only carried out when the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle.

Dongle_GetKeyFlag - Checks whether a 128-bit XTEA key
different to zero is available in the dongle.

Syntax

```
short Dongle_GetKeyFlag( long   UserCode,  
                        short  DongleNr,  
                        short  PortNr );
```

Supports

LPT, USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
DongleNr	<p>This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function.</p> <p>A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.</p>
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85).

Returns

The return value is 1 if a key is available in the dongle or 0 if the key is not present in the dongle (Zero-Key -> all key bytes = 0).

- 1 128-bit key is present in the dongle.
- 0 128-bit key is not present in the dongle (Zero-Key).
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 7 The dongle does not support this function. (Cryptography is only supported from hardware version 2.1 onwards. The models ML-12/MK-12 do not support cryptography at all).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: As the 128-bit key cannot be read from the dongle, this function allows you to check whether a 128-bit key is available in the dongle. A "Zero-Key", i.e. with all bytes=0 is not interpreted as a valid key.

The reading process is only carried out if the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle.

Dongle_EncryptData - Transmit a 8 bytes clear data block to the dongle.

This is returned in XTEA-encrypted form.

Syntax

```
short Dongle_EncryptData( long   UserCode,  
                           long   *DataBlock,  
                           short  DongleNr,  
                           short  PortNr );
```

Supports

LPT, USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
DataBlock	This is a pointer to a 8 bytes data buffer with the values to be encrypted.
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85).

Returns

If encryption is successful, the function has to return a value higher than 0.

- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 7 The dongle does not support this function. (Cryptography is only supported from hardware version 2.1 onwards. The models ML-12/MK-12 do not support cryptography at all).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: The encryption process is only carried out when the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle.

Dongle_DecryptData - Transmit a 8 bytes encrypted data block to the dongle. This is returned decrypted as clear data.

Syntax

```
short Dongle_DecryptData( long   UserCode,  
                           long   *DataBlock,  
                           short  DongleNr,  
                           short  PortNr );
```

Supports

LPT, USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
DataBlock	This is a pointer to a 8 bytes data buffer with the values to be decrypted.
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.
PortNr	With LPT dongles, this is the number of the LPT port to which the dongles are connected. With LPT dongles, the port number must be between 1 and 3. With USB, this parameter must include the value 'U' (ASCII 85).

Returns

If decryption is successful, the function has to return a value higher than 0.

- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The LPT port cannot be acquired because it is already in use by other devices.
- 6 A fault occurred when the LPT port was accessed.
- 7 The dongle does not support this function. (Cryptography is only supported from hardware version 2.1 onwards. The models ML-12/MK-12 do not support cryptography at all).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: The decryption process is only carried out when the "UserCode" you specify is the same as the "UserCode" from the Matrix-dongle.

Dongle_SetDriverFlag - Sets the dongle to the desired USB operating mode “HID-Mode” or “Driver-Mode”.

Syntax

```
short Dongle_SetDriverFlag( long   UserCode,  
                             short  Mode,  
                             short  DongleNr,  
                             short  PortNr );
```

Supports

USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
Mode	Can have the value 0 or 1: 0 - “Driver-Mode” - with a proprietary USB driver. 1 - “HID-Mode” - without any proprietary USB driver.
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 127 dongles to the USB connection.
PortNr	Is the port to which the dongles are connected. This parameter must include the value ‘U’ (ASCII 85).

Returns

If successful the return value is **1**.

- 1 the desired operating mode were successfully set.
- 0 The specified 'Mode' is not valid.
- 1 A communication error has occurred or the dongle specified via 'DngNr' was not found at the port specified via 'PortNr'.
- 2 Is returned if the 'UserCode' specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the "Anti-Hacker" lock.
- 5 The USB port cannot be acquired because it is already in use by other devices.
- 7 The dongle does not support this function. (Setting of USB operating mode is only supported from hardware version 5.0 onwards.).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

Important note: Once the USB operating mode has been changed, the dongle has to be disconnected/connected, to allow the operating system to acknowledge the new operating mode.

Dongle_GetDriverFlag - Reads the current USB operating mode of the dongle “HID-Mode” or “Driver-Mode”.

Syntax

```
short Dongle_GetDriverFlag( long   UserCode,
                             short  DongleNr,
                             short  PortNr );
```

Supports

USB

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 127 dongles to the USB connection.
PortNr	Is the port to which the dongles are connected. This parameter must include the value 'U' (ASCII 85).

Returns

A return value **1** or **0** is the requested USB operating mode of the dongle.

- 1 The USB operating mode of the dongle is “HID-Mode”.
- 0 The USB operating mode of the dongle is “Driver-Mode”.
- 1 A communication error has occurred or the dongle specified via ‘DngNr’ was not found at the port specified via ‘PortNr’.
- 2 Is returned if the ‘UserCode’ specified is not the same as the UserCode from the dongle.
- 4 Is returned when the dongle is locked by the “Anti-Hacker” lock.
- 5 The USB port cannot be acquired because it is already in use by other devices.
- 7 The dongle does not support this function. (Setting of USB operating mode is only supported from hardware version 5.0 onwards.).
- 25 The list of USB devices could not be created.
- 26 The USB device could not be opened.
- 27 The USB device is not valid.
- 28 The USB device could not be configured.
- 29 No USB support in this operating system (for example Win-NT).

SetConfig_MatrixNet - Activates or deactivates network access.**Syntax**

```
short SetConfig_MatrixNet( short nAccess,  
                           char *nFile );
```

Supports

NET

Parameter

- nAccess Can have the value 0 or 1:
 0 - The dongle is connected locally.
 1 - Activates MxNET network access.
- nFile The name of the server file for MxNet network access is specified in this variable. The server file must be given with the absolute path.
 (e.g. \\servername\TEMP\MXFILE.NET).
 This variable is ignored (can be NULL) if access for a locally connected dongle is set with nAccess = 0.

Returns

- 0 Is returned if the network access was deactivated (local access).
- 1 Is returned if the network access was activated (network access).

Note: This function must always be called up before the functions LogIn_MatrixNet and LogOut_MatrixNet. The setting for network access can be deactivated by again calling up the function with the value nAccess = 0. If network access is deactivated, the API accesses a locally connected dongle.

GetConfig_MatrixNet - Returns from Server-File parameters configured with the MxNet server program.

Syntax

```
long GetConfig_MatrixNet( short Category );
```

Supports

NET

Parameter

- Category This variable describe which MxNET parameter should be returned.
Possible values for Category are:
- 0 - The versions number of the Server-File should be returned.
The returned version number is of type Long and have the format:
HIWORD = MajorNumber
LOWORD = MinorNumber
 - 1 - The configured 'Refresh-Time' (in minutes) should be returned.
 - 2 - The number of minutes elapsed since last File-Refresh should be returned.
 - 3 - The configured 'User-TimeOut' (in minutes) should be returned.

Returns

A return value higher or equal 0 is the requested MxNET parameter.

- 1 A error occured during reading Category, or the network access was not activated by SetConfig_MatrixNet.
- 7 Requested Category is not valid.
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Note: All parameters are provided directly from the encrypted server file. The parameters will be actualized automatically from the MxNet server program each time the server file is refreshed.

LogIn_MatrixNet - Logs on the network client and acquires/
refreshes the user slot in the server file.

Syntax

```
short LogIn_MatrixNet( long   UserCode,
                       short AppSlot,
                       short DongleNr );
```

Supports

NET

Parameter

UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
AppSlot	This is the number of the variable in the dongle which includes the number of network licenses ("Application-Slot").
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function.

Returns

The return value is greater or equal 0 when the user was logged in successfully and represents the number of free user slots.

- 1 The dongle specified via 'DngNr', or the variable in the dongle ("Application-Slot") specified via 'AppSlot' was not found.
- 2 Is returned if the specified 'UserCode' is not the same as the UserCode from the dongle.
- 31 Is returned if no User-Slot is free.
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.
- 35 The MxNet server program is not active.

Important note: Your application must call up the function from time to time in order to refresh the UserSlot entry before the TimeOut limit is reached. (see "Settings of the MxNet server program").

LogOut_MatrixNet - Logs off the network client and releases the UserSlot in the server file again.

Syntax

```
short LogOut_MatrixNet( long   UserCode,
                        short AppSlot,
                        short DongleNr );
```

Supports

NET

Parameter

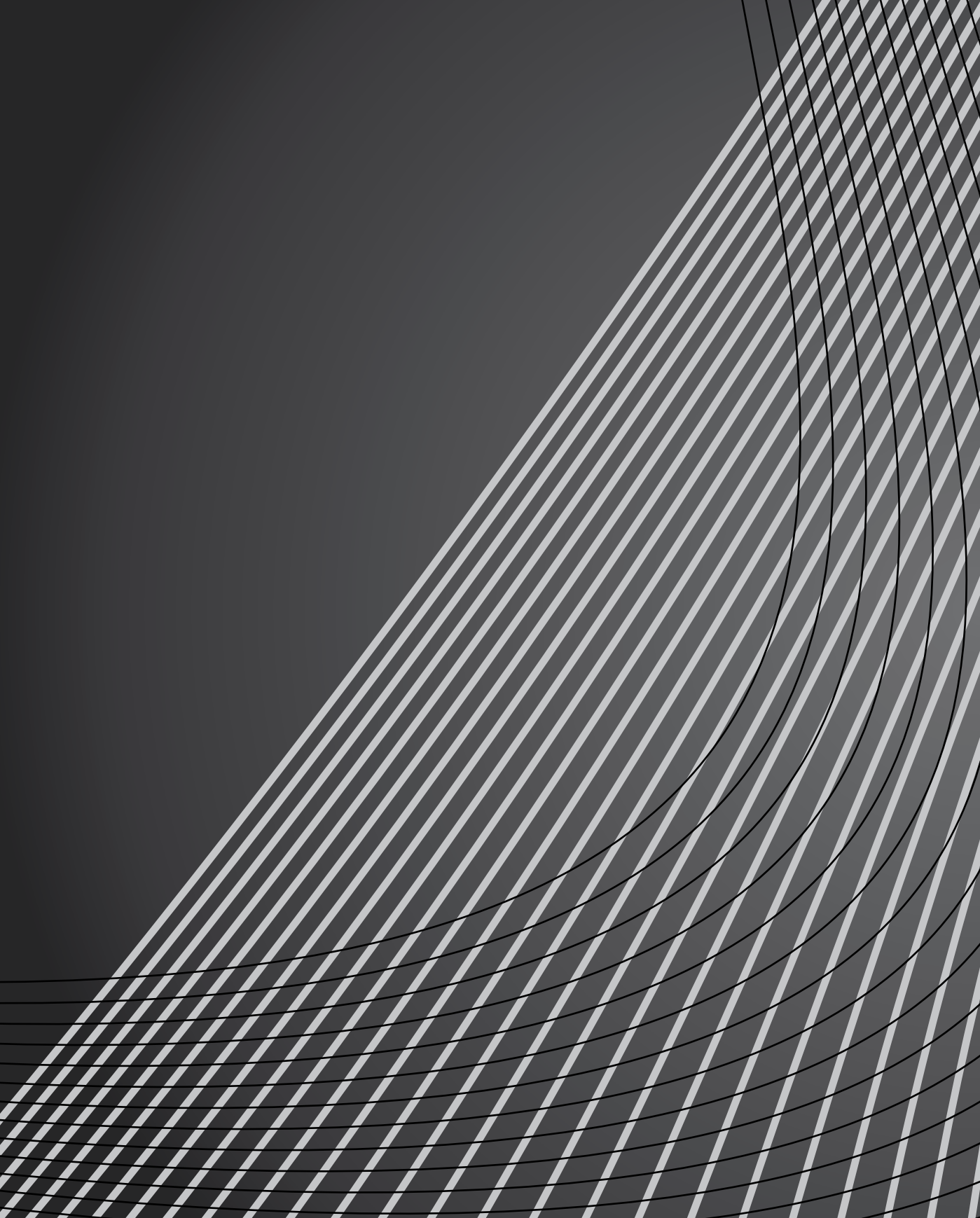
UserCode	This is your personal UserCode and must correspond with the internal UserCode from the dongle.
AppSlot	This is the number of the variable in the dongle which includes the number of network licenses ("Application-Slot").
DongleNr	This is the dongle number when several dongles are plugged in one after the other. The maximum number of dongles available can be determined via the Dongle_Count function. A maximum of 99 dongles can be connected to the LPT port and a maximum of 127 dongles to the USB connection.

Returns

The return value is greater or equal 0 when the user was logged out successfully and represents the number of free user slots.

- 1 The dongle specified via 'DngNr', or the variable in the dongle ("Application-Slot") specified via 'AppSlot' was not found.
- 2 Is returned if the specified 'UserCode' is not the same as the UserCode from the dongle.
- 31 Is returned if no User-Entry was found in the server file.
- 32 The server file was not found.
- 33 The server file cannot be accessed because it is being used by another application.
- 34 A fault occurred during access to the server file.

Important note: It is always necessary to carry out a LogOut before the application is terminated in order to release the UserSlot again.





7

Miscellaneous

Miscellaneous

Technical data

Connection	LPT/USB
Processor	RISC
	Also available separately for your own hardware development
Coding	Fixed, unchangeable customer code
Available memory	Depends on model: LPT: ML-12 → 12 Bytes = 3 variables (3 x 32-Bit) ML-60 → 60 Bytes = 15 variables (15 x 32-Bit) ML-316 → 316 Bytes = 79 variables (79 x 32-Bit) USB: MLU-60 → 60 Bytes = 15 variables (15 x 32-Bit) MLU-316 → 316 Bytes = 79 variables (79 x 32-Bit) Larger memory on request
Data integrity	> 40 years
Read/write cycles	1,000,000 cycles guaranteed
Stacking	Several modules can be stacked: LPT: A maximum of 99 modules with individual addressing USB: A maximum of 127 modules with individual addressing
Communication	Coded – Dongle ↔ PC
Operating systems	LPT: DOS, Windows 3.x/95/98/ME/NT/2000/XP/Vista, Linux, Mac OS X USB: Windows 98/ME/2000/XP/Vista, Linux, Mac OS X
Software	Programming software API and drivers available for Windows 3.x, 95, 98, ME, NT, 2000, XP/Vista, Linux and Mac OS X
Power supply	From the LPT interface - Supply voltage 2,5V to 5,5V From the USB interface - Supply voltage 3,3V to 5,5V
Supply current	Standby: < 950µA Read data: < 1,7 mA Write Data: < 2,6 mA
Dimensions	LPT: 43 x 54 x 14 mm (± 0,5 mm) USB: 55 x 16 x 8 mm (± 0,5 mm)
Miscellaneous	Special solutions on request

Prices and Delivery Terms

Model	Memory	Crypt / Decrypt	Graded price scale for bulk purchase (Euro)									
			1+	10+	100+	300+	500+	1K+	3K+	5K+	10K+	
ML-12	<div>LPT</div> 12 Bytes (3 data fields)		28,-	18,-	16,-	15,-	14,-	13,-	12,-	11,-	10, ⁵⁰	
ML-60	<div>LPT</div> 60 Bytes (15 data fields)		31,-	20,-	18,-	17,-	16,-	15,-	14,-	13,-	11, ⁵⁰	
ML-316	<div>LPT</div> 316 Bytes (79 data fields)		32,-	22,-	20,-	19,-	18,-	17,-	16,-	15,-	13, ⁵⁰	
MLU-60	<div>USB</div> 60 Bytes (15 data fields)		34,-	24,-	22,-	21,-	20,-	17,-	14,-	11,-	9,-	
MLU-316	<div>USB</div> 316 Bytes (79 data fields)		36,-	26,-	24,-	23,-	21,-	18,-	15,-	12,-	9, ⁵⁰	
MLU Piccolo USB*	12 Bytes (3 data fields)		Bulk - min. order size 10K									5, ⁸⁷

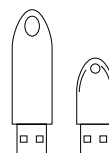
Matrix-dongles with larger memory available on request.

All prices are in EUR and exclude VAT and transport costs.

Discounts can be given by agreement for fixed number of units bought.

* MLU Piccolo – only available as bulk in min. quantities of 10K. Customized case color and Logo for free.

All USB dongles are available in two sizes: USB short and USB long. They have the same price and functionality.



All functions inclusive; network ability as well as all software, customer adjustments, advancements and support are contained in the price.

The prices of MK/MKU series are the basic prices of ML/MLU series + EUR 1,50 per unit. Data programming is only possible with the corresponding MasterKey for the MK/MKU series.

The MasterKey-dongle is included free of charge with the delivery of the first order to customers of the MK/MKU series.

Customers of the ML/MLU series can order additional an MasterKey.

General Terms of Trade

1. General

- a) All deliveries and services are based on the following terms of trade only. Divergent agreements must be made in writing. All present and future delivery transactions are based on these terms of trade.
- b) All information relating to the equipment we sell in the form of product descriptions, brochures etc. is always not binding if it is not expressly declared as binding. This is true in particular for changes with regard to technical progress or the maintenance of our ability to deliver.
- c) Unless subject to special agreement in writing, our prices do not include special accessories, retrofits, installation, training and other extras. The order placed by the customer can be accepted within 14 days by sending the order conformation or by delivery and subsequent invoicing.

2. Delivery Deadlines, Delays, Transfer of Risks

- a) Binding delivery deadlines must be agreed in writing. They require prior clarification of technical issues and self-supply. If we fail to deliver on time due to reasons within our control, the customer has the right to withdraw from the contract in accordance with his statutory rights after giving an additional delivery period of at least two weeks.
- b) We accept no responsibility for common negligence.
- c) We have the right to make partial deliveries. The merchandise is sent according to our own choice, and we only assume the transport costs if a special agreement was made in writing. We have the right but not the obligation to insure the merchandise at the expense of the customer.

3. Terms of Payment, Delay in Accepting Delivery

- a) If no other agreement was made in writing, our invoices are immediately due and payable net without any deductions whatsoever. If the buyer fails to settle accounts on time, interest on delinquent accounts of 4% above the bank rate of the Deutsche Bundesbank, but at least 9%, must be paid.
- b) If the buyer does not accept delivery, we have the option of insisting on acceptance or demanding 25% of the purchase price as damages, whereby the customer is obliged to furnish proof that little or no damage has been caused. Setting off is excluded except when the counter-demands are recognised as having legal force, are uncontested or are accepted by us.

4. Extended Reservation of Ownership

The merchandise delivered remain the property of the company TDi GmbH until settlement of accounts, in the case of payment by check or draft until they are honoured. The customer has the right to resell the merchandise in the ordinary course of business, but he then immediately forfeits all claims towards his buyers or to third persons from reselling for the final amount on the invoice (including VAT). In the case of breach of contract of the order, especially if he fails to pay, we have the right to take back the object of sale. This is not to be seen as withdrawal from the contract unless we declare this expressly in writing.

5. Transfer of Risks

When the merchandise is handed over to the customer or his representative or, when sent, handed over to the transporter, the risks are transferred to the buyer, regardless of who covers the transport costs.

6. Warranty, Limitation of Liability

- a) If the object of purchase has a fault for which we are responsible, we have the choice of remedying the defects or paying compensation. We must be notified of obvious faults within 14 days after receipt of the merchandise.
- b) The paragraphs §§ 377, 378 HGB valid in commercial legal relations remain untouched. If TDi GmbH is not notified of faults within 10 days of delivery of the merchandise to its destination, the merchandise is considered as accepted.
- c) The warranty claim of the customer expires in the case of interventions, repairs or attempts at repair on the part of the buyer or of unauthorised third persons. The assignment of warranty claims is excluded. Replaced parts become our property. The same warranty is given for successful replacements or repairs.
- d) If we are not willing to make, or capable of making, a replacement delivery or if three unsuccessful attempts are made to eliminate faults, the order has the right to choose to withdraw from the contract or to demand a corresponding reduction of the purchase price.
- e) We expressly declare that, due to the large number of different configurations and constant improvements with regard to printers, scanners and other peripherals, perfect functioning of the Matrix modules cannot be guaranteed in every case. The software included in the delivery and the data carriers are checked for computer viruses, but we accept no responsibility for undiscovered viruses and the resulting damage.
- f) If no other agreement is made, further claims of the customer are excluded, regardless of the legal basis. We accept no responsibility for damage which did not occur in the delivered item itself, and especially not for lost profit or other financial loss to the customer. The aforesaid limitation of liability does not apply if the damage is due to premeditation, gross negligence or the absence of a warranted qualification, infringement of substantial contractual obligations, delay in performance, impossibility or claims according to §§ 1, 4 Produkthaftungsgesetz (product liability law). If our liability is excluded or limited, this also applies to our employees, workers, representatives, servants and assistants.

7. Place of Performance

If the order conformation does not define it differently, our seat of business is the place of performance for payment and delivery.

8. Place of Jurisdiction

If the customer is a registered trader, our seat of business is the place of jurisdiction. However, TDi GmbH also has the right to sue the customer before the court responsible for his domicile.

