TECHNODATA INTERWARE GMBH

MATRIX LOCK

HIGH-QUALITY SOFTWARE PROTECTION

BENUTZERHANDBUCH

Benutzerhandbuch

Die in diesen Unterlagen enthaltenen Angaben und Daten können ohne vorherige Ankündigung geändert werden. Ohne ausdrückliche schriftliche Erlaubnis der TDi GmbH darf kein Teil dieser Unterlagen für irgendwelche Zwecke vervielfältigt oder übertragen werden, unabhängig davon, auf welche Art und Weise oder mit welchen Mitteln - elektronisch oder mechanisch - dies geschieht.
Es gelten die AGB der TDi GmbH. Hiervon abweichende Vereinbarungen bedürfen der Schriftform.

Windows ist ein eingetragenes Warenzeichen der Microsoft Corporation.
Das Windows-Logo ist ein eingetragenes Warenzeichen (TM) der Microsoft Corporation.

Software Lizenz
Die Software und die mitgelieferte Dokumentation sind urheberrechtlich geschützt. Durch die Installation erklären Sie sich mit den Vertragsbedingungen des Lizenzvertrages einverstanden.

Lizenzvertrag
TDi GmbH (kurz TDi) gewährt dem Käufer das einfache nicht ausschließliche und nicht übertragbare Lizenz-Recht, die Software auf einem einzelnen Computer, bzw. vernetzten Computersystem (LAN) zu benutzen. Das Kopieren oder jede anderweitige Vervielfältigung von Teilen oder der gesamten Software sowie das Mischen und Verbinden mit anderer Software ist ausdrücklich untersagt. Zu Sicherungszwecken darf der Käufer eine einzelne Kopie der Software für sich anfertigen (Back-up).TDi behält sich vor, die Software zu ändern, weiterzuentwickeln, zu verbessern oder durch eine neue Entwicklung zu ersetzen. Es besteht keine Verpflichtung für TDi, den Käufer über Änderungen, Neu- und Weiterentwicklungen sowie Verbesserungen zu informieren oder ihm diese zur Verfügung zu stellen. Eine rechtlich verbindliche Zusicherung bestimmter Eigenschaften wird nicht gegeben. TDi haftet nicht für Schäden, es sei denn, ein Schaden ist durch Vorsatz oder grobe Fahrlässigkeit auf Seiten der TDi oder deren Erfüllungs- und Verrichtungsgehilfen verursacht worden. Jede Haftung für indirekte sowie für Begleit- und Folgeschäden ist ausgeschlossen.

Einhaltung der CE/FCC-Bestimmungen

CE

Dieses Gerät wurde auf Einhaltung der Grenzwerte für Digitalgeräte der Klasse B geprüft und zugelassen.

Der Betrieb unterliegt folgenden Bedingungen:
1. Das Gerät darf keine schädlichen Störstrahlungen verursachen
2. Das Gerät muß Störstrahlungen verarbeiten können, einschließlich solcher Strahlungen, die zu einem unerwünschten Betrieb führen könnten.

Das Produkt erfüllt die Grenzwerte gemäß EN55022 Class B, EN50081-1, EN50082-1 and EN55024.
Eine Änderung des Produktes ohne die ausdrückliche Zustimmung von TDi GmbH, kann dazu führen, daß die CE/FCC-Bestimmungen nicht mehr erfüllt sind. In diesem Fall erlischt das Nutzungsrecht des Anwenders für dieses Produkt.

Vorwort
Geehrte Anwender! 6
Über uns 7

I Einführung
Allgemeine Beschreibung 10
Matrix-Eigenschaften im Überblick 11
Eine kurze Einführung in die Kryptographie 12
Sicherheitsstufen 14
Die »Anti-Hacker«-Sperre 15
Dongle-Modellreihen 16
Cross-Plattform 17
Architektur 17

II Installation
Installation der Software und Treiber unter Windows 22
Installation der Software und Treiber unter Linux und Mac OS X 25

III Erstellung eines geschützten Programms
Einige Hinweise bevor Sie mit dem Schutz Ihrer Anwendung beginnen 28

Vorbereitung der Matrix-Dongles 30
Verwaltung und Speicherung von Daten im Matrix-Dongle 30

Automatische Einbindung des Schutzes ohne Änderungen im Source-Code 36
Allgemeine Beschreibung 36
Grundeinstellungen 37
Geschütztes Programm mit beschränkter Laufzeit erstellen (Demo) 40
Geschütztes Programm mit unbeschränkter Laufzeit erstellen (kein Demo) . 42
Matrix-Crypt von der Command-Line verwenden 42

Manuelle Einbindung des Schutzes im eigenen Source-Code 43
Methoden der Einbindung 43
Beispiel für die Einbindung in C/C++ 44
Beispiel für die Einbindung in Visual Basic 48
Beispiel für die Einbindung in Pascal 54
Kryptographische Authentifizierung des Matrix-Dongles 59
XTEA-Verschlüsselungsreferenz 62

Netzwerk Lizenz-Management 64
Verwaltung von Netzwerklizenzen 64
Verwaltung von Netzwerklizenzen im Dongle 65
Einstellungen des MxNET Server-Programms 66
Beispiel für den Schutz einer Anwendung im Netzwerk mit MxNET 69
Was Sie unbedingt in Ihrer Anwendung berücksichtigen sollten 72
Vorteile und Nachteile des MxNET Lizenz-Management 72
Vorteile und Nachteile des Lizenz-Management mit je einem Dongle pro Arbeitsplatz 73

IV Auslieferung von geschützten Programmen
Anwendungen für Windows-Betriebssysteme 76
Was Sie Ihren Kunden mitliefern müssen 77
Anwendungen für die Betriebssysteme Linux and Mac OS X 79

Remote Update 80
Allgemeine Beschreibung 80
Vorteile von Matrix Remote Update 80
Wie funktioniert Matrix Remote Update? 81
Grundeinstellungen 81

V Richtlinien für einen guten Softwareschutz
Gibt es das »unknackbare« Programm? 86
Werkzeuge des Hackers 86
Abfrage des Dongle beim Programmstart 87
Häufige Abfragen 87
Schutzmaßnahmen während des Programmablaufes 87
Verstreuen der Abfragen im Code 87
Verwendung des Matrix-Speichers 87
Arbeiten mit der Kryptographie 88
Verwendung der Verschlüsselung für User-Authentifizierung »on-the-fly« 88
Programmiertechnik 88
Maßnahmen nach Erkennen eines Angriffes 89

VI API-Funktionen Referenz
Übersicht der API-Funktionen 92
Beschreibung der API-Funktionen 94

VII Sonstiges
Technische Daten 136
Preise und Lieferkonditionen 137
Allgemeine Geschäftsbedingungen 138

Vorwort

Geehrte Anwender!

Danke, dass Sie sich für Produkte aus dem Hause *TechnoData Interware* interessieren. Mit diesem Benutzerhandbuch möchten wir Sie so gut wie möglich im Umgang mit unterstützen. Sollen Sie noch Fragen oder Anregungen haben, stehen wir Ihnen gern auch direkt zur Verfügung.

Dieses Handbuch enthält allgemeine und spezielle Informationen zum Kopierschutz von Software und zur Einbindung von in Ihre Software. Auf unserer Site finden Sie stets die aktuelle Software und über dieses Handbuch hinausgehende Beispiele und Readme-Dateien für Mac OS X, Linux sowie zusätzliche Tools zur Optimierung Ihrer Arbeit mit . Für Verbesserungsvorschläge und Tipps sind wir stets offen.

Sie erreichen uns via per E-Mail: support@tdi-matrix.de

Über uns

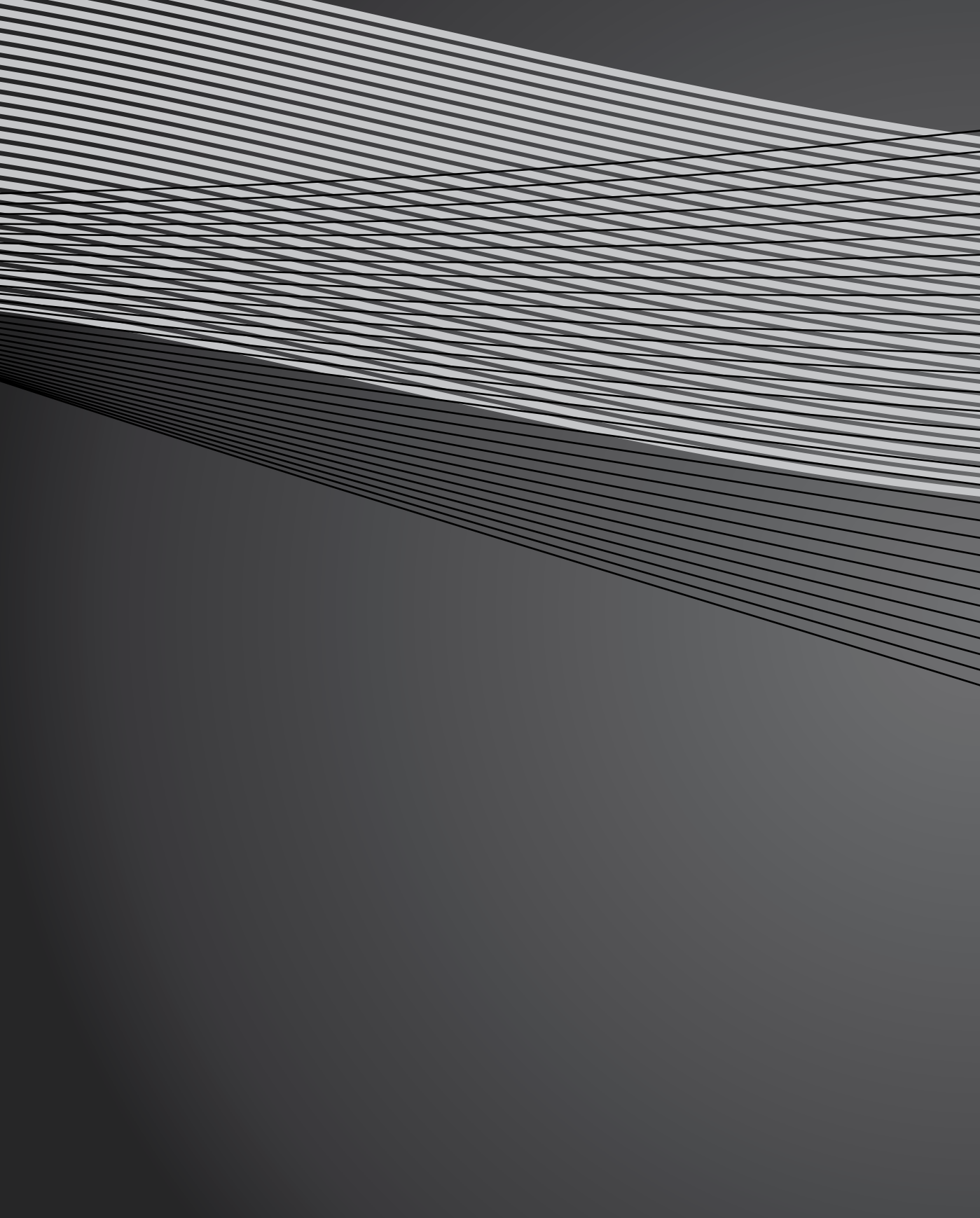
Als erfolgreicher Hersteller von Softwareprogrammen wissen wir, dass Lizenzbedingungen von den Anwendern oft nicht im erforderlichen Maß ernst genommen werden. Kontrollen und allerlei trickreiche Kopierschutzmaßnahmen, wie z. B. die Generierung von geräteabhängigen Lizenznummern, sind immer wieder umgangen worden.

Da wir auf dem Markt kein hinreichend sicheres Kopierschutz-System zu akzeptablen Preisen finden konnten, haben wir seinerzeit selbst eine Lösung für unser Raubkopierproblem entwickelt:

ist mittlerweile weltweit in unterschiedlichsten Konfigurationen im Einsatz und schützt erfolgreich Investitionen in der Software-Entwicklung. Und so kommt es nach Einführung des Dongles häufiger zu bemerkenswerten Nachbestellungen bei unserer Kunden, deren User die Software vermutlich gleich mehrfach installiert hatten.

Verschiedene Strategien der Dongle-Umgehung, z. B. die Simulation der Hardware durch Hilfsprogramme oder die Simulation der Kommunikation zwischen dem Dongle und den geschützten Programmen, sind durch die bei eingesetzte Verschlüsselungstechnik zum Scheitern verurteilt. Die vom Dongle gelieferten Werte sind nicht zu standardisieren, sie werden bei jeder Abfrage durch nach dem Zufallsprinzip erzeugte Parameter verändert. Aus unserer bisherigen Praxis ist uns kein Fall bekannt, bei dem eines unser Programme geknackt werden konnte. Zumindest wird durch Matrix der Aufwand für potentielle »Dongle-Knacker« unvertretbar hoch.

Profitieren Sie also von unseren Erfahrungen und schützen auch Sie Ihre Entwicklungsinvestitionen erfolgreich mit .



1

Einführung

Einführung

Allgemeine Beschreibung

Matrix ist ein zuverlässiges Sicherheitssystem zum Schutz Ihrer Software vor unautorisierter Vervielfältigung. Die Verwendung von *Matrix* entfaltet sich mit den Ideen und mit der Kreativität des Entwicklers. Gerade im Bereich der Überwachung, der Auswertung und der Distributions-Unterstützung (z. B. Pay By Use) gibt es viele Einsatzfelder. Des Weiteren kann *Matrix* die Basis für IT-Sicherheit und Datenschutz sein!

Matrix wurde für die Druckerschnittstelle und für die USB-Schnittstelle des PCs entwickelt:

- Die Matrix-Dongle der ML und der MK-Serie für die Druckerschnittstelle werden einfach auf die LPT-Schnittstelle gesteckt und funktionieren dann problemlos ohne Störungen für jene Peripheriegeräte (Drucker, Scanner usw.) die noch dahinter angeschlossen werden.

- Die Matrix-Dongle der MLU- und MKU-Serie für die USB-Schnittstelle sind funktionsgleich mit den Dongle für den LPT-Anschluss und ermöglichen den Softwareschutz auch bei Laptops und PCs, die über keine parallele Druckerschnittstelle verfügen.

Bei der Entwicklung wurde besonders auf *transparentes Verhalten, hohe Sicherheit* durch Verwendung ei nes RISC-Prozessors, *einfache Anbindung* in Ihre Software und *große Zuverlässigkeit* in der Praxis geachtet.

Die Matrix-Dongle sind zusätzlich stapelbar - das bedeutet, dass mehrere Dongle gleichzeitig an die LPT- und/oder USB-Schnittstelle angeschlossen werden können. Auch beim Anschluss mehrerer Dongle an der gleichen Schnittstelle, kann man selbstverständlich jeden einzelnen Dongle gezielt ansprechen/abfragen.

Für die Einbindung in Ihre Software stehen Ihnen die mitgelieferten Matrix-DLLs zur Verfügung. Die mitgelieferten DLLs gewährleisten eine einfache Einbindung in jeder Programmiersprache.

Matrix-Eigenschaften im Überblick

Das »Matrix Software Protection System« bietet folgende Vorteile:

- Verfügbar für die LPT- und für die USB-Schnittstelle
- Erstellung von Demo-Versionen
- Speicherung fester Daten in den Matrix-Dongles
- Verwendung eines einzigen Matrix-Dongles zum Schutz mehrerer Anwendungen
- Verwaltung von Netzwerklizenzen mit nur einem einzigen Matrix-Dongle
- Die gleiche Hardware wird eingesetzt für den Schutz von Einzelplatz- und Netzwerkprogrammen
- 128-Bit Ver- und Entschlüsselung von Daten über den Matrix-Dongle
- Für die Ver-/Entschlüsselung frei wählbarer 128-Bit Schlüssel
- Der 128-Bit Schlüssel ist aus dem Matrix-Dongle nicht auslesbar
- Cross-Plattform (Windows, Linux, Mac) ohne proprietären USB-Treiber
- API-Unterstützung für 16, 32 und 64-Bit Programme
- Integration des Schutzes in 32-Bit EXE-Dateien ohne Programmieraufwand
- Sichere und flexible Änderung Ihrer Kunden-Dongles per »Remote-Update«
- Stapelbarkeit mehrerer Dongle hintereinander
- Mehrere Speichergrößen verfügbar
- Unschlagbares Preis-/Leistungsverhältnis
- Der gleiche Dongle kann eigesetzt werden für Softwareschutz und für Web-Logon

Eine kurze Einführung in die Kryptographie

Die ersten Verschlüsselungstechniken waren denkbar einfach. Man legte eine Vertauschungsregel fest, nahm die zu verschlüsselnde Nachricht und tauschte die Buchstaben jeweils durch einen später im Alphabet folgenden Buchstaben aus. Wichtige Nachrichten waren hierdurch für Unbefugte unleserlich und konnten erst durch Kenntnis der Vertauschungsregel wieder lesbar gemacht werden.

Verschlüsselung und Entschlüsselung

Unverschlüsselte Daten werden Klartext genannt. Um diese Daten vor unbefugtem Zugriff zu schützen, werden diese chiffriert - also mit einem Verschlüsselungsverfahren in zu einer unleserlichen Zeichenkette konvertiert. Mit der Entschlüsselung wird diese Zeichenkette wieder in den ursprünglichen Klartext rückgewandelt.

Was versteht man unter Kryptographie?

Vertrauliche Daten, die z. B. über das Internet übertragen werden sollen, können wegen der Gefahr des Missbrauchs nicht im Klartext gesendet werden. Mit den Techniken zur Sicherung dieser Daten befasst sich die Wissenschaft von der Kryptographie (»Geheimschrift«), zu der auch die Kryptoanalyse gehört. Bei der Kryptoanalyse wird versucht, Strukturen innerhalb der verschlüsselten Daten zu erkennen und so den passenden Decodierschlüssel zu ermitteln.
Die Kryptographie bedient sich mathematischer Verfahren zur Ver- und Entschlüsselung von Daten.

Stärke der Verschlüsselung

Der Grad oder die Stärke der Verschlüsselung wird daran gemessen, welcher Aufwand und welche Zeit benötigt wird, um die Entschlüsselung durchzuführen. Im Idealfall entsteht durch das Verschlüsselungsverfahren ein chiffrierter Text, der ohne ein geeignetes Decodierverfahren kaum zu entschlüsseln ist.
Es gibt Theorien, nach denen eine starke Verschlüsselung selbst mit allen derzeit verfügbaren Rechnerkapazitäten nicht vor dem Ende des Universums zu knacken ist. Absolute Sicherheit kann man hieraus aber leider nicht ableiten; vielleicht widerlegt bereits eine der nächsten Computer-Generationen diese Theorien.
Auch bei stärkster Verschlüsselung ist eine gewisse Wachsamkeit und ein konservativer, vorsichtiger Umgang mit sensiblen Daten sicherlich ein unverzichtbarer Teil des Datenschutzes. Matrix verwendet ein starkes Kryptographie-Verfahren: Mit Hilfe eines 128 Bit-Schlüssels werden die zu schützenden Daten in einer XTEA-Routine 32 mal hintereinander verschlüsselt. Das bedeutet, die Daten werden verschlüsselt, die dann verschlüsselten Daten ein weiteres mal verschlüsselt und nochmal verschlüsselt und nochmal.... – wie gesagt 32 mal. Nach dem heutigen Kenntnisstand der Kryptoanalyse wird bereits

eine sechsmalige Verschlüsselung als ausreichend sicher angesehen. Die Daten durchlaufen bei der Entschlüsselung ebenfalls 32 mal die Entschlüsselungs-Routine.

Funktionsweise der Kryptographie

Zur Ver- und Entschlüsselung der Daten wird eine mathematische Funktion verwendet, auch Verschlüsselungs-Algorithmus genannt. Dieser Algorithmus erhält als Input die zu verschlüsselnden Daten und einen Schlüssel, z. B. eine Zahl. Aus den Eingabedaten im Klartext wird mit dem ebenfalls übergebenen Schlüssel das Ergebnis, also der chiffrierte Text errechnet. Ein und derselbe Klartext führt bei Verwendung verschiedener Schlüssel zu unterschiedlichem verschlüsseltem Text.

Hierbei bestimmt die Art des Algorithmus und die Strenge der Geheimhaltung des Chiffrierschlüssels den Grad der Datensicherheit. Im Matrix-Dongle ist sowohl der Codierschlüssel als auch der Verschlüsselungsalgorithmus gespeichert. Diese Daten sind für unbefugte nicht mehr zugänglich.

Konventionelle Verschlüsselung

Unter der konventionellen Verschlüsselung wird die Verschlüsselung mit symmetri-schen bzw. Geheimschlüsseln verstanden. Hier wird zur Ver- bzw. Entschlüsselung der Daten jeweils der selbe Schlüssel verwendet. Der so genannte XTEA (»eXtended Tiny Encryption Algorithm«) Verschlüsselungs-Algorithmus des Matrix-Dongle ist ein konventionelles Verschlüsselungssystem.

Schlüsselverwaltung und konventionelle Verschlüsselung

Der Vorteil der konventionellen Verschlüsselung ist die Geschwindigkeit, dieses Verfahren ist sehr schnell. Das Problem jedoch ist die Geheimhaltung und die Übertragung des Codierschlüssels an den Empfänger. Wird der Schlüssel bei der Übertragung abgefangen, kann er jederzeit zur Decodierung der verschlüsselten Daten verwendet werden und der beabsichtigte Schutz ist unwirksam. Die Schwachstelle bei diesem Verfahren ist also der Weg des Schlüssels bzw. die Übertragung zum Empfänger. Wie also beseitigt man dieses Problem?

Das Matrix-Verschlüsselungskonzept und die Schlüsselverwaltung

Mit Matrix wurde das Problem der Schlüsselverteilung so gelöst, dass die Verschlüsselung oder Entschlüsselung immer im sichersten Teil des gesamten Kopierschutz-Systems, nämlich im Matrix-Dongle stattfindet. Die Anwendung sendet nur Datenpakete zum Ver-/Entschlüsseln an den Dongle, aber niemals den Schlüssel selbst.
Der Schlüssel, der der strengsten Geheimhaltung unterliegt, befindet sich nur im Dongle und tritt somit niemals in Erscheinung – weder an der Schnittstelle Applikation <-> DLL, noch an der LPT/USB-Schnittstelle; zudem kann der Schlüssel auch nicht auf irgendeine Weise aus dem Dongle ausgelesen werden. Alles folgt dem Prinzip: *»Was nicht übertragen wird, kann auch nicht abgefangen und mißbraucht werden«.*

Sicherheitsstufen

Die ML- und MK-Modelle

Die Dongle der LPT und der USB-Typen sind jeweils in zwei Modellreihen gegliedert: Die ML-Serie und die MK-Serie.

Beide Modellreihen bieten die gleichen Leistungen. Für eine höhere Sicherheit gegen Datenmanipulation ist jedoch die Speicherung der Daten in den Dongle der MK-Serie nur mit einem sogenannten *MasterKey*-Dongle möglich.

Der MasterKey-Dongle wird zusammen mit dem Kopierschutz-Dongle ebenfalls an den LPT/USB-Anschluss gesteckt. Der Masterkey wird für jeden Besteller der MK-Serie kostenlos einmalig angefertigt. Dadurch wird immer sichergestelt, dass die Daten im Dongle nur von demjenigen, der über einen gültigen MasterKey verfügt, geändert werden können.

Die Matrix-Dongle werden mit einem Kunden-Code, *UserCode* genannt, geliefert. Dieser Kunden-Code ist nicht veränderbar und bietet die Sicherheit, dass jeder Software-Hersteller nur seine eigenen Dongle programmieren kann. Der UserCode wird einmalig bei der Erstbestellung vergeben und bleibt auch bei Nachbestellungen unverändert.

Bei allen Modellreihen ist die Kommunikation mit dem PC verschlüsselt. Die zwischen dem PC und dem Dongle in beide Richtungen ausgetauschten Daten werden vor der Übertragung verschlüsselt und erst bei der Auswertung wieder entschlüsselt. Die Entschlüsselung findet sowohl auf der Seite des PC als auch im Dongle statt.
Der Verschlüsselungsalgorithmus für die Kommunikation mit dem PC verändert sich ständig, somit sind die ausgetauschten Informationen für den unautorisierten Hacker wertlos.

128-Bit Ver- und Entschlüsselung von Daten

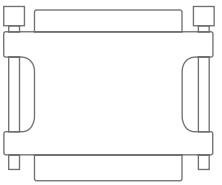
Die Dongle ab ML/MK-60 bzw. ab MLU/MKU-60 ermöglichen die interne Verschlüsselung und Entschlüsselung von Daten mit einem aus dem Dongle nicht auslesbaren 128Bit-Schlüssel. Dieser Schlüssel kann vom Software-Hersteller beliebig festgelegt werden.

Die »Anti-Hacker«-Sperre

Die Daten können aus dem Dongle gelesen bzw. in den Dongle gespeichert werden, und dies nur durch Angabe des korrekten UserCodes. Bei dem Versuch, den UserCode zu ermitteln, indem man die Read/Write-Funktionen – z. B. mit UserCodes zwischen 1 und nnnnn – aufruft, wird eine sogenannte »Anti-Hacker«-Sperre aktiviert. Dadurch wird der Dongle funktionsunfähig und kann auch durch Angabe des richtigen UserCodes nicht mehr angesprochen werden.
Die »Anti-Hacker«-Sperre kann nur noch von dem Software-Hersteller mit einem MasterKey aufgehoben werden.
Die »Anti-Hacker«-Sperre stellt keine Gefahr dar, wenn z. B. mehrere Dongle hintereinander gesteckt sind und Sie dann mit Ihrem Programm jedes einzelne abfragen, um Ihr eigenes im Stapel zu finden.
Zwangsläufig werden Sie fremde Dongle (von anderen Software-Herstellern) mit Ihrem UserCode zu lesen versuchen, aber dadurch wird die »Anti-Hacker«-Sperre nicht aktiviert. Die Sperre aktiviert sich nur in dem Fall, wenn ein Angreifer immer den gleichen Dongle mit ständig falschen und unterschiedlichen UserCodes zu lesen versucht – das heißt z. B. wenn ständig nur der Dongle 1 mit falschen und wechselnden UserCodes (z. B. in einer Schleife von 1 bis nnnn) abgefragt wird.

Dongle-Modellreihen

Für Ihr maßgeschneidertes Lizenzmanagement bieten wir Ihnen verschiedene Speichergrößen an:



Modelle für die LPT-Schnittstelle		
ML-12	12 Byte	Speicherung von max. 3 Zahlencodes (je 32-Bit unsigned)
ML-60	60 Byte	Speicherung von max. 15 Zahlencodes (je 32-Bit unsigned) 128-Bit Ver-/Entschlüsselung von Daten möglich.
ML-316	316 Byte	Speicherung von max. 79 Zahlencodes (je 32-Bit unsigned) 128-Bit Ver-/Entschlüsselung von Daten möglich.
MK-Serie		Identisch mit den Modellen der ML-Serie. Für die Speicherung der Daten wird jedoch ein zusätzlicher Dongle (MasterKey) benötigt.

Modelle für die USB-Schnittstelle		
MLU-60	60 Byte	Speicherung von max. 15 Zahlencodes (je 32-Bit unsigned) 128-Bit Ver-/Entschlüsselung von Daten möglich.
MLU-316	316 Byte	Speicherung von max. 79 Zahlencodes (je 32-Bit unsigned) 128-Bit Ver-/Entschlüsselung von Daten möglich.
MKU-Serie		Identisch mit den Modellen der MLU-Serie. Für die Speicherung der Daten wird jedoch ein zusätzlicher Dongle (MasterKey) benötigt.

Alle Matrix-Modelle sind stapelbar/anreihbar; das bedeutet, dass mehrere Dongle gleichzeitig an den selben LPT-und/oder USB-Anschluss gesteckt werden können. Dafür sind ***keine*** Änderungen in den Parametern der Dongle notwendig. Der gleichzeitige Betrieb mehrerer Dongle erfolgt ohne jeden Aufwand und das Abfragen eines bestimmten Dongle im Stapel ist aufgrund der gezielten Abfragemöglichkeit problemlos.

Cross-Plattform

Matrix-Dongle unterstützen zwei USB-Betriebsarten:
- Mit eigenem Treiber »Driver-Mode«
- Ohne eigenen Treiber »HID-Mode«

Diese zwei Betriebsarten sind besonders unter Windows relevant, da unter Windows nur USB-Geräte von Typ »HID« ohne einen proprietären Treiber verwendet werden können.

Matrix USB-Dongle funktionieren in den Betriebssystemen Windows, Linux, Mac ohne proprietären USB-Treiber:
- Unter Linux ohne eigenen Treiber in beiden Betriebsarten.
- Unter Windows ohne eigenen Treiber nur in der Betriebsart »HID-Mode«.
- Unter Mac-OSX ohne eigenen Treiber in beiden Betriebsarten.

Der USB-Dongle kann auf die gwünschte USB-Betriebsart eingestellt werden.
Wenn Sie den Dongle auf »HID-Mode« einstellen, braucht Ihr Kunde unabhängig vom eingesetzten Betriebssystem gar keinen USB-Treiber zu installieren.

Architektur

Die Kommunikation zwischen Ihrer Software und Matrix-Dongle erfolgt über die mitgelieferte Matrix-API:
- DLL-Dateien unter Windows
- Shared-Libraries unter Linux und Mac-OSX.

Mit Matrix können unter Windows sowohl 16-Bit, 32-Bit wie auch 64-Bit Programme geschützt werden.
Auch 16-Bit Programme, die unter Windows NT/2000/XP/Vista ablaufen müssen, können problemlos mit Matrix geschützt werden, da der Übergang von 16 Bit auf 32 Bit automatisch von den Matrix-DLLs gesteuert wird.

LPT-Anschluss

Die Abfrage der Dongle für den LPT-Anschluss in Ihrem Programm wird unter Windows 95/98 über einen VXD-Treiber und unter Windows NT/2000/XP/Vista über einen SYS-Treiber gesteuert.
Die Verwendung des VXD-Treibers unter Windows 95/98 ist optional, hier kann die Abfrage auch direkt hardwaremäßig ohne VXD-Treiber erfolgen.

USB-Anschluss

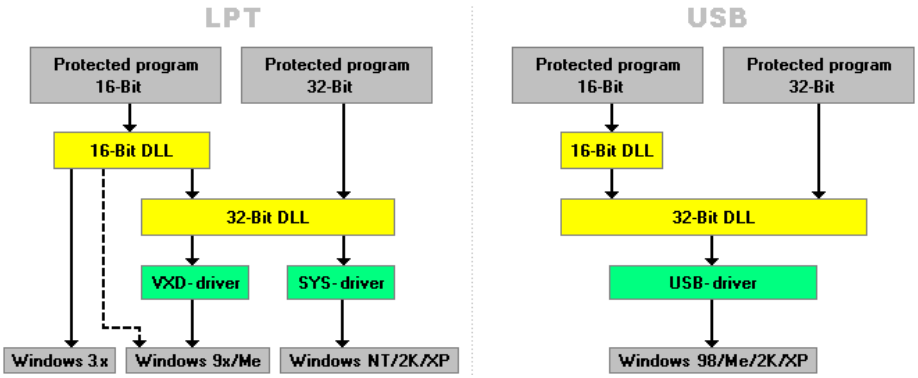
Die Matrix USB-Dongle unterstützen grundsätzlich die Betriebsart »Driver-Mode«, was soviel bedeutet, dass für den Donglebetrieb ein Systemtreiber installiert werden muss. Ab Hardware Version 5.0 unterstützen die USB-Dongle auch zusätzlich die Betriebsart »HID-Mode«.

- In der Betriebsart »HID-Mode« wird der Matrix-Dongle automatisch vom System erkannt. Der Vorteil dieser Betriebsart besteht darin, dass keine Treiber ausgeliefert und installiert werden müssen.

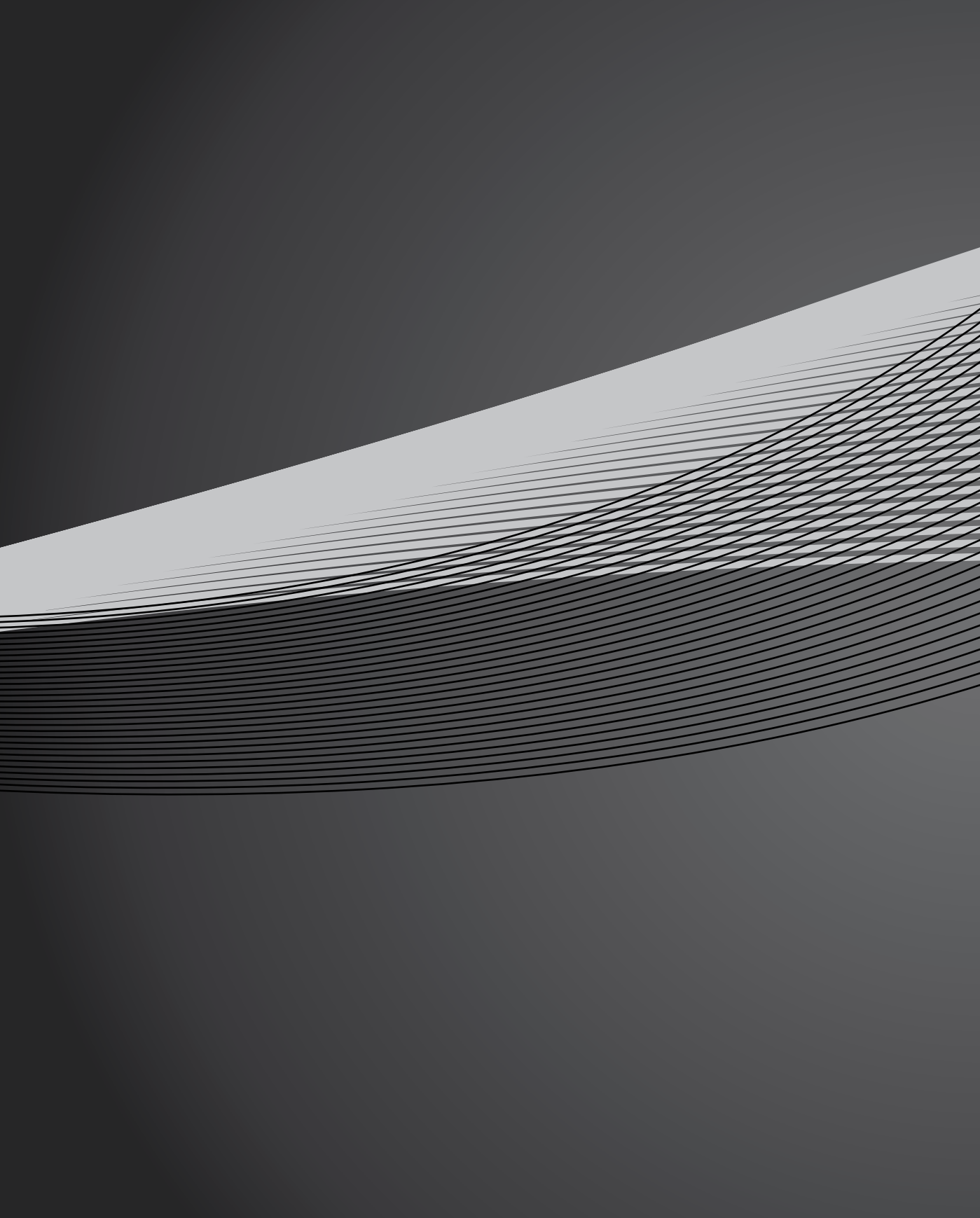
- In der Betribsart »Driver-Mode« ist es erforderlich, einen USB-Treiber auszuliefern und auf dem Zielrechner zu installieren.
  In dieser Betriebsart ist der Dongle geringfügig schneller als in »HID-Mode«.

Sie haben die Möglichkeit, mit den mitgelieferten Software-Tools die gewünschte USB-Betriebsart einzustellen (mit oder ohne Treiber).

Mit dem folgenden Schema wird die Funktionsweise erläutert, wie jene 16-Bit und 32-Bit Programme, die mit Matrix geschützt sind, unter Windows 3.x, 95/98, NT, 2000 und XP/Vista funktionieren.



Die Kommunikation mit dem LPT-Dongle kann unter Windows 95/98 auch ohne VXD-Treiber erfolgen. Um Konflikte beim Zugriff auf dem LPT-Port zu vermeiden, empfiehlt sich jedoch die Kommunikation über den VXD-Treiber zu betreiben.



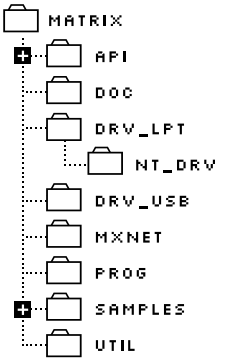
2

Installation

Installation

Installation der Software und Treiber unter Windows

Starten Sie von der CD das Programm ***setup.exe***. Nach der Installation finden Sie auf Ihrem Laufwerk das Verzeichnis ***matrix*** mit folgendem Aufbau:



API *Application Programming Interface*

In diesem Unterverzeichnis befinden sich die 16 Bit-, 32 Bit und 64-Bit API mit den entsprechenden Bibliotheken, die Sie für die Einbindung in Ihre Software benötigen.

DOC

Hier finden Sie dieses Benutzerhandbuch im PDF-Format.

DRV_LPT *Driver LPT*

Dieses Verzeichnis beinhaltet die LPT-Treiber für Win9x/ME und NT/2000/XP/Vista.

NT_DRV

In diesem Verzeichnis finden Sie das Installationsprogramm für den Windows NT/2000/XP/Vista LPT-Treiber.

DRV_USB *Driver USB*

Dieses Verzeichnis beinhaltet den USB-Treiber für Win98/ME/2000/XP/XP-64,Vista, sowie ein Installationsprogramm mit dem der Treiber vorinstalliert werden kann.

MXNET

Hier finden Sie das MxNet-Serverprogramm für die Verwaltung von Netzwerklizenzen.

PROG

Unter diesem Verzeichnis finden Sie die Programme, die Sie zum inhouse Programmieren Ihrer Dongle benötigen.

SAMPLES

Hier finden Sie Beispiele für die Integration des Schutzes in verschiedenen Programmiersprachen.

UTIL

Hier finden Sie das Programm MxCheck für die Überprüfung der installierten API und Treiber im System. Dieses Programm können Sie an Ihren Kunden als Support-Tool liefern.

Bei der Installation kopiert die Setup-Routine die Matrix-API ***matrix16.dll*** und ***matrix32.dll*** und den LPT-Treiber ***iwport.vxd*** für Windows 9.x/ME in das Verzeichnis ***\windows\system*** und den LPT-Treiber ***iwport.sys*** für Windows NT/2000/XP/Vista in das Verzeichnis ***\windows\system32\drivers***.

LPT-Treiberinstallation unter Windows 95/98/ME

Unter Windows-9x/ME erfolgt die Kommunikation mit dem Dongle am LPT-Anschluss über einen entsprechenden VXD-Treiber. Da dieser Treiber nicht explizit installiert und registriert werden muss, reicht es aus, die Treiberdatei ***iwport.vxd*** in das Verzeichnis ***\windows\system*** zu kopieren.

LPT-Treiberinstallation unter Windows NT/2000/XP/Vista

Unter Windows-NT/2000/XP/Vista erfolgt die Kommunikation mit dem Dongle am LPT-Anschluss über einen entsprechenden SYS-Kerntreiber, der installiert und automatisch registriert wird.

Die Treiberinstallation kann automatisch oder manuell erfolgen:

- Die automatische Installation wird von der Matrix-API beim ersten Programmstart vorgenommen, sofern die Treiber-Datei ***iwport.sys*** im Verzeichnis ***\windows\system32\drivers*** vorhanden ist und sofern Sie über Admin-Rechte verfügen. Durch diese eingebaute Installationsfunktion wird der Installationsvorgang deutlich vereinfacht. Es reicht also aus, die Treiberdatei ***iwport.sys*** ins Verzeichnis ***\windows\system32\drivers*** zu kopieren. Die einzige Voraussetzung ist, dass Sie beim erstmaligen Programmstart über Admin-Rechte verfügen.

- Die manuelle Installation/Deinstallation des Treibers kann mit dem dazugehörigen Installationsprogramm ***drv_inst.exe*** aus dem Verzeichnis ***\nt_drv*** durchgeführt werden. Zusätzliche Informationen zum Programm ***drv_inst.exe*** finden Sie in der Readme-Datei im Verzeichnis ***\nt_drv***.

Bedenken Sie, dass unter Windows NT/2000/XP/Vista nur Benutzer mit entsprechenden Zugriffsrechten (wie z. B. der Administrator) Treiber installieren können.

USB-Treiberinstallation unter Windows 98/ME/2000/XP/Vista

Die Matrix USB-Module unterstützen grundsätzlich die Betriebsart »Driver-Mode«. In dieser Betriebsart benötigt der USB-Dongle einen entsprechenden Systemtreiber, der installiert werden muss.
Ab Hardware-Version 5.0 unterstützen die USB-Dongle zusätzlich auch die Betriebsart »HID-Mode«.
In der Betriebsart »HID-Mode« wird der systemeigene HID-Treiber verwendet. Eine Treiberinstallation ist in dieser Betriebsart nicht erforderlich. Unter Windows 2000/XP/Vista ist der HID-Treiber standardmäßig im System vorhanden. Unter Windows 98/ME wird der HID-Treiber nur nach Bedarf installiert. Aus diesem Grund wird die Windows-CD verlangt, wenn der HID-Treiber noch nicht im System vorliegt.

Damit Ihnen beide Betriebsarten zur Verfügung stehen, wird mit dem Setup-Programm auch der proprietäre USB-Treiber installiert.
Im folgenden Abschnitt wird detailiert die Installation des proprietären USB-Treibers beschrieben:

Da der Matrix USB-Dongle ein Plug&Play-Gerät ist, werden Sie beim erstmaligen Anschließen des Dongle automatisch vom System aufgefordert, einen entsprechenden Treiber anzugeben.
Im Plug&Play-Wizard wird dann angegeben, in welchem Verzeichnis sich der USB-Treiber befindet (z. B. das Verzeichnis ***\drv_usb***).

Die einmalige Treiberinstallation ist unter Windows 98/ME/2000 ausreichend. Danach findet und installiert Windows den Treiber automatisch, auch wenn der Matrix USB-Dongle an einen anderen USB-Anschluß eingesteckt wird (z.B. an einem anderen Steckplatz eines HUBs).

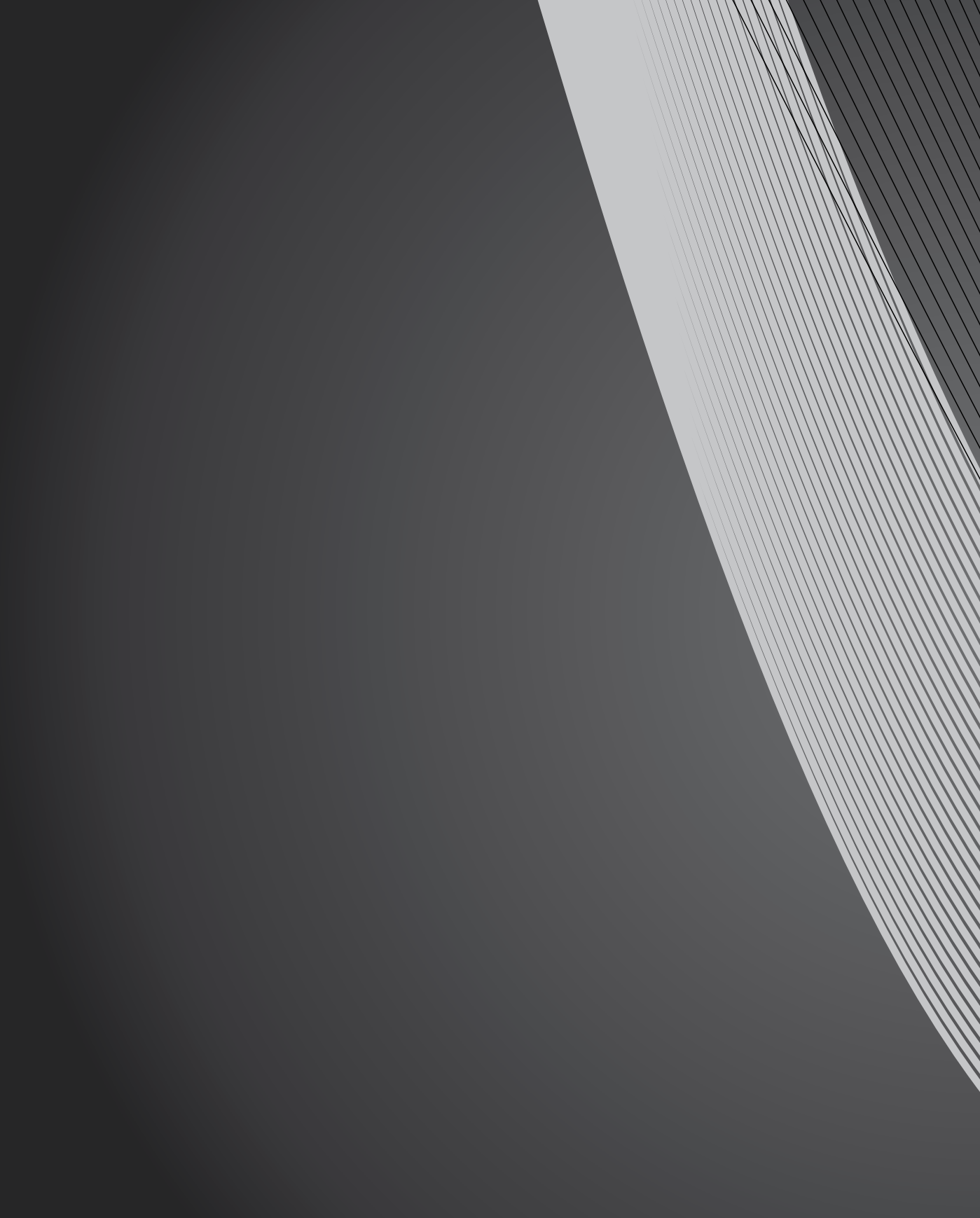
Unter Windows XP/Vista dagegen werden Sie aufgefordert, für jeden verfügbaren USB-Steckplatz den Treiber neu zu installieren. Diese lästige Wiederholung der Treiberinstallation kann umgangen werden, indem die INF-Datei des USB-Treibers vorinstalliert wird. Für diesen Zweck steht das Programm ***inf_inst.exe*** im Verzeichnis ***\drv_usb*** zur Verfügung. Durch die Vorinstallation (auch »Pre-Installation« genannt) der INF-Datei, wird dem Windows Hardware-Manager bekannt gemacht, wo sich der USB-Treiber befindet. Auch nach der Pre-Installation der INF-Datei wird der Plug&Play-Wizard erscheinen, aber die Angabe der Treiberposition ist dann nicht mehr nötig. Ein Klick auf die Schaltfläche ***Weiter >*** reicht aus und alles erfolgt automatisch.

Hinweis: Es ist empfehlenswert die Pre-Installation der INF-Datei in allen Windows-Versionen durchzuführen. Zusätzliche Informationen zum Programm ***inf_inst.exe*** *finden Sie in der Datei Readme im Verzeichnis* ***\drv_usb****.*

Installation der Software und Treiber unter Linux und Mac-OS X

Für Linux und Mac-OS X lesen Sie bitte die Installations-Anleitung aus der entsprechenden Readme Datei.

Hinweis: Unter www.tdi-matrix.de finden Sie immer die aktuellen Versionen und neuesten Tools zum Download.



3

Erstellung eines geschützen Programms

Erstellung eines geschützten Programms

Einige Hinweise bevor Sie mit dem Schutz Ihrer Anwendung beginnen

Das Matrix Security System stellt Ihnen mehrere Methoden zur Verfügung, um Ihre Anwendung zu schützen:

Automatische Einbindung des Schutzes (ohne Änderungen im Source-Code)
Ist die einfachste und schnellste Methode, um 32-Bit Windows Anwendungen, ohne Änderungen im Source-Code, zu schützen (Matrix-Crypt).

Manuelle Einbindung des Schutzes (in den eigenen Source-Code)
Ermöglicht die Integration des Schutzes in Ihren Source-Code mittels API-Funktionen.

Sie können eine Methode verwenden, oder auch beide kombinieren. Folgende Vergleichstabelle soll Ihnen dabei helfen, die passende Schutzmethode für Ihre Anforderungen auszuwählen.

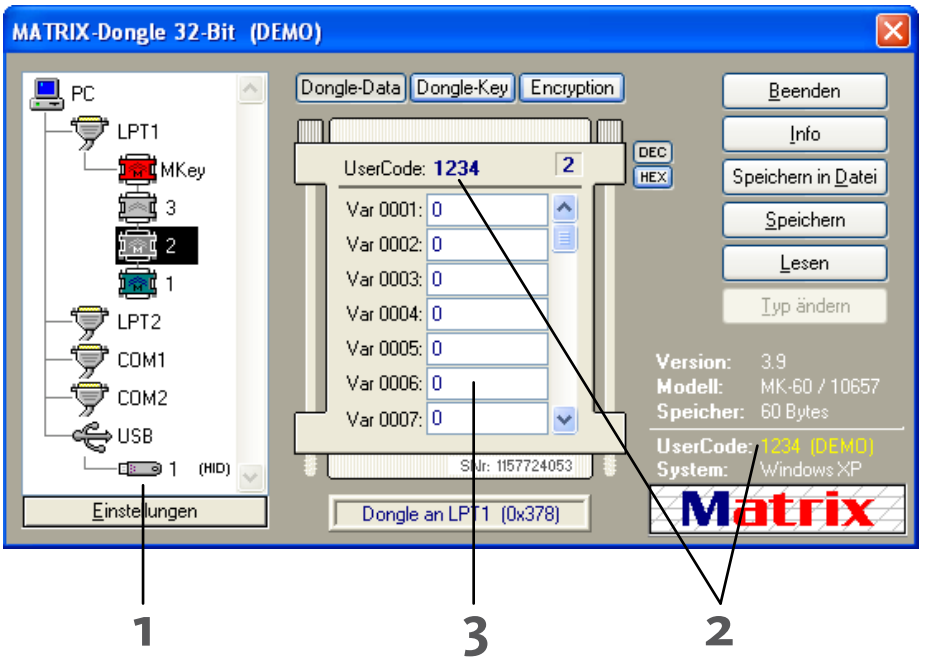
Automatische Einbindung	Manuelle Einbindung
Erfordert keine Änderungen im Source-Code.	Source-Code muss verfügbar sein.
Schneller und einfacher Schutz ohne jeden Programmieraufwand; es ist auch die einfachste Methode, um Demo-Versionen zu erstellen (z.B. mit limitierter Anzahl der Starts). Diese Methode bietet jedoch nur eine feste Anzahl an Einstellungen, die Sie für den Schutz vornehmen können.	Erfordert etwas mehr Aufwand, bietet Ihnen aber die maximale Flexibilität. Durch die Integration in Ihren Source-Code haben Sie die Möglichkeit zu entscheiden, welche Dongle Ihrer Software aktiv sind und welche nicht. Sie liefern an alle Kunden die gleiche Software und sparen dadurch Zeit und Administrationskosten.
Einfache Integration von Schutzmaßnahmen gegen Debug und »Reverse Engineering«.	Sie müssen Ihre eigenen Schutzmaßnahmen gegen Debug und »Reverse Engineering« verwenden, oder Sie können auch den Automatischen-Schutz zusätzlich zum Manuellen-Schutz verwenden.
Wenn der Matrix-Dongle nicht angeschlossen ist, stoppt die Anwendung.	Sie können Ihr eigenes Reaktionsschema definieren, wie Ihre Anwendung reagieren soll, wenn der Matrix-Dongle nicht angeschlossen ist. Zum Beispiel: Programm stoppen, Menus ausschalten, Wechsel zum Demo-Modus, usw.
Ermöglicht die Integration des Schutzes in »Stand-Alone« Anwendungen. Für Netzwerk-Anwendungen benötigen Sie einen Matrix-Dongle für jede Arbeitsstation.	Sie können Ihre Netzwerk-Verwaltung mit einem einzigen Matrix-Dongle pro Netzwerk realisieren. Die starke Dongle-Verschlüsselung können Sie nutzen, um Anwendungen aus dem Bereich User-Authentication, Secure Business-To-Business oder Two-Factor-Authentication zu erstellen.

Empfehlung: *Sofern Ihnen der Quelltext Ihrer Anwendung vorliegt, empfehlen wir grundsätzlich die manuelle Einbindung. Dies erfordert zunächst mehr Arbeit. Dafür ermöglicht es ein maßgeschneidertes Lizenzmanagement und eine Ausnutzung beliebiger Matrix-Funktionen, volle Felxibilität und einen differenzierten Zugriff. Auch haben Sie weitreichende Möglichkeiten der Interaktion zwischen Ihrer Applikation und dem Dongle.*
Eine Netzwerkanwendung ist beispielsweise nur mit manueller Einbindung möglich.

Vorbereitung der Matrix-Dongles

Verwaltung und Speicherung von Daten im Matrix-Dongle

Für die Verwaltung eigener Daten im Matrix-Dongle steht Ihnen nach der Installation ein komfortables Programm zur Verfügung. Das Verwaltungsprogramm wird in zwei Ausführungen geliefert: 16-Bit und 32-Bit. Demnach können auch noch Widows 3.x-Anwender die Dongle problemlos programmieren.



Die Bedienung dieses Programms ist recht einfach.

1. Auf der linken Seite werden die im PC vorhandenen Ports und die angeschlossenen Matrix-Dongle angezeigt. Mit der Schaltfläche »Lesen« kann jederzeit die Anzeige aktualisiert werden.

2. Die Kundennummer, auch »UserCode« genannt, wird Ihnen bei der ersten Bestellung zugeordnet und ist nicht veränderbar. Diese exklusive Nummer gewährleistet den Zugriff ausschließlich auf Daten im eigenen Dongle.

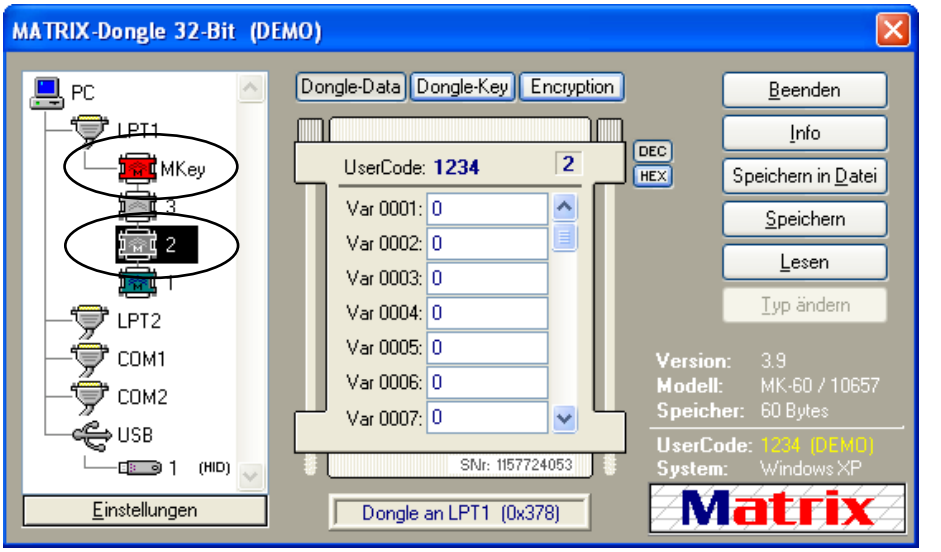
3. Die Dongle-Felder (Var001-VarXXX) können Sie frei verwalten.
   Es können Zahlen im Bereich von 0 - 4294967295 (hex: 0x00 - 0xFFFFFFFF) aufgenommen werden.

Hinweis: Die Speichergröße des Dongle dividiert durch 4 ergibt die maximale Anzahl der speicherbaren Variablen.

Beispiel: Bei einem Dongle vom Typ ML-60 mit 60 Byte stehen 60 / 4 = 15 Variablen zur Verfügung.

Sonderfunktion der MK-Serie

Dongles der MK-Serie (kurz für »MasterKey«-Serie) lassen sich nur programmieren, wenn beim Programmiervorgang ein sogenannter »MasterKey«-Dongle aufgesteckt wird. Bei Bestellern der MK-Serie gehört der MasterKey-Dongle zum kostenlosen Lieferumfang der ersten Bestellung.



Bei der Programmierung der MK-Serie muss der MasterKey-Dongle auf demselben oder einem anderen LPT/USB-Port des Computers aufgesteckt werden. Nur dann sind Schreibzugriffe auf die zu programmierenden Dongle möglich.

Die Existenz des MasterKey-Dongle wird ebenfalls im linken Fensterbereich dargestellt. Dongle der MK-Serie sind, um Verwechselungen auszuschließen, mit einem **M** gekennzeichnet.

Mit der Schaltfläche »Speichern« werden die Daten in den links markierten Dongle gespeichert.

Mit der Schaltfläche »Speichern in Datei« werden die Dongle-Daten in einer ASCII-Datei abgelegt. Diese kann, um Tipparbeit zu sparen, im Programm Mx-Crypt übernommen werden.

Mit der Schaltfläche »Typ ändern« kann die Betriebsart der USB-Dongle ab V5.0 umgestellt werden: »HID-Mode«/»Driver-Mode«.

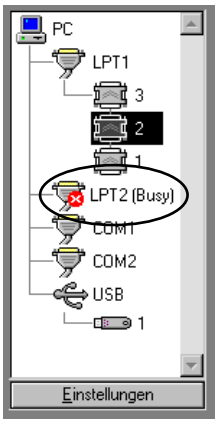
Mit der Schaltfläche »Info« erhalten Sie die Versionsnummer der im System vorhandenen Matrix-API und Treiber.

Mit der Schaltfläche »Einstellungen« kann die Art des Zugriffs auf die LPT-Ports eingestellt werden. Diese Funktion wird auf den folgenden Seiten weiter erläutert.

Mit den Schaltflächen »Dongle-Key«, »Encryption« können Sie zwischen den verschiedenen Programmfunktionen hin- und herschalten. Diese werden ebenfalls auf den Folgeseiten näher beschrieben.

Zugriff auf den LPT-Port

Der Matrix-Treiber belegt die LPT-Schnittstelle nur, sofern diese frei ist. Wird die Schnittstelle gerade von anderen Geräten – z. B. Drucker, Scanner usw. – belegt, so wartet der Matrix-Treiber auf die Freigabe der Schnittstelle. Als Standard ist die Wartezeit auf 10 Sekunden eingestellt.



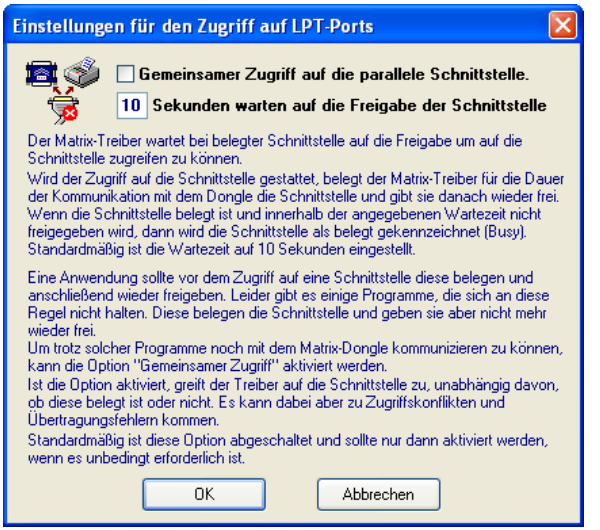
Wird die Schnittstelle innerhalb der Wartezeit frei, dann belegt der Matrix-Treiber für die Dauer der Kommunikation mit dem Dongle die Schnittstelle und gibt Sie danach wieder frei.

Sofern die Schnittstelle innerhalb der Wartezeit nicht frei wird, wird diese als belegt (»Busy«) gekennzeichnet.

Vor dem Zugriff auf eine Schnittstelle sollte eine Anwendung immer diese erst belegen und anschließend wieder freigeben. Leider gibt es einige Programme, die sich an diese Regel nicht halten und die Schnittstelle belegen, dann aber nicht mehr freigeben. Um trotz solcher externen Programm-Hürden mit dem Dongle kommunizieren zu können, muss der Treiber die Schnittstelle ansprechen können, selbst wenn diese belegt ist – dies versteht man dann als »gemeinsamen Zugriff«.

Diese Art von Zugriff auf die Schnittstelle kann aber zu Zugriffskonflikten und Übertragungsstörungen führen und ist aus diesem Grund als Standard ausgeschaltet.

Mit der Schaltfläche »Einstellungen« kann diese Option eingeschaltet werden, sollte aber nur dann aktiviert werden, wenn es unbedingt erforderlich ist.
Wenn die Option »Gemeinsamer Zugriff« ausgeschaltet ist – also belegt der Treiber ordnungsgemäß die Schnittstelle nur dann, wenn diese auch frei ist –, kann hier die Wartezeit in Sekunden eingestellt werden.



Diese Einstellungen werden in der Datei ***matrix.ini*** im Verzeichnis ***\windows\system*** gespeichert.
Diese Datei können Sie ebenfalls mit Ihrer geschützten Anwendung ausliefern, damit Sie gegebenenfals auch bei Ihren Kunden diese Einstellungen verändern können.

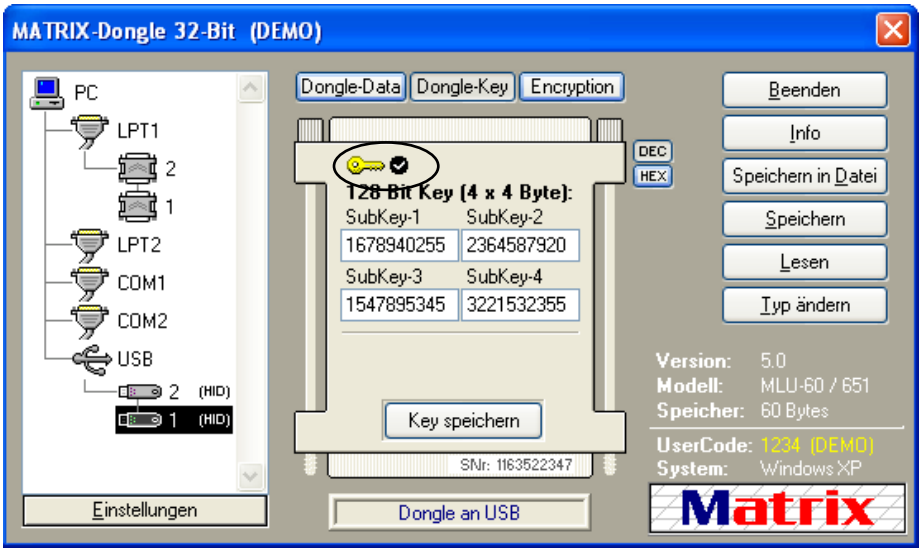
Die Datei ***matrix.ini*** hat folgenden Aufbau:

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

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

Die Einstellungen AccessLPT=ON und AccessUSB=ON können auf OFF gesetzt werden, um die Unterstützung des jeweiligen Anschlusses zu deaktivieren. Bei manchen PCs oder Laptops ohne LPT-Anschluss kann es hilfreich sein, die LPT-Unterstützung zu deaktivieren.

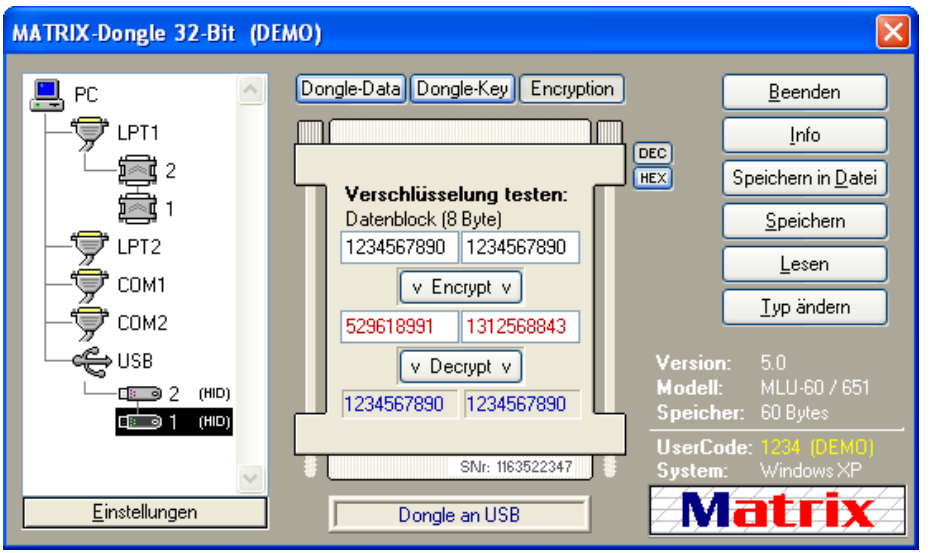
Mit der Funktion ***Dongle-Key*** kann ein aus 4x4 Bytes (= 128 Bit) bestehender Codierschlüssel für die Encrypt-/Decrypt-Funktionalität des Dongle gespeichert werden. Ein einmal gespeicherter Schlüssel wird aus Sicherheitsgründen nicht mehr angezeigt.



Wenn bereits ein Schlüssel ungleich Null gespeichert wurde, erscheint oberhalb der Eingabefelder das gelbe Schlüsselsymbol.
Schlüssel werden gelöscht, wenn bei leeren Eingabefeldern der Button »Key speichern« betätigt wird. Ein so genannter »Zero-Key« (alle Schlüssel-Bytes = 0), wird also als nicht vorhandener Schlüssel interpretiert. Man könnte die Ver-/Entschlüsselung auch mit einem »Zero-Key« durchführen, aber dies bietet keine besondere Sicherheit.

Hinweis: Die Speicherung des Schlüssels erfolgt im jeweiligen Dongle und kann aus diesem nicht mehr ausgelesen werden.

Mit der Funktion ***Encryption*** kann die Encrypt-/Decrypt-Funktionalität der Dongle getestet werden. Zur Prüfung kann man die XTEA-Verschlüsselungs-Referenztabelle weiter hinten in diesem Handbuch verwenden.



Nach Eingabe des 8-Byte Datenblocks (Klartext) betätigt man den Button »Encrypt« und erhält das Verschlüsselungs-Ergebnis in den beiden mittleren Anzeigefeldern. Nach Betätigung des Buttons »Decrypt« müssen dann die Ausgangsdaten (Klartext) wieder in den beiden unteren Datenfeldern erscheinen.

Diese Funktion können Sie z. B. benutzen, um Konstanten und Programmparameter zu verschlüsseln. Die so verschlüsselten Werte können Sie dann in Ihr Programm einsetzen.
Erst zur Programmlaufzeit werden die verschlüsselten Werte entschlüsselt, um diese für Berechnungen oder sonstige Zwecke zu verwenden.

Damit Ihr Programm also noch sinnvolle Arbeit leistet, muss der Dongle zur Laufzeit immer an den Computer angeschlossen sein, denn nur durch den Dongle lassen sich die Werte korrekt entschlüsseln. Eine Raubkopie ohne Dongle wäre also unbrauchbar.

Hinweis: Die USB-Modelle unterstützen grundsätzlich die beschriebene Encrypt-/Decrypt-Funktionalität. Bei den LPT-Modellen ausser ML-12/MK-12 wurde die Crypt-/Decryptfunktionalität ab der Hardwareversion 2.1 implementiert.

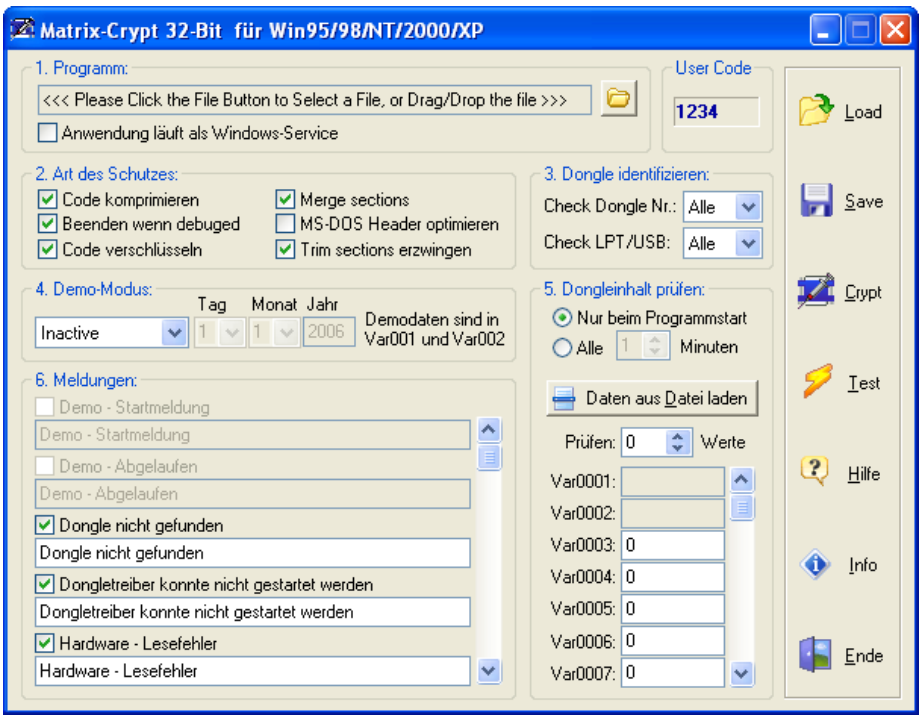
Automatische Einbindung des Schutzes ohne Änderungen im Source-Code

Allgemeine Beschreibung

Matrix-Crypt für Win32 ist ein professionelles Tool zum Schutz Ihrer Programme. Mit Matrix-Crypt können Sie einfach und schnell den Matrix-Schutz in Ihre Windows 95/98/NT/2000/XP/Vista 32 Bit EXE- und DLL-Programme integrieren. Für die Integration des Schutzes benötigen Sie keine Änderungen in dem Source-Code Ihres Programms. Alles erfolgt automatisch und so können auch Personen ohne Programmier-Erfahrung jedes Programm schützen.

Mit ***Matrix-Crypt*** können Sie:

- Die Dongleabfrage in Ihrer Anwendung integrieren.
- Ihre Software schützen vor dem so genannten »Reverse Engineering« und Debug.
- »Anti Memory Dump« Schutz in Ihre Anwendung integrieren.
- Ihre Anwendung mit »Two Secure Encryption Layers« verschlüsseln.
- Demo-Programme mit begrenzter Anzahl von Starts oder Tagen erstellen.
- Programme mit unbegrenzter Anzahl von Starts erstellen.



Um ein Programm zu schützen, muss erst eine EXE-Datei mit der Schaltfläche »File« ausgewählt werden.
Falls Ihre EXE ein Windows-Service ist, dann müssen Sie die Option »Anwendung läuft als Windows-Service« einschalten, damit diese auch nach der Verschlüsselung korrekt vom Windows Service-Manager behandelt werden kann.
Zur Erleichterung der Arbeit mit Matrix-Crypt können Sie mit der Schaltfläche »Save« den gesamten Arbeitsbereich mit den gewählten Einstellungen in ein sogenanntes »Projekt« speichern.

Sollten Sie dann zu einem späteren Zeitpunkt die erneute Verschlüsselung eines Programms wünschen, dann können Sie das jeweilige Projekt mit der Schaltfläche »Load« laden. Diese Funktion ist besonders hilfreich z. B. in der Massenproduktion von Demoversionen oder von Programmen mit modularem Aufbau, bei denen die Freischaltung von Programm-Modulen über den Dongle gesteuert wird.

Grundeinstellungen

Folgende Grundeinstellungen stehen Ihnen in Matrix-Crypt zur Verfügung:

»Art des Schutzes«

Wählen Sie in diesem Abschnitt aus, wie Sie Ihr Programm gegen »Reverse Engineering« schützen möchten:

Code komprimieren
Um das gesamte Code-Segment Ihres Programms zu komprimieren. Durch Verwendung dieser Option wird die Dateigröße Ihrer verschlüsselten Anwendung verringert.

Beenden wenn debugged
Um Ihr Programm gegen Debuggen (z. B. mit »SoftIce«) zu schützen. Das Programm schließt sich automatisch, wenn vesucht wird, dieses mit dem Debugger zu analysieren. Um höchste Sicherheit für Ihre Anwendung zu gewährleisten, ist diese Option immer aktiv und kann nicht ausgeschaltet werden.

Code verschlüsseln
Damit wird der gesamte Code-Segment Ihres Programms verschlüsselt.

Merge sections
Wenn Ihre Anwendung virtuelle Sektionen beinhaltet, dann werden diese mit der vorherigen Sektion verbunden (merged). Durch Verwendung dieser Option wird auch die Dateigröße Ihrer verschlüsselten Anwendung verringert.

MS-DOS Header optimieren

Damit werden Optimierungen im Header Ihrer Anwendung vorgenommen. Durch Verwendung dieser Option wird auch die Dateigröße Ihrer verschlüsselten Anwendung verringert.

Trim sections erzwingen

In den meisten Fällen, fügen Compiler nicht verwendete Bytes ans Ende der EXE/DLL Sektionen (tail, padding Bytes). Diese Option entfernt unbenutzte Bytes am Ende der Sektionen. Damit werden eine bessere Kompressionsrate und die Verringerung der Dateigröße Ihrer verschlüsselten Anwendung erzielt.

Hinweis: Wir empfehlen Ihnen mit einem Minimum an Schutzoptionen anzufangen. Wenn alles gut funktioniert, versuchen Sie dann eine weitere Schutzstufe zu verwenden, durch Aktivierung einer zusätzlichen Option. Jede zusätzliche Option fügt weitere Schutzmechanismen ein und erhöht die Sicherheit Ihrer Anwendung.

»Dongle identifizieren«

Check Dongle-Nr.

Wenn mehrere Dongles hintereinander gesteckt sind, kann – sofern gewünscht – die Dongle-Nummer explizit angegeben werden. Mit der Einstellung »Alle« wird der Dongle automatisch vom Programm in dem gesamten Dongle-Stapel (falls mehrere) gesucht.

Check LPT/USB

Hiermit können Sie angeben, an welchem LPT/USB-Port Ihr Dongle eingesteckt ist. Mit der Einstellung »Alle« sucht Ihr Programm automatisch den Dongle an allen vorhandenen LPT/USB-Ports.

»Dongleinhalt prüfen«

Nur beim Programmstart

Sofern Sie diese Einstellung wählen, wird die Dongle-Prüfung nur einmalig beim Programmstart durchgeführt.

Alle ___ Minuten
Für einen höheren Schutz kann hier ein Zeitintervall für die Prüfung des Dongles eingetragen werden. Dann führt Ihre verschlüsselte Anwendung die Dongle-Prüfung alle »n« Minuten aus. Die Zeit-Prüfung kann mit DLL-Dateien nicht verwendet werden.

Prüfen___Werte
Zusätzlich kann Ihre geschützte Anwendung auch den Inhalt der Variablen ab Var0003 im Dongle prüfen. Tragen Sie hier die Anzahl der Variablen ein, die geprüft werden sollen. Ohne Angabe von Vergleichsvariablen in *»Prüfen___Werte«* wird lediglich der *UserCode*, also nur die Existenz des Dongles geprüft.
Die Variablen Var0001 und Var0002 sind für die Erstellung von Demo-Programmen mit begrenzter Laufzeit reserviert und werden im folgenden Abschnitt beschrieben.

»Meldungen«

In den Feldern dieses Abschnitts können Sie Ihre eigenen Meldungen hinterlegen. Damit erzeugen Sie maßgeschneiderte Interaktions-Texte.

Geschütztes Programm mit beschränkter Laufzeit erstellen (Demo)

Für die Erstellung Ihrer mit dem Matrix-Dongle geschützten Demo-Programme stehen Ihnen mehrere Einstellungen zur Verfügung. Für die Demo-Modi wurden die Variablen Var0001 und Var0002 im Dongle reserviert. Wählen Sie im Abschnitt »Demo-Modus« eine der folgenden Demo-Arten aus:

Number of Runs

Damit legen Sie die Anzahl der möglichen Programmstarts fest. Bei der Anfertigung Ihres Dongles müssen Sie die erlaubte Anzahl an Starts mit dem Matrix-Verwaltungsprogramm in die Variable Var0001 des Dongles speichern. Nach jedem Start des geschützten Programms wird die Anzahl der möglichen Starts um **1** vermindert und in den Dongle-Speicher zurückgeschrieben. Erreicht der Variablen-Inhalt den Wert **0**, ist die Demo-Laufzeit erschöpft und das Programm lässt sich nicht mehr starten (dafür dann den passenden Interaktions-Text festlegen).

Number of Days

Diese Einstellung kann genutzt werden, um die Laufzeit Ihres Demo-Programms auf eine bestimmte Anzahl von Tagen zu begrenzen. Bei der Anfertigung Ihres Dongles müssen Sie mit dem Matrix-Verwaltungsprogramm in die Variable Var0001 des Dongles die erlaubte Anzahl der Tage speichern. Die Variable Var0002 muss den Wert 0 enthalten. Beim ersten Start des Demo-Programms sorgt diese Verschlüsselungsart automatisch für die Speicherung des Tagesdatums im Dongle in der Variable Var0002. Dieses Datum markiert dann den Beginn der Demozeit beim Programm-Anwender.
Die Anzahl der Tage in Var0001 wird automatisch um **1** vermindert und in den Dongle-Speicher zurückgeschrieben. Erreicht der Inhalt von Var0001 den Wert **0**, dann ist die Demo-Laufzeit erschöpft und das Programm lässt sich nicht mehr starten.

Valid thru

Damit wird die Laufzeit des Programms bis zu einem bestimmten Datum begrenzt. Das Ablaufdatum können Sie in den Feldern »Tag«, »Monat«, »Jahr« angeben. Bei der Anfertigung des Dongles müssen Sie mit dem Matrix-Verwaltungsprogramm in den Variablen Var0001 und Var0002 des Dongles den Wert **0** speichern.

Achtung: Wir empfehlen Ihnen, die Demo-Variante »Valid thru« nicht zu verwenden, weil diese keinen ausreichenden Schutz gegen die Manipulation des Systemdatums bietet. Verwenden Sie – soweit möglich – die Demo-Variante »Number of Runs«.

Hinweis: Auch mit dem Demo-Modus »Number of Runs« können Sie Ihre Anwendung auf eine bestimmte Anzahl von Tagen eingrenzen.

Im folgenden Beispiel wird dargestellt, wie »Number of Runs« für die Begrenzung der Tage eingesetzt werden kann:

- nehmen wir an, dass der Anwender Ihr Programm im Durchschnitt 5 Mal pro Tag startet.

- wenn Sie die Laufzeit auf 30 Tage begrenzen möchten, dann wird die Anwendung insgesamt 5 x 30 = 150 Mal gestartet in diesem Zeitraum.

- verwenden Sie die Option »Number of Runs« mit 150 Starts und die Anwendung wird dann in etwa den gewünschten Zeitraum lauffähig sein. Sie können dabei auch etwas Sicherheit einkalkulieren und 160 statt 150 Starts zulassen.

Inactive
In diesem Modus werden Programme verschlüsselt, die nicht für Demo-Zwecke bestimmt sind. Weitere Infos dazu in *»Geschütztes Programm mit unbeschränkter Laufzeit erstellen (kein Demo)«*.

Geschütztes Programm mit unbeschränkter Laufzeit erstellen (kein Demo)

Für die Erstellung Ihrer mit dem Matrix-Dongle geschützten Programme, die unbeschränkt funktionsfähig sind (kein Demoprogramm), setzen Sie einfach den Demo-Modus auf »Inaktive«.

Zusätzlich kann beim Start Ihres Demo-Programms auch der Inhalt der Variablen ab Var0003 im Dongle geprüft werden. Ohne Angabe von Vergleichsvariablen wird lediglich der UserCode geprüft. Ist die Existenz eines Dongles mit korrektem UserCode bestätigt, kann das geschützte Programm gestartet werden.

Nachdem Sie die gewünschten Einstelungen gewählt haben, klicken Sie anschließend auf die Schaltfläche »Crypt«, um die Dongle-Abfrage in Ihr Programm zu integrieren. Beim Verschlüsseln Ihres Programms wird immer die original ungeschützte EXE/DLL als Sicherung (Backup) mit dem Namen *programname.exe.old* abgelegt.

Achtung: Sie sollen Ihr geschütztes Programm vor der Auslieferung immer gründlich testen. Für eine einfache Handhabung steht Ihnen die Schaltfläche »Test« zur Verfügung, mit der Sie das geschützte Programm starten können.

Matrix-Crypt von der Command-Line verwenden

Um die Verschlüsselung der EXE Dateien in Ihrer eigenen Entwicklungsumgebung zu integrieren, können Sie Matrix-Crypt als Command-Line Tool verwenden:

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

-p *ProjectFile.mcp*
Mit diesem Parameter wird angegeben welche Mx-Crypt Projekt-Datei (.mcp) für die Verschlüsselung verwendet werden soll.
Wichtig: Die Projekt-Datei muss mit vollem Pfad (full path) angegeben werden.

-e *File.exe*
Dieser Parameter ist optional. Durch Verwendung diesen Parameters, wird angegeben welche EXE-Datei verschlüsselt werden soll, anstelle der EXE aus der Projekt-Datei.
Wichtig: Die EXE-Datei muss mit vollem Pfad (full path) angegeben werden.

-o *OutMessageFile*
Dieser Parameter ist optional. Durch Verwendung diesen Parameters, werden alle Bildschirm-Meldungen in die angegebene Datei umgeleitet.

Manuelle Einbindung des Schutzes im eigenen Source-Code

Methoden der Einbindung

Zur Einbindung des Dongle in Ihre Anwendung stehen Ihnen mehrere Funktionen zur Verfügung; zwei Methoden bieten sich da an:

Static
Diese Methode kann in Programmen eingesetzt werden, in denen eine statische Bibliothek (LIB) gelinkt werden kann (z. B. VC++). Für die statische Methode befinden sich die API-Funktionen in den 16 Bit und 32 Bit-LIBs.

Dynamic
Diese Methode kann in den Programmen eingesetzt werden, die eine DLL aufrufen können (z. B. VC++, Delphi, VisualBasic, Access etc.). Für die dynamische Methode befinden sich die API-Funktionen in den 16 Bit und 32 Bit DLLs.

Folgende Dateien sind für die Einbindung verfügbar:

Dynamic link			**Static link**		
16-bit	***32-bit***	***64-bit***	***16-bit***	***32-bit***	***64-bit***
matrix16.h	matrix32_64.h	matrix32_64.h	mxst16.h	matrix32_64.h	matrix32_64.h
matrix16.lib	matrix32.lib	matrix64.lib	mxst16.lib	mxst32.lib	mxst64_vc80.lib
matrix16.dll	matrix32.dll	matrix64.dll			

Die folgenden Beispiele demonstrieren Ihnen, wie Sie die API-Funktionen in den bekanntesten Programmiersprachen (C/C++, Visual Basic, Pascal/Delphi) integrieren können. Die primären Aufgaben dieser Beispiele sind eine gute Lesbarkeit und Verständlichkeit; nicht aber die höchsten Schutzmaßnahmen zu erzielen.
Bitte beachten Sie dies bei der Implementierung in Ihre Anwendung und machen Sie auch Gebrauch von den Schutzmaßnahmen, beschrieben in den Kapiteln »Kryptographische Authentifizierung des Matrix-Dongles« und »Richtlinien für einen guten Software-Schutz«.
Im Kapitel »API-Funktionen Referenz« finden Sie eine detaillierte Beschreibung aller verfügbaren API-Funktionen.

Mehrere Beispiele für verschiedene Programmiersprachen finden Sie auch im Verzeichnis *\matrix\samples*.

Beispiel für die Einbindung in C/C++

```
#include "matrix32.h"

long  DataIn[256];          /* Buffer zum Lesen der Dongle-Daten        */
long  DataOut[256];         /* Buffer für Daten zum Speichern           */
long  DataCrypt[2];         /* Buffer für Daten zum Verschlüsseln       */
short RetCode;              /* Rückgabewert                             */
long  DNG_Version;          /* Dongle-Versionsnummer          */
long  DNG_SerNr;/* Eindeutige Serien-Nr. des Dongles*/
short DNG_Port;             /* LPT/USB-Port                             */
short DNG_LPTADR;           /* Adresse des LPT-Anschlusses              */
short DNG_Count;            /* Anzahl der Dongles an dem LPT-Port       */
short DNG_Mem;              /* Speichergröße des Dongles                */
short DNG_MaxVar;           /* Maximale Anzahl der Daten-Felder         */
short i;

        /*-------------------------------------------------------------------- */
        /* Anschluss (LPT oder USB) für folgende Aufrufe.                     */
        /* Mögliche Werte für DNG_Port sind:                                  */
        /* 1-3 = LPT1-LPT3, ‚U‘ oder 85 = USB                                 */
        /*-------------------------------------------------------------------- */
        DNG_Port = 1;   /* 85 = USB */

        /*--------------------------------------------------------------------*/
        /* Init Matrix-API.                                                   */
        /*-------------------------------------------------------------------- */
        RetCode = Init_MatrixAPI();
        if(RetCode < 0)
        {
                printf("Init_MatrixAPI Return-Code: %d", RetCode);
        }
        /*-------------------------------------------------------------------- */
        /* Prüfen ob LPT1 vorhanden ist.                                      */
        /*-------------------------------------------------------------------- */
        DNG_LPTADR = GetPortAdr(1);  /* 1 = LPT1 */
        if(DNG_LPTADR == 0)
        {
                printf("Der Anschluss LPT1 ist nicht vorhanden!");
                Release_MatrixAPI();
                exit;
        }
```

... Fortsetzung Beispiel C/C++

```
/*---------------------------------------------------------------------- */
/* Die Anzahl der Dongles an LPT/USB ermitteln.                          */
/*---------------------------------------------------------------------- */
DNG_Count = Dongle_Count(DNG_Port);
if(DNG_Count == 0)
{
        printf("Matrix-Dongle an LPT/USB nicht gefunden!");
        Release_MatrixAPI();
        exit;
}

/*----------------------------------------------------------------------*/
/* Speichergröße des letzten Dongles an LPT/USB lesen.                  */
/*----------------------------------------------------------------------*/
DNG_Mem = Dongle_MemSize(DNG_Count, DNG_Port);
if(DNG_Mem == 0)
{
        printf("Speichergröße konnte nicht gelesen werden!");
        Release_MatrixAPI();
        exit;
}
DNG_MaxVar = DNG_Mem / 4;    /* Anzahl der Datenfelder    */

/*----------------------------------------------------------------------*/
/* Dongleversion des letzten Dongle an LPT/USB lesen.                   */
/*----------------------------------------------------------------------*/
DNG_Version = Dongle_Version(DNG_Count, DNG_Port);
if(DNG_Version <= 0)
{
        printf("Versionsnummer konnte nicht gelesen werden!");
        Release_MatrixAPI();
        exit;
}
printf("Version-Nr. des Dongles: %d.%d",        HIWORD(DNG_Version),
                                                LOWORD(DNG_Version));
```

... Fortsetzung Beispiel C/C++

```
/*---------------------------------------------------------------------- */
/* Die eindeutige Serien-Nummer des letzten Dongle an                    */
/* LPT/USB mit dem UserCode 1234 lesen und anzeigen.                     */
/*---------------------------------------------------------------------- */
DNG_SerNr = Dongle_ReadSerNr(1234, DNG_Count, DNG_Port);
if(DNG_SerNr < 0)
{
        printf("Fehler beim Lesen der Serien-Nummer!");
        Release_MatrixAPI();  exit;
}
printf("Dieser Dongle hat die Serien-Nr.: = %ld", DNG_SerNr);

/*---------------------------------------------------------------------- */
/* 15 Datenfelder aus dem letzten Dongle mit dem                         */
/* UserCode 1234 an LPT/USB lesen und anzeigen.                          */
/*----------------------------------------------------------------------*/
RetCode = Dongle_ReadData(1234,   DataIn,   15,        DNG_Count,
                                                       DNG_Port);
if(RetCode < 0)
{
        printf("Fehler beim Lesen der Daten!");
        Release_MatrixAPI();  exit;
}
for(i=0; i<15; i++)
{
        printf("Wert %d = %ld", i+1, DataIn[i]);
}

/*---------------------------------------------------------------------- */
/* 15 Daten im letzten Dongle mit dem UserCode 1234                      */
/* an LPT/USB speichern. Als Beispiel werden die Werte                   */
/* 101,102,...115 gespeichert.                                           */
/*----------------------------------------------------------------------*/
for(i=0; i<15; i++)
{
        DataOut[i] = 101 + i;
}
RetCode = Dongle_WriteData(1234,   DataOut, 15,        DNG_Count,
                                                       DNG_Port);
if(RetCode < 0)
{
        printf("Fehler beim Schreiben der Daten!");
        Release_MatrixAPI();  exit;
}
```

... Fortsetzung Beispiel C/C++

```
/*---------------------------------------------------------------------- */
/* Daten über den Dongle verschlüsseln und das                           */
/* verschlüsselte Ergebnis anzeigen.                                     */
/* Als Beispiel werden die Daten '1234567890 1234567890'                 */
/* verschlüsselt.                                                        */
/* Die Verschlüsselung erfolgt im Dongle mit dem Key,                    */
/* der im Dongle gespeichert ist.                                        */
/*----------------------------------------------------------------------*/
DataCrypt[0] = 1234567890;
DataCrypt[1] = 1234567890;
RetCode = Dongle_EncryptData(1234, DataCrypt, DNG_Count,
                                                DNG_Port);
if(RetCode < 0)
{
        printf("Fehler beim Verschlüsseln der Daten!");
        Release_MatrixAPI();
        exit;
}
printf("Verschlüsselte Daten = %lu : %lu",      DataCrypt[0],
                                                DataCrypt[1]);

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

Beispiel für die Einbindung in 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
```

... Fortsetzung
Beispiel Visual Basic

```
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
```

... Fortsetzung Beispiel Visual Basic

```
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

'*****************************************************************  *
'* Anschluss (LPT oder USB) für folgende Aufrufe festlegen.          *
'* Mögliche Werte für DNG_Port sind:                                 *
'* 1-3 = LPT1-LPT3, 85 (ASCII 'U') = USB                             *
'*******************************************************************
DNG_Port = 1    '*** 85 = USB ***

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

'*****************************************************************  *
'* Prüfen ob der Port LPT1 vorhanden ist.                            *
'*****************************************************************  *
DNG_LPTADR = GetPortAdr(1)    '*** 1 = LPT1 ***

If DNG_LPTADR = 0 Then
        MsgBox "LPT1 ist nicht vorhanden!"
End If
MsgBox "Die Adresse von LPT1 lautet: " & Hex(DNG_LPTADR)
```

... Fortsetzung
Beispiel Visual Basic

```
'****************************************************************  *
'* Anzahl der Dongles an LPT/USB suchen.                         *
'****************************************************************  *
DNG_Count = Dongle_Count(DNG_Port)

If DNG_Count = 0 Then
        MsgBox "Kein Dongle vorhanden an Port: " & DNG_Port
End If
MsgBox "Gefunden: " & DNG_Count & " Dongles an Port: " _
                                & DNG_Port

DNG_Nr = DNG_Count

'****************************************************************  *
'* Speichergröße des letzten Dongles an LPT/USB lesen und        *
'* die maximale Anzahl der Datenfelder berechnen.                *
'****************************************************************  *
DNG_Mem = Dongle_MemSize(DNG_Nr, DNG_Port)

If DNG_Mem = 0 Then
        MsgBox "Fehler beim ermitteln des Dongle-Speichers!"
End If
DNG_MaxVar = DNG_Mem / 4
MsgBox "Anzahl der Variablen dieses Dongles: " & DNG_MaxVar

'****************************************************************  *
'* Dongleversion aus dem letzten Dongle an LPT/USB lesen.        *
'****************************************************************  *
DNG_Version = Dongle_Version(DNG_Nr, DNG_Port)

If DNG_Version <= 0 Then
        MsgBox "Fehler beim Lesen der Dongle-Version!"
Else
        VerMinor = CInt(DNG_Version And 65535)
        VerMajor = CInt(DNG_Version \ Shift16)
        MsgBox "Dieser Dongle hat die Versions-Nr:  " &      VerMajor      _
                                                    &      "." VerMinor
End If
```

... Fortsetzung Beispiel Visual Basic

```
'*****************************************************************  *
'* Eindeutige Serien-Nummer des letzten Dongle an LPT/USB          *
'* mit dem UserCode 1234 lesen und anzeigen.                       *
'*****************************************************************  *
DNG_SerNr = Dongle_ReadSerNr(1234, DNG_Nr, DNG_Port)

If DNG_SerNr < 0 Then
        MsgBox "Fehler beim Lesen der Serien-Nummer!"
Else
        MsgBox "Dieser Dongle hat die Serien-Nummer: " & DNG_SerNr
End If

'*****************************************************************  *
'* 15 Datenfelder aus dem letzten Dongle an LPT1 mit dem           *
'* UserCode 1234 lesen und anzeigen.                               *
'*****************************************************************  *
RetCode = Dongle_ReadData(1234, DataIn(0), 15, DNG_Nr, _
                                                DNG_Port)

If RetCode < 0 Then
        MsgBox "Fehler beim Lesen der Dongle-Daten!"
End If
For i = 0 To 14
        MsgBox "Inhalt der Variable: " & i + 1 & ": " & DataIn(i)
Next i

'*****************************************************************  *
'* 15 Datenfelder im letzten Dongle mit dem UserCode 1234          *
'* an LPT/USB speicher. Die Werte 101,102...115 speichern.         *
'*****************************************************************  *
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 "Fehler beim Schreiben der Dongle-Daten!"
Else
        MsgBox "Die Dongle-Daten wurden erfolgreich geschrieben!"
End If
```

... Fortsetzung Beispiel Visual Basic

```
'************************************************************** *
'* Daten über den Dongle verschlüsseln und das Ergebnis        *
'* anzeigen. Verschlüsselt wird mit dem Key aus dem Dongle.    *
'************************************************************** *
DataCrypt(0) = 1234567890
DataCrypt(1) = 1234567890
RetCode = Dongle_EncryptData(1234, DataCrypt,      DNG_Nr, _
                                                   DNG_Port)

If RetCode < 0 Then
        MsgBox "Fehler beim Verschlüsseln der Daten!"
Else
        MsgBox "Verschlüsselte Daten: "    &       DataCrypt(0) & " " _
                                           &       DataCrypt(1)
End If

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

Beispiel für die Einbindung in 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

        {*********************************************************}
        {* Anschluss (LPT oder USB) für folgende Aufrufe festlegen       *}
        {* Mögliche Werte für DNG_Port sind:                             *}
        {* 1-3 = LPT1-LPT3, 85 (ASCII 'U') = USB                         *}
        {*********************************************************}
        DNG_Port := 1;   {*** 85 = USB ***}

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

        if RetCode < 0 then
        begin
```

... Fortsetzung Beispiel Pascal

```
        write('*** Init_MatrixAPI Return-Code: ');
        writeln(RetCode);
end;

{*********************************************************}
{* Prüfen ob LPT1 vorhanden ist.                         *}
{*********************************************************}
DNG_LPTADR := GetPortAdr(1);   {*** 1 = LPT ***}

if DNG_LPTADR = 0 then
begin
        writeln('*** LPT1 nicht vorhanden!');
        Release_MatrixAPI();
        exit;
end;

{*********************************************************}
{* Anzahl der Dongles an LPT/USB suchen.                 *}
{*********************************************************}
DNG_Count := Dongle_Count(DNG_Port);

if DNG_Count = 0 then
begin
        writeln('Matrix-Dongle an LPT/USB nicht gefunden!');
        Release_MatrixAPI();
        exit;
end;

DNG_Nr := DNG_Count;

{*********************************************************}
{* Speichergröße des letzten Dongles an LPT/USB lesen und *}
{* die maximale Anzahl der Datenfelder berechnen.        *}
{*********************************************************}
DNG_Mem := Dongle_MemSize(DNG_Nr, DNG_Port);

if DNG_Mem = 0 then
begin
        writeln('Fehler beim Lesen der Speichergröße!');
        Release_MatrixAPI();
        exit;
end;
```

... Fortsetzung Beispiel Pascal

```
{*********************************************************}
{* Dongleversion aus dem letzten Dongle an LPT1 lesen.    *}
{*********************************************************}
DNG_Version := Dongle_Version(DNG_Nr, DNG_Port);

if DNG_Version = 0 then
begin
        writeln('Fehler beim Lesen der Versionsnummer!');
        Release_MatrixAPI();
        exit;
end;

VerMinor := (DNG_Version and 65535);
VerMajor := (DNG_Version shr 16);
write('Dieser Dongle hat die Versions-Nummer: ');
write(VerMajor);
write('.');
writeln(VerMinor);

{*********************************************************}
{* Eindeutige Serien-Nummer des letzten Dongle an LPT/USB *}
{* mit dem UserCode 1234 lesen und anzeigen.              *}
{*********************************************************}
DNG_SerNr := Dongle_ReadSerNr(1234, DNG_Nr, DNG_Port);

if DNG_SerNr < 0 then
begin
        writeln('Fehler beim Lesen der Serien-Nummer!');
        Release_MatrixAPI();
        exit;
end;

        write('Dieser Dongle hat die Serien-Nr.: ');
        writeln(DNG_SerNr);
```

... Fortsetzung Beispiel Visual Basic

```
{*****************************************************************}
{* 15 Datenfelder aus dem letzten Dongle mit dem                 *}
{* UserCode 1234 an LPT/USB lesen und anzeigen.                  *}
{*****************************************************************}
RetCode := Dongle_ReadData(1234,  @DataIn,          15,       DNG_Nr,
                                                              DNG_Port);

if RetCode < 0 then
begin
        writeln('Fehler beim Lesen der Daten!');
        Release_MatrixAPI();
        exit;
end;

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

{*****************************************************************}
{* 15 Daten im letzten Dongle mit dem UserCode 1234 an           *}
{* LPT/USB speichern. Die Werte 101,102...115 speichern.         *}
{*****************************************************************}
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('Fehler beim Schreiben der Daten!');
        Release_MatrixAPI();
        exit;
end;

writeln('Die Dongle-Daten wurden erfolgreich geschrieben!');
```

... Fortsetzung Beispiel Pascal

```
{************************************************************}
{* Daten über den Dongle verschlüsseln und das Ergebnis     *}
{* anzeigen.                                                *}
{* Als Beispiel den Datenblock '1234567890 1234567890'      *}
{* verschlüsseln.                                           *}
{* Verschlüsselt wird mit dem Key aus dem Dongle.           *}
{************************************************************}
DataCrypt[1] := 1234567890;
DataCrypt[2] := 1234567890;

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

if RetCode < 0 then
begin
        writeln('Fehler beim Verschlüsseln der Daten!');
        Release_MatrixAPI();
        exit;
end;

write('Verschlüsselte Daten: ');
write(DataCrypt[1]); write(' : '); writeln(DataCrypt[2]);

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

Kryptographische Authentifizierung des Matrix-Dongles

Ein Hacker wird in der Regel versuchen, die Matrix-Treiber/API zu emulieren. Um solche Angriffe abzuwehren, ist eine Authentifizierung des Dongle erforderlich. Dieses Verfahren stellt sicher, dass die Rückmeldungen tatsächlich vom Dongle und nicht von einem simulierenden Treiber stammen.

Der sicheren Authentifizierung des Dongle liegt folgende Logik zugrunde:
In der zu schützenden Anwendung (EXE) wird eine 64 Bit Zufallszahl (in der Praxis zwei zusammengesetzte 32 Bit Zufallszahlen) erzeugt. Diese Zufallszahl wird dann auf zwei Wegen verschlüsselt.

1. Der erste Weg führt über die Verschlüsselungsfunktion MxApp_Encrypt innerhalb Ihrer Applikation (EXE). Die 64 Bit Zufallszahl wird mit einem 128 Bit Schlüssel verschlüsselt. Dieser Schlüssel und der im Dongle gespeicherte Schlüssel müssen gleich sein.

2. Der zweite Weg führt über den Dongle mit der Funktion Dongle_EncryptData. Damit wird die gleiche 64 Bit Zufallszahl im Dongle verschlüsselt. Diese Verschlüsselung erfolgt mit Hilfe des gleichen im Dongle vorhandenen 128 Bit Schlüssels.

Die Authentizität des Dongle wird durch einen Vergleich der beiden Verschlüsselungs-Ergebnisse festgestellt. Der Schlüssel tritt bei diesem Verfahren nicht in Erscheinung, weil er nicht übertragen wird und deshalb nicht belauscht werden kann.

Beispiel
Im folgenden Programmcode wird ein einfaches Beispiel für die Authentifizierung vorgestellt.

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

/*-------------------------------------------------------*/
/* 1. 'Klardaten' aus Zuffalszahlen erstellen            */
/*  Zwei Buffers mit gleichen 'Klardaten' initialisieren.*/
/*-------------------------------------------------------*/
Data_App[0] = Zuffalszahl_1;
Data_App[1] = Zuffalszahl_2;

Data_Dng[0] = Zuffalszahl_1;
Data_Dng[1] = Zuffalszahl_2;
```

... Fortsetzung Beispiel

```
/*-------------------------------------------------------*/
/* 2. Verschlüsselung der 'Klardaten' in der Anwendung.  */
/*-------------------------------------------------------*/
MxApp_Encrypt(Data_App, Key);

/*-------------------------------------------------------*/
/* 3. Verschlüsselung der 'Klardaten' im Dongle.         */
/*-------------------------------------------------------*/
Dongle_EncryptData(UserCode, Data_Dng, DongleNr, PortNr);

/*-------------------------------------------------------*/
/* 4. Vergleich der Verschlüsselungs-Ergebnisse.         */
/*-------------------------------------------------------*/
if(Data_App[0] == Data_Dng[0] && Data_App[1] == Data_Dng[1])
{
  //-- Success: Verschlüsselungs-Ergebnisse sind gleich --
}
```

Hinweis: Die XTEA-128 Verschlüsselungsfunktion MxApp_Encrypt muss in Ihre Anwendung integriert werden. Diese Funktion ist verfügbar für C/C++, VB und Delphi in den Dateien mxtea.h, mxtea.bas bzw. mxtea.pas aus dem Verzeichnis \matrix\samples\...

Empfehlungen

Für die Erhöhung der Sicherheit gegen Code-Angriffe können folgende Punkte berücksichtigt werden:

- Wenn Sie 64 Bit Zufallszahlen benutzen, ist gewährleistet, dass für die Datenübertragung stets andere Daten benutzt werden.
  Durch diese Methode wird verhindert, dass ein Hacker versteht, was an dieser Stelle passiert.

- Zur Vereinfachung des Beispiels wurden die 64 Bit Zufallszahlen mit festen Werten belegt. Wenn Sie diese Zahlen zur Laufzeit durch einen Algorithmus zusammensetzen, erfolgt eine zusätzliche Erhöhung der Sicherheit gegen »Reverse Engineering«.

- Sie sollten die Zufallszahlen nur für die Dauer der Verschlüsselung mit gültigem Inhalt belegen und danach diese wieder initialisieren (löschen).

- Zur Vereinfachung des Beispiels wurden die Ergebnisse durch einen einfachen Vergleich überprüft. Sie sollten aber die Überprüfung durch Differenzbildung auf Bit- und Byte-Ebene realisieren.

- Sie sollten die Vergleichsoperation in der Code-Lokalisierung von den Verschlüsselungsoperationen trennen.

- Sie sollten mehrere Sequenzen von Zufallszahlen verschlüsseln und die Vergleiche in anderer Reihenfolge (vermischt) und an anderen Stellen im Programm-Code durchführen.

XTEA - Verschlüsselungsreferenz

Nachfolgend finden Sie fünf 8-Byte lange Datenblöcke, die mit dem 128-Bit Beispiel-Schlüssel verschlüsselt wurden.
Einige Programmiersprachen, wie z. B. Visual Basic, unterstützen keine 32-Bit *unsigned integers* (4 Byte vorzeichenlos). Diese werden aber für Klardaten und verschlüsselte Datenblöcke verwendet. In diesem Fall werden automatisch *unsigned integers* in *signed integers* konvertiert; somit werden große vorzeichenlose Werte als negative Zahlen dargestellt.

Ein Beispiel:
Ein *unsigned byte* mit dem Wert 128 konvertiert zum *signed byte* ist -128
Ein *unsigned byte* mit dem Wert 129 konvertiert zum *signed byte* ist -127

Das ist mathematisch korrekt, denn der Wertebereich für *unsigned byte* ist von 0 bis 255 und der Wertebereich für *signed byte* ist von -128 bis +127.
Aber das kann in der Praxis auch zu Verwirrungen führen.

Für solche Programmiersprachen empfehlen wir die hexadezimale Darstellung der Werte, um Konfusionen zwischen *signed* und *unsigned* zu vermeiden.

Der hexadezimale Wertebereich ist unabhängig von dem Variablentyp (*signed* oder *unsigned*). Für ein Byte ist er immer von 0x00 bis 0xFF.
Als Hilfe für die Implementierung der Verschlüsselung in Ihre Software ist folgende Verschlüsselungsreferenz in beiden Formaten gelistet:
Unsigned Decimal und *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	1234567890	4264218190	2765200066	(dec)
	499602D2	499602D2	FE2ACE4E	A4D19AC2	(hex)
2.	2233445566	1122445577	0128004498	2776431824	(dec)
	851FACBE	42E72909	07A13192	A57CFCD0	(hex)
3.	1468205773	0934113682	2273514099	1600112827	(dec)
	57830ACD	37AD7192	87831273	5F5FC8BB	(hex)
4.	3942612857	2557492693	3416279985	0347058491	(dec)
	EAFF7F79	98703DD5	CBA04BB1	14AFB13B	(hex)
5.	2670314019	1725837151	1447417338	0381438979	(dec)
	9F29C223	66DE2F5F	5645D5FA	16BC4C03	(hex)

Netzwerk Lizenz-Management

Verwaltung von Netzwerklizenzen

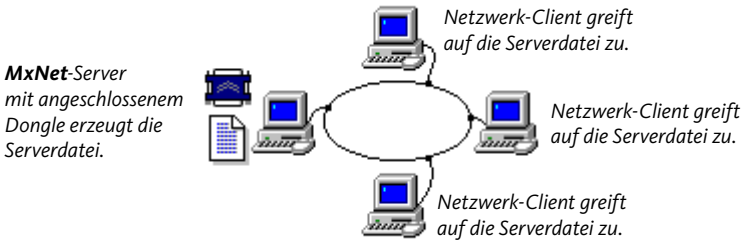
Bei dem Schutz von Software in einem Netzwerk ist es möglich, eine bestimmte Anzahl von Lizenzen zu definieren, welche vom Anwender gleichzeitig benutzt werden können. Diese Funktionalität, *»Lizenz-Management«* genannt, kann auf zwei verschiedene Arten realisiert werden:

1. Der Einsatz eines Dongle an jedem Arbeitsplatz
2. Der Einsatz eines einzigen Dongle für das gesamte Netzwerk

Die erste Variante – mit je einem Dongle pro Arbeitsplatz – unterscheidet sich nicht von der Schutztechnik eines Einzelplatz-Programms, da die Anwendung immer einen lokal angeschlossenen Dongle verwendet. Aus diesem Grund wird hier diese Methode nicht näher erläutert.

Die zweite Variante – mit einem einzigen Dongle für das gesamte Netzwerk – basiert dagegen auf einem etwas anderen Konzept als die Einzelplatz-Lösung. Diese Methode, *»MxNet«* genannt, bietet die Möglichkeit, im Netzwerk einen einzigen Dongle zu verwenden, der an jedem beliebigen Arbeitsplatz angeschlossen werden kann. Die Verwaltung der Lizenzen erfolgt über eine sogenannte *»Server-Datei«*, die von dem *MxNet*-Serverprogramm erzeugt wird. Alle im Netzwerk vorhandenen Clients können über die Matrix-API auf die Serverdatei wie auf einen lokalen Dongle zugreifen.

Mit dem folgenden Schema wird die Funktionsweise im Netzwerk mit der *MxNet*-Technik erläutert:



Der Netzwerkschutz über MxNet verwendet keine Netzwerk-Protokolle und kann somit in jedem beliebigen Netzwerk-System eingesetzt werden. Auf dem MxNet-Server, also der PC mit angeschlossenem Dongle, läuft das Server-Programm MxNet, von dem die Server-Datei erzeugt wird. Diese Datei wird in vordefinierten Zeitabständen aktualisiert

und in verschlüsselter Form abgelegt.
Der Verschlüsselungs-Algorithmus verändert sich jedes Mal, sobald die Datei aktualisiert wird; dies bietet höchste Sicherheit gegen unerwünschte Manipulation.

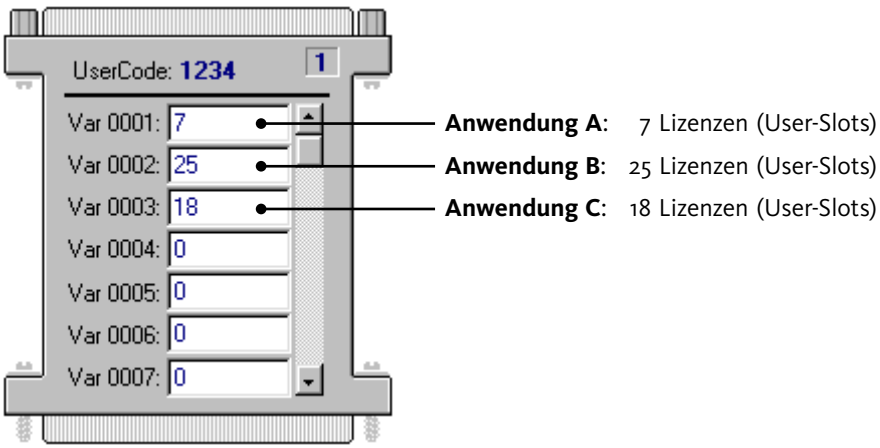
Die Verwaltung der gleichzeitig laufenden Kopien (Lizenzen) der geschützten Anwendung im Netzwerk erfolgt über sogenannte *»User-Slots«*. Ein User-Slot ist ein Eintrag in der Server-Datei, der von jedem Netzwerk-Client vorgenommen wird. Der User-Slot wird von jedem Client solange belegt, wie die geschützte Anwendung läuft und anschließend beim Beenden der Anwendung wieder freigegeben.

Die Anzahl der zulässigen Lizenzen pro Anwendung wird in den Matrix-Dongle gespeichert. Er stellt diese Information dem *MxNet*-Serverprogramm zur Verfügung.

Die Verwaltung der Lizenzen mit dem Matrix-Dongle bietet höchste Flexibilität und ermöglicht für mehrere oder modular aufgebaute Anwendungen problemlos den Schutz mit nur einem Dongle.
Im folgenden Abschnitt wird die Verwaltung der Lizenzen für drei verschiedene Programme mit nur einem Dongle erläutert.

Verwaltung von Netzwerklizenzen im Dongle

Für jede Anwendung wird im Dongle jeweils eine Variable verwendet. In diesen Variablen wird also die Anzahl der Lizenzen für jede Anwendung gespeichert. In der folgenden Abbildung sind 7 Lizenzen für die Anwendung A, 25 für Anwendung B und 18 für Anwendung C dargestellt.



Diese anwendungsspezifischen Variablen, »Application-Slots« genannt, beinhalten also die Anzahl der möglichen User-Slots, die in der Server-Datei verwaltet werden. In unserem Beispiel können also in die Server-Datei von den geschützten Anwendungen maximal 7 User zum Application-Slot der Anwendung A (Var0001), 25 User zum Application-Slot der Anwendung B (Var0002) und 18 User zum Application-Slot der Anwendung C (Var0003) angemeldet werden.

Die Maximale Anzahl der User-Slots pro Anwendung ist 32500. Sollte der Application-Slot (Eintrag im Dongle) größer 32500 sein, dann wird diese Anzahl in der Server-Datei automatisch auf 32500 reduziert.

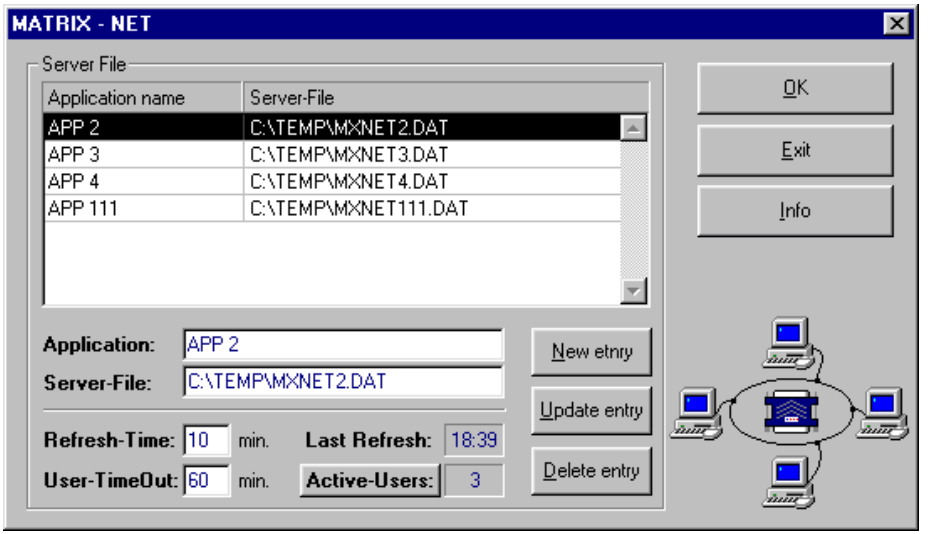
Einstellungen des MxNET Server-Programms

Das Programm *Matrix-NET* ist die *MxNet* Server-Anwendung und muss auf dem *MxNet* Server-PC (Computer mit angeschlossenem Dongle) laufen. Diese Anwendung erzeugt und aktualisiert die *MxNet* Server-Datei.

Matrix-NET

Beim Start des *Matrix-NET* Programms wird in der Task-Leiste ein Dongle-Symbol angezeigt. Mit einem Klick auf dieses Icon kann das *Matrix-NET* Dialogfenster aktiviert werden.

Die Einstellungsmöglichkeiten werden im folgenden Abschnitt beschrieben.



Falls erforderlich, ist es möglich, für jede Applikation eine eigene Server-Datei zu erzeugen. Die Eintragung der Server-Datei erfolgt über die Eingabefelder »Application« und »Server-File«. Jeder Eintrag muss aus »Name der Anwendung« und »Name der Server-Datei« bestehen. Die Server-Datei muss mit absolutem Pfad eingetragen werden.

Refresh-Time

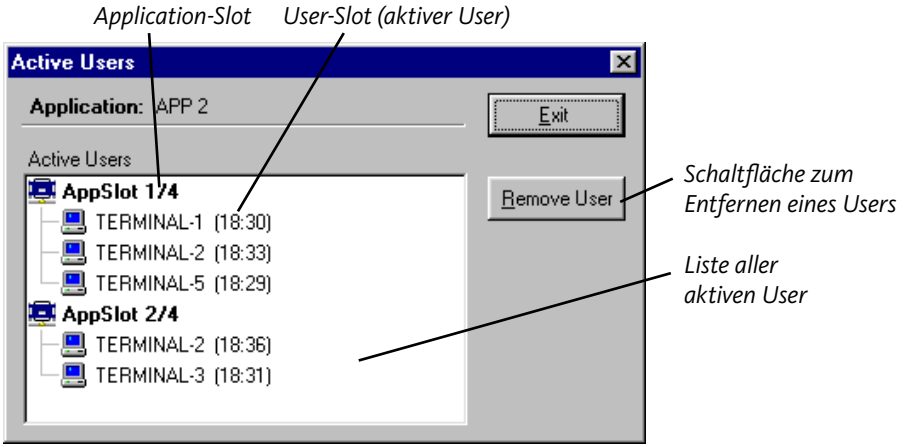
In diesem Feld kann das Zeitintervall für die Aktualisierung (Refresh) der Server-Datei eingestellt werden. Die letzte durchgeführte Aktualisierung wird in »Last-Refresh« angezeigt. In der Regel sollte das Zeitintervall für die Aktualisierung zwischen 5 und 10 Minuten gewählt werden. Diese Angabe ist global; das bedeutet, dass sie für alle Server-Dateien gilt.

User-TimeOut

In diesem Feld kann der sogenannte »User-TimeOut« eingestellt werden. Der User-TimeOut repräsentiert das Zeitlimit nach dem der User automatisch aus der Server-Datei entfernt wird. Diese Funktion sorgt im Falle einer abnormalen Beendigung der Anwendung (Absturz) auf einem Client für die Freigabe des User-Slots in der Server-Datei, dieser würde sonst belegt bleiben. Zusätzlich kann so ein User-Eintrag auch manuell entfernt werden. Der User-TimeOut ist global, das bedeutet, dass der eingestellte TimeOut für alle Server-Dateien gilt.

Aktive-User

Dieses Feld wird ständig aktualisiert und zeigt für die markierte Anwendung die Gesamtanzahl aller aktiven User an. Mit der Schaltfläche »Aktive-Users« kann eine Detail-Liste der aktiven User angezeigt werden. In dieser Liste werden die Application-Slots mit aktiven User-Slots angezeigt.
Die Application-Slots sind im Format Application-Slot-Nummer/Dongle-Nummer dargestellt. Zum Beispiel bedeutet AppSlot 1/4: Application-Slot 1 aus dem Dongle 4.



In diesem Beispiel ist eindeutig zu erkennen, dass an dem MxNet-Server vier oder mehrere Matrix-Dongle angeschlossen sind, wobei die Application-Slots 1 und 2 des 4. Dongles mit Usereinträgen belegt sind. Jeder belegte User-Slot zeigt den jeweiligen Terminal-Namen an, auf dem die Anwendung läuft, sowie die Uhrzeit, der letzten Aktualisierung des User-Slots durch die Anwendung.

Sollte Ihre Anwendung auf irgend einem Terminal abnormal beendet werden, dann können Sie entweder den User-Slot manuell aus der Liste entfernen oder den User-Time-Out abwarten. Beim Erreichen des User-TimeOut wird der User-Slot mit dem nächsten Refresh der Server-Datei automatisch vom MxNet-Programm entfernt.

Hinweis: Es ist nicht empfehlenswert, aktive »User-Slots« aus der Liste zu enfernen, solange die Anwendung noch aktiv ist. Aber selbst wenn aus Versehen ein »User-Slot« gelöscht wird, besteht keine Gefahr für die Funktionalität Ihrer Anwendung.

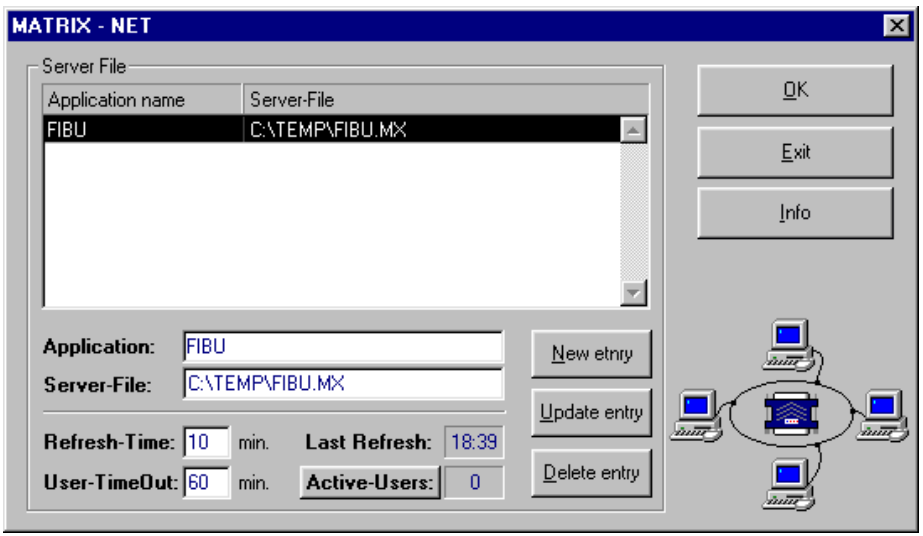
Achtung!
Es ist sehr wichtig, dass die Systemzeit der Rechner im Netzwerk synchronisiert ist, sonst kann der »Datei-TimeOut/User-TimeOut« nicht korrekt berechnet werden. Die maximal zugelassene Abweichung der Systemzeit zwischen Client und Server darf nicht die Anzahl der Minuten übersteigen, die im MxNet-Serverprogramm in »Refresh-Time« eingestellt wurde. Die Systemzeit der Client-Rechner kann mit folgendem Kommando mit der Systemzeit des Servers synchronisiert werden. Zur Automatisierung, kann dieses Kommando beim Bootvorgang des Client-Rechners ausgeführt werden.

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

Beispiel für den Schutz einer Anwendung im Netzwerk mit MxNet.

Hiermit soll an einem Beispiel verdeutlicht werden, wie der Schutz eines Programms im Netzwerk schnell zu realisieren ist.
Schließen Sie den Dongle an einen (beliebigen) Computer des Netzwerks an und installieren Sie die mitgelieferte Software. Auf dem Rechner mit dem Dongle starten Sie das Programm Matrix-NET (MxNet32.exe), mit dem Sie für jedes zu schützende Programm eine sogenannte »Server-Datei« erzeugen können. Man kann aber auch mehrere Anwendungen über die gleiche Server-Datei verwalten.

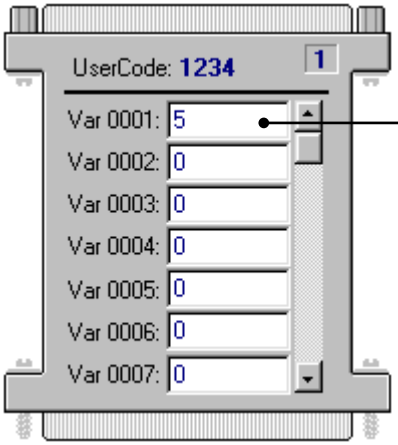
In unserem Beispiel soll die zu schützende Anwendung ein Finanzbuchhaltungsprogramm sein, das von maximal fünf Benutzer gleichzeitig verwendet werden soll.
In das Eingabefeld »Application« des Matrix-NET-Programms tragen Sie einen Namen für das zu schützende Programm ein; in diesem Beispiel könnte man die Bezeichnung »FIBU« verwenden. Der Name ist frei wählbar und muss nicht dem richtigen Programmnamen entsprechen. Die Verwaltung der Netzwerk-Lizenzen sollen in diesem Beispiel über die Server-Datei *C:\TEMP\FIBU.MX* verwaltet werden.



Der Name der Server-Datei muss immer mit absolutem Pfad eingetragen werden und den Namenskonventionen des Betriebssystems entsprechen.
Das Zeitintervall für die automatische Aktualisierung der Server-Datei kann in diesem Beispiel auf 10 Minuten eingestellt werden.

Das Zeitintervall für den User-TimeOut – also die Zeit nach der der User automatisch freigegeben werden soll, falls sich dieser nicht mehr meldet (Absturz) – soll hier auf 60 Minuten eingestellt werden.

Programmieren Sie jetzt den Dongle mit der entsprechenden Anzahl der gleichzeitig zulässigen Programmbenutzer; in unserem Beispiel wird also die »5« in der ersten Dongle-Variable eingetragen und gespeichert.



Mit dieser Einstellung wird ein Application-Slot (Var0001) mit 5 Usert-Slots definiert.

Jetzt müssen nur noch die Aktivierung, das *LogIn* und *LogOut* an das Netzwerk mit den mitgelieferten API-Funktionen (siehe »API-Funktionen Referenz« weiter hinten im Buch) in das zu schützende Programm eingebaut werden. Beachten Sie bei den Aufruf-Parametern der Funktionen, dass Sie den korrekten Namen der Server-Datei und den richtigen – d.h. für diese Anwendung benutzten – Application-Slot verwenden.

Beispiel für die Funktionsaufrufe

Für die Vereinfachung des Beispiels gehen wir davon aus, dass nur ein Dongle angeschlossen ist. Die Angabe des Anschlusses *DNG_PORT* hat beim Netzwerkzugriff keine Bedeutung.

```
/*-- Matrix-API starten --*/
Init_MatrixAPI();

DNG_PORT = 1;  // hat beim Netzwerkzugriff keine Bedeutung
DNG_NR   = 1;
AppSlot  = 1;

/*-- Aktivierung des Netzwerk-Zugriffs in der Matrix-API --*/
SetConfig_MatrixNet(1, „F:\\TEMP\\FIBU.MX“);

/*-- LogIn: Das Programm belegt jetzt einen 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: Den User-Slot freigeben --*/
LogOut_MatrixNet(UserCode, AppSlot, DNG_NR);

/*-- Matrix-API beenden --*/
Release_MatrixAPI();
exit;
```

Was Sie unbedingt in Ihrer Anwendung berücksichtigen sollten

- Ihre geschützte Anwendung muss immer beim Start einen User-Slot belegen. Ist kein User-Slot mehr frei, dann sollte die Anwendung mit einer entsprechenden Meldung beendet werden.
- Ihre Anwendung muss von Zeit zu Zeit den User-Slot Eintrag aktualisieren, damit das TimeOut-Limit nicht erreicht wird.
- Wählen Sie das Zeitintervall »Refresh-Time« und »User-TimeOut« nicht zu klein, damit das System möglichst wenig belastet wird.
- Bei abnormaler Beendigung (Absturz) Ihrer Anwendung bleibt der User-Slot belegt. Dieser kann dann manuell oder automatisch vom MxNet Server-Programm – beim erreichen des TimeOut-Limits (siehe »Einstellungen des MxNET Server-Programms«) – entfernt werden. Es ist nicht notwendig, den User-Slot zu löschen bevor die abnormal beendete Anwendung erneut gestartet wird. Diese findet den vorhandenen User-Slot und aktualisiert ihn.
- Beim Beenden Ihrer Anwendung auf einem Client sollten Sie immer den User-Slot freigeben.
- Sie sollten auf jedem Client die Datei *matrix.ini* in *\windows\system* installieren und die Einträge AccessLPT und AccessUSB in dieser Datei auf OFF setzen. Somit besteht keine Notwendigkeit, die LPT/USB-Treiber auf dem Netzwerk-Client zu installieren.

Vorteile und Nachteile des MxNET Lizenz-Management

Vorteile

- Die MxNet-Methode bietet eine kosteneffektive Verwaltung der Netzwerk-Lizenzen für mehrere Anwendungen mit nur einem Dongle.
- Da MxNet unabhängig von Netzwerk-Protokollen funktioniert, kann es problemlos in jedem Netzwerk-System eingesetzt werden.
- Alle Funktionen der Matrix-API – mit Ausnahme jener zum Schreiben von Daten in den Dongle (Dongle_WriteData und Dongle_WriteDataEx) – können sowohl für Einzelplatz-Anwendungen sowie auch für das Netzwerk-Management verwendet werden.
- Für die Verwaltung der Netzwerklizenzen (User-Slots) werden nur zwei zusätzliche API-Funktionen verwendet. Dadurch gestaltet sich die Integration des Schutzes in Ihre Anwendung sehr einfach.

Nachteile

- Die MxNet-Methode ist ein asynchroner Prozess. Das bedeutet, dass die Abfrage der Dongle-Informationen und die Abfrage der Server-Datei zeitversetzt stattfindet. Somit können über MxNet keine Daten in den Dongle gespeichert werden.

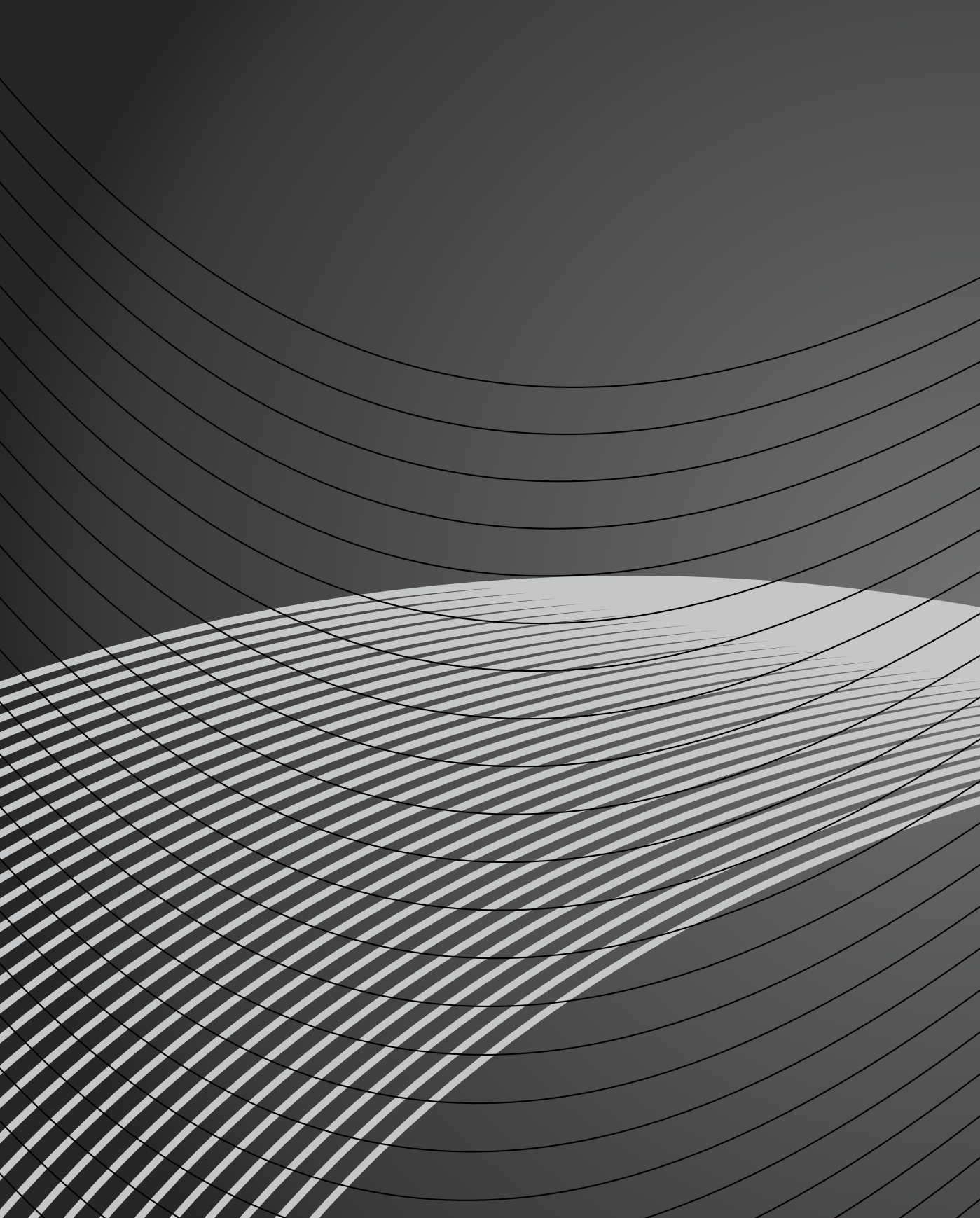
Vorteile und Nachteile des Lizenz-Management mit je einem Dongle pro Arbeitsplatz

Vorteile

- Diese Methode ist effektiver, sofern Sie die Nutzung Ihrer Anwendung nur als Inhouse-Lösung im lokalen LAN Netzwerk zulassen möchten, so dass externe Arbeitsplätze (Remote) die Anwendung ohne einen entsprechenden Dongle nicht nutzen sollten.
- Je nach Bedarf können die Dongles unterschiedliche Informationen beinhalten.

Nachteile

- Bei vielen Arbeitsplätzen ist diese Methode kostenintensiver.



4

Auslieferung von geschützen Programmen

Auslieferung von geschützten Programmen

Anwendungen für Windows-Betriebssysteme

Die mit Matrix geschützte Software benötigt bei der Laufzeit folgende Dateien, die Sie zusammen mit Ihrer Software mitliefern müssen:

Matrix-API

matrix16.dll	–	DLL für 16 Bit-Anwendungen
matrix32.dll	–	DLL für 32 Bit-Anwendungen
matrix64.dll	–	DLL für 64 Bit-Anwendungen
matrix.ini	–	Konfigurationsdatei zum Ändern der Einstellungen bei Ihrem Kunden

LPT-Treiber

iwport.vxd	–	LPT-Treiber für Windows 95/98/ME
iwport.sys	–	LPT-Treiber für Windows NT/2000/XP/Vista
drv_inst.exe	–	Installationsprogramm für den Windows NT/2000/XP/Vista LPT-Treiber

USB-Treiber (erforderlich nur in der Betriebsart »Driver-Mode«)

iwusb.sys	–	USB-Treiber für Windows 98/ME/2000/XP/Vista
iwusb_x64.sys	–	USB-Treiber für Windows XP-64
iwusb.inf	–	USB-Treiberinformationsdatei
inf_inst.exe	–	Installationsprogramm für den Windows 98/ME/2000/XP/Vista USB-Treiber

Netzwerk-Anwendung (erforderlich nur im Netzwerkbetrieb)

mxnet32.exe	–	MxNET Server-Anwendung

Support-Tool

mxcheck.exe	–	Tool für die Überprüfung der installierten Treiber und Komponenten

16 Bit-Anwendungen

16 Bit-Anwendungen benötigen unter Windows 3.x nur die *matrix16.dll* und keine Treiber.

16 Bit-Anwendungen benötigen unter Windows 95/98/ME und NT/2000/XP/Vista die Dateien *matrix16.dll*, *matrix32.dll* und den entsprechenden LPT/USB Treiber.

32 Bit-Anwendungen

32 Bit-Anwendungen benötigen nur die Datei *matrix32.dll* und den entsprechenden LPT/USB Treiber.

64 Bit-Anwendungen
64 Bit-Anwendungen benötigen nur die Datei *matrix64.dll* und den entsprechenden USB Treiber.

Was Sie Ihren Kunden mitliefern müssen

API
Die DLL-Dateien müssen Sie bei Ihrem Kunden ins Verzeichnis *\windows\system* kopieren.
Sofern Sie eine 32 Bit-Anwendung haben, reicht es aus, die Datei *matrix32.dll* an Ihren Kunden auszuliefern. Wenn Sie die Matrix-API statisch eingebunden haben oder Ihre Anwendung nur mit Matrix-Crypt geschützt haben, ist keine DLL erforderlich, da die API bereits in Ihrer Anwendung integriert ist.

LPT-Treiber
Den LPT VXD-Treiber *iwport.vxd* für Windows 95/98/ME müssen Sie bei Ihrem Kunden in das Verzeichnis *\windows\system* kopieren.

Der LPT SYS-Treiber *iwport.sys* für Windows NT/2000/XP/Vista muss installiert werden.
Die Installation dieses Treibers kann entweder automatisch durch die Matrix-API oder manuell mit dem Programm *drv_inst.exe* und der dazugehörigen Treiber-Datei *iwport.sy_* erfolgen.

Falls Sie die automatische Installation bevorzugen, reicht es aus, den Treiber in das Verzeichnis *\windows\system32\drivers* zu kopieren. Die Treiberinstallation wird dann beim Start Ihrer geschützten Anwendung automatisch durchgeführt.
Fals Sie die manuele Installation bevorzugen, liefern Sie das Instalationsprogram *drv_inst.exe* und die dazugehörige Treiberdatei *iwport.sy_* an Ihren Kunden aus. Kopieren Sie diese Dateien durch Ihre Setup-Routine in ein Verzeichnis Ihrer Wahl (z. B. in ein Unterverzeichnis Ihrer Anwendung). Das Installationsprogramm *drv_inst.exe* können Sie aus Ihrer Setup-Routine aufrufen.
Mehr dazu finden Sie im Kapitel »Installation«, Abschnitt »LPT-Treiberinstallation unter Windows NT/2000/XP/Vista« und in der Datei *Readme* aus dem Verzeichnis *\nt_drv.*

Hinweis: Vergessen Sie nicht, dass in beiden Fällen unter Windows NT/2000/XP/Vista nur Benutzer mit entsprechenden Zugriffsrechten (wie z. B. der Administrator) Treiber installieren dürfen.

USB-Treiber

Wenn Sie den Dongle in der Betriebsart »HID-Mode« ausliefern, ist keine Treiberinstallation erforderlich.

Wenn Sie den Dongle in der Betriebsart »Driver-Mode« ausliefern, gehen Sie wie folgt vor:

Den USB-Treiber *iwusb.sys, iwusb_x64, iwusb.inf* für Windows 98/ME/2000/XP/Vista und das dazugehörige Installationsprogramm *inf_inst.exe* sollten Sie an Ihren Kunden ausliefern und durch Ihre Setup-Routine in ein Verzeichnis Ihrer Wahl kopieren (z. B. in ein Unterverzeichnis Ihrer Anwendung). Das Installationsprogramm *inf_inst.exe* können Sie aus Ihrer Setup-Routine aufrufen.

Mehr dazu finden Sie im Kapitel »Installation«, Abschnitt »USB-Treiberinstallation unter Windows 98/ME/2000/XP/Vista« und in der Datei *Readme* aus dem Verzeichnis *\drv_usb*.

Netzwerkanwendung

Das MxNet Server-Programm ist nur dann notwendig, wenn Sie das Lizenz-Management für Ihr Netzwerkprogramm mit MxNet abwickeln. Diese Anwendung sollte in das Verzeichnis *\windows\system* kopiert werden.

Das MxNet Server-Programm kann auch als Windows-Service registriert werden, so dass das Programm automatisch beim Boot-Vorgang von Windows gestartet wird. Der Vorteil eines Services gegenüber einer Autostart-Eintragung ist, dass der Service auch dann gestartet wird, wenn kein User-LogIn in Windows erfolgt.

Die Registrierung von MxNet als Service können Sie direkt aus Ihrer Setup-Routine vornehmen, indem Sie MxNet mit dem entsprechenden Parameter starten. Die verfügbaren Aufrufparameter lauten:

mxnet32.exe	-i	(MxNET-Service installieren)
mxnet32.exe	-r	(MxNET-Service deinstallieren)

Konfiguration und Tools

Optional können Sie auch die Datei *matrix.ini* ausliefern. Mit entsprechenden Einträgen in dieser Datei können Sie z. B. – falls erforderlich – die LPT- oder USB-Unterstützung vollständig ausschalten (AccessLPT=ON/OFF, AccessUSB=ON/OFF). Wenn z. B. die LPT Unterstützung ausgeschaltet wird, ist auch die Installation der LPT-Treiber nicht mehr erforderlich. Dies macht dann Sinn, wenn Sie z. B. nur USB-Dongle auf Rechner/Notebooks verwenden, die über keinen LPT-Anschluß verfügen.

Diese Datei sollte in das Verzeichnis *\windows\system* kopiert werden. Falls diese Datei nicht vorhanden ist, werden von der Matrix-API die Standardeinstellungen verwendet (alles ON).

Das Programm *mxcheck.exe* für die Überprüfung der installierten API und Treiber im System können Sie an Ihren Kunden als Support-Tool ausliefern.

Anwendungen für die Betriebssysteme Linux und Mac-OS X

Für Linux- und Mac-OS X Anwendungen lesen Sie bitte die Datei *Readme_redist*.

Hinweis: Unter www.tdi-matrix.de finden Sie immer die aktuellen Versionen und neuesten Tools zum Download.

Remote Update

Allgemeine Beschreibung

Matrix Remote Update ermöglicht Ihnen, den Speicherinhalt eines Dongle, der bereits bei Ihrem Endkunden im Einsatz ist, nachträglich zu prüfen und/oder zu ändern. Dafür ist es nicht notwendig, dass Ihnen der Kunde den Dongle zurückschickt. Das macht dann Sinn, wenn z. B. weitere Programmteile der geschützten Software nachträglich freigeschaltet werden sollen.

Matrix Remote Update besteht aus einem Tool, genannt »MxGenDat«, mit dem eine verschlüsselte EXE-Datei, genannt *MxModUpd.exe*, erzeugt werden kann. Das erzeugte Update-Programm wird dann zum Endkunden geschickt (z. B. per Email) und von ihm gestartet. Der Kunde muss dafür nicht online sein.
Das Update-Programm korrespondiert mit dem aufgesteckten Dongle und führt dann die gewünschten Operationen vor Ort bei Ihrem Kunden aus (verändert und/oder prüft den Speicherinhalt des Dongle). Durch bestimmte optionale Einstellungen kann die Authentifizierung des Dongle beliebig eingeschränkt werden. Nach dem Ausführen des Update-Programms ist dann der Speicherinhalt des Dongle beim Endkunden in gewünschter Weise abgeändert.
Das Update-Programm ist in der Lage, auch MK/MKU Module zu ändern, die sonst nur über einen MasterKey änderbar sind.
MxGenDat ist eine effiziente Alternative zum Online-Update, welches komplizierter ist und oft vom Endkunden abgelehnt wird.

Vorteile von Matrix Remote Update

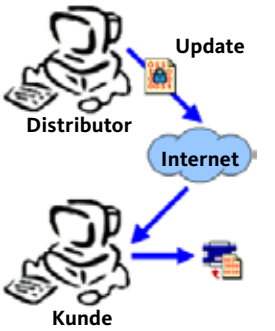
- Es werden nachträglich weitere Programmteile ihrer Software frei geschaltet
- Die Nutzerrechte Ihres Kunden werden angepasst
- Die Lizenzbedingungen ändern sich nachträglich, z. B. Leihsoftware
- Der Kunde kauft ein weiteres Softwarepaket und benutzt denselben Dongle dafür
- Spart Zeit und Geld, der Versand des Dongle per Post ist nicht nötig
- Das Update-Programm kann kundenspezifisch für einen bestimmten Dongle erzeugt werden
- Der Kunde kann die neuen Softwarepakete sofort ohne Zeitverzögerung nutzen
- Reduziert substanziell Ihren Support-Aufwand
- Es können auch MK/MKU Module geändert werden

Wie funktioniert Matrix Remote Update?

Matrix Remote Update ist sehr einfach zu verwenden. Nur zwei Schritte sind bis zum geänderten Dongle nötig:

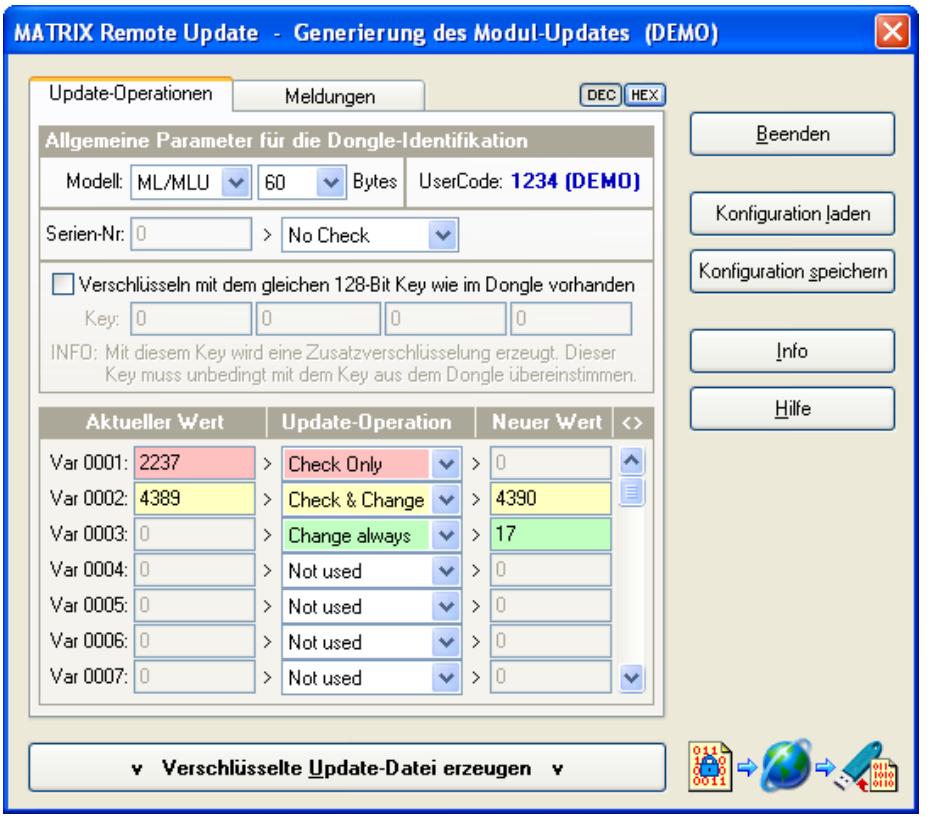
1. Der Distributor erzeugt das kundenspezifische Update-Program und sendet dieses dem Kunden zu (z. B. per Email).

2. Der Kunde empfängt das Update-Program und führt dieses aus.



Grundeinstellungen

Folgende Grundeinstellungen stehen Ihnen im *MxGenDat* zur Verfügung:



Registerkarte »Update Operationen«

Modell
Tragen Sie hier das Modell des Dongle ein, welches der Endkunde benutzt.

Serien-Nr. (optional)
Tragen Sie hier die Seriennummer des Dongle ein, für welches das Update-Programm erzeugt werden soll. Wenn keine Prüfung der Seriennummer aktiviert wird (No Check), ändert das Update-Programm den zuerst gefundene Dongle mit passendem UserCode. Durch Angabe der Seriennummer kann zusätzlich die Stärke der Authentifizierung des Kunden bzw. des Dongle gesteigert werden.

Hinweis: Achten Sie auf die korrekte Angabe der Seriennummer!

Verschlüsseln mit dem gleichen 128-Bit Key wie im Dongle vorhanden (optional)
Durch Aktivierung dieser Option können Sie zusätzlich die Stärke der Authentifizierung steigern. Die Update-EXE wird bei der Erzeugung zusätzlich mit diesem 128-Bit Key verschlüsselt. Die Entschlüsselung wird dann bei der Ausführung durch den Dongle Ihres Kunden durchgeführt. Eine korrekte Entschlüsselung bzw. Ausführung kann nur dann erfolgen, wenn der Dongle des Kunden den gleichen Key beinhaltet.
Ausgehend davon, dass jeder Kunde einen anderen Key hat, wird mit dieser Methode verhindert, dass Ihre Kunden Updates ausführen können, die für andere Kunden bestimmt sind.

Hinweis: Achten Sie auf die korrekte Angabe des 128-Bit Keys!

Variablenliste

Aktueller Wert:	Wert, der bereits im Dongle vorhanden ist
Update Operation:	Not Used/Check Only/Check & Change/Change always
Neuer Wert:	Wert, auf den die Variable des Dongle verändert werden soll

Diese zunächst komplex wirkende Feldgruppe ermöglicht einen differenzierten Zugriff auf den Speicher des Dongle:

‹ Change always ›
Ist die einfachste Variante und speichert den Inhalt aus dem Feld »Neuer Wert« in das Variablenfeld des Dongle. Unabhängig davon, welchen Inhalt diese Variable im Dongle bereits hat.

‹ Check & Change ›
Ist die komplexere Variante und vergleicht erst das Feld »Aktueller Wert« mit dem aus dem Matrix-Dongle. Wenn diese gleich sind, dann wird der neue Inhalt aus dem Feld »Neuer Wert« in den Dongle gespeichert.

‹ Check Only ›
Vergleicht nur das Feld »Aktueller Wert« mit dem aus dem Dongle, verändert diesen aber nicht.

‹ Not Used ›
Kann ausgewählt werden, wenn die jeweilige Variable nicht benutzt wird.

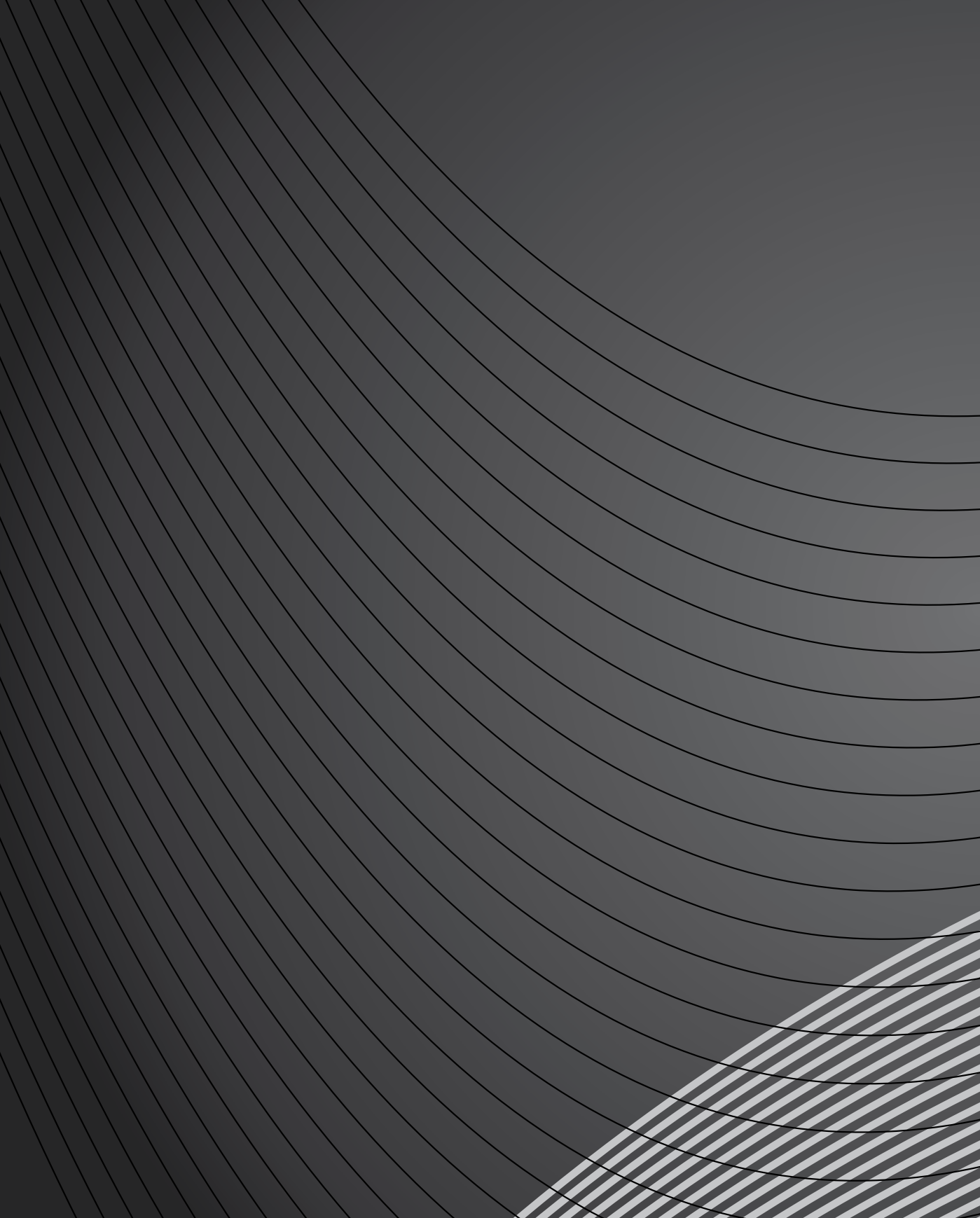
Registerkarte »Meldungen«

Die Meldungen, die vom Update-Programm angezeigt werden, können Sie hier Ihren Anforderungen entsprechend anpassen (z.B. verschiedene Sprachen).

Konfiguration speichern/laden
Erleichtert die Verwaltung verschiedener Einstellungen. Die vorgenommenen Einstellungen können als Projekt abgespeichert werden. Abgespeicherte Projekte können dann zu einem späteren Zeitpunkt geladen werden.

v Verschlüsselte Update-Datei erzeugen v
Erzeugt das Update-Programm *MxModUpd.exe*, das Sie Ihrem Kunden schicken.

Hinweis: Wenn Sie das Update-Programm für MK/MKU Modelle erzeugen möchten, dann muss der MasterKey angeschlossen sein!
Das erzeugte Update-Programm ist dann in der Lage, auch MK/MKU Dongle bei Ihrem Kunden ohne MasterKey zu ändern.



5

Richtlinien für einen guten Softwareschutz

Richtlinien für einen guten Softwareschutz

Um ein Programm effektiv vor Raubkopien zu schützen, muss man sich immer über die Grenzen der Schutzvorkehrungen im Klaren sein. Bei einem qualifizierten Softwareschutz wie Matrix ist die Hardware-Sicherheit so hoch, dass es hier kaum Manipulationsmöglichkeiten für Hacker gibt. **Die Schwachstelle ist also die Einbindung des Schutzes in Ihre Programme.** Ein Hacker wird hier ansetzen und versuchen, die angestrebte Einheit aus Dongle und der zu schützenden Software zu trennen. Ziel muss es also sein, Matrix untrennbar bzw. schwer trennbar in Ihre Programme einzubinden.

Es kann natürlich keine Patentlösung für einen absolut sicheren Schutz geben, letztlich können Sie aber mit Phantasie und Kreativität dafür sorgen, dass Ihre Investition sicher geschützt und den Raubkopierern das Leben schwer gemacht wird.

Beachten Sie: *Eine schnelle Einbindung in Ihr Programm kann niemals so sicher sein wie eine durchdachtere und eventuell etwas zeitaufwendigere Lösung.*

Um einen guten Schutz für Ihre Anwendung zu erreichen, sollten Sie die folgenden Aspekte berücksichtigen und diese bei der Entwicklung Ihres Schutzes einfließen lassen.

Gibt es das »unknackbare« Programm?

Diese Frage muss zweifelsfrei mit »nein« beantwortet werden – denn jeder Schutz kann erkannt und beseitigt werden. Jeder Anbieter von Sicherheits-Produkten, der verspricht, er könnte den perfekten Schutz für Ihre Software garantieren, kann durchaus als wenig glaubwürdig angesehen werden. Durch die mit Matrix zur Verfügung gestellten Maßnahmen ist es jedoch möglich, den Aufwand für die Analyse und die Beseitigung des Schutzes beliebig zu steigern.

Werkzeuge des Hackers

Der Hacker wird in der Regel ein Debug-Programm benutzen, z.B. im Minimalfall *DEBUG.EXE*, das mit dem Betriebssystem ausgeliefert wird. Sie sollten aber immer davon ausgehen, dass er aufwendigere Werkzeuge benutzt, mit denen Ihr Programm in einzelnen Schritten abgearbeitet werden kann. Durch Analyse des Assembler-/Programm-Codes wird er versuchen, die Aufrufe des Dongle zu finden und diese so zu verändern, dass das zu knackende Programm auch ohne angeschlossenen Dongle funktioniert.
Wenn Sie z. B. die Dongle-Aufrufe wahllos in Ihrem Programm verstreut haben, wird der Aufwand für den Hacker natürlich höher als bei einem einmaligem Aufruf beim Programmstart.

Abfrage des Dongle beim Programmstart

Beim Programmstart sollte immer eine einfache Matrix-Abfrage durchgeführt werden. Damit wird sichergestellt, dass der Dongle vorhanden ist. Dieser erste Schritt dient lediglich zur korrekten Erkennung des Dongle. Für den eigentlichen Schutz der Software ist diese Maßnahme jedoch kaum relevant, da sie relativ einfach zu umgehen ist. Im weiteren Verlauf der Anwendung kann jetzt aber vorausgesetzt werden, dass der Dongle vorhanden ist.

Häufige Abfragen

Der reine Schutz beim Start kann also leider ziemlich leicht erkannt und umgangen werden. Außerdem kann der Anwender den Dongle schlicht an einen anderen Computer stecken und eine zweite (dritte, vierte...) Kopie Ihres Programms starten. Jede beabsichtigte Verhinderung der Mehrfach-Nutzung wäre von vornherein zum Scheitern verurteilt. Aus diesem Grund sollten Sie in Ihrem Programm den Dongle häufig abfragen.

Schutzmaßnahmen während des Programmablaufes

Die Dongle-Abfragen sollten – soweit möglich – auch in die tiefsten Programm-Ebenen eingestreut werden. Breite Verteilung der Schutzmaßnahmen ist jeder einfach zu erkennenden Einzelmaßnahme deutlich überlegen.

Verstreuen der Abfragen im Code

Einen guten Schutz erreichen Sie einfach durch den Einbau von Matrix-Aufrufen verstreut über das ganze Programm. Programmsequenzen zur Abfrage des Dongle sollten nicht immer gleich zusammenhängend programmiert werden, da sonst immer (fast) identischer Code erzeugt würde. Damit wären die Dongleabfragen für den Angreifer leichter zu lokalisieren. Sie können den Schutz verstärken, wenn Sie die Programmsequenzen mit den Dongleabfragen im gesamten Source-Code – soweit möglich – verteilen.

Verwendung des Matrix-Speichers

Sie sollten auch den verfügbaren Speicher aus dem Dongle in die Sicherheits-Abfragen mit einbeziehen. So könnten Sie z. B. Kunden-Codes, Seriennummer oder Programmvariablen in den Speicher des Dongle ablegen und diese während des Programmablaufs auswerten.

Sie können Ihr Programm auf einfache Weise so schützen, dass ein Knackversuch ohne den richtigen Dongle praktisch aussichtslos ist, da für den Programmablauf notwendige Daten, Parameter oder Codeteile nur im Dongle vorhanden sind. Da der Hacker diese Informationen nicht erraten kann, muss er für den Knackversuch den richtigen Dongle haben.

Hinterlegen Sie im nicht genutzten Speicher des Dongles irgendwelche Zufallszahlen, deren Vorhandensein bzw. Wert Sie dann im weiteren Programmablauf wieder auslesen und prüfen.

Arbeiten mit der Kryptographie

Nutzen Sie die Möglichkeiten, die die Encrypt- und Decrypt-Funktionalität der Dongle bietet, um Ihre Programmdaten zu ver-/entschlüsseln. Mit einem von Ihnen selbst festgelegten 128-Bit XTEA-Schlüssel, der nicht aus dem Dongle ausgelesen werden kann, lassen sich wichtige, fixe Programmdaten erst zur Programmlaufzeit entschlüsseln und in sinnvolle Werte zurückverwandeln.

Die Verschlüsselung können Sie z.B. mit der Matrix-Programmiersoftware durchführen und dann diese verschlüsselten Werte – z.B. Konstanten, Berechnungsvariablen usw. – in Ihr Programm einbauen.

Damit Ihr Programm noch sinnvolle Arbeit leistet, muss der Dongle zur Laufzeit immer an den Computer angeschlossen sein, denn nur durch den Dongle lassen sich die Werte entschlüsseln. Eine Raubkopie ohne Dongle wäre also unbrauchbar.

Verwendung der Verschlüsselung für User-Authentifizierung »on-the-fly«

Implementieren Sie flexible Authentifizierung und e-Commerce Lösungen durch Verwendung des Dongle als »Crypto Engine«. Sie können zufällig verschlüsselte Sequenzen bei der Laufzeit erzeugen, um User-Passworte »on-the-fly« zu prüfen, ohne die echten Passworte oder irgendwelche User-Authentifizierungscodes in den Matrix-Dongle zu speichern.
Ein Hacker kann diese Authentifizierung nicht simulieren oder ersetzen, weil es sich hier um ständig wechselnde Bedingungen bei der Laufzeit handelt.

Programmiertechnik

Verlieren Sie keine Zeit dafür, die Abfragen in Ihrer Programmiersprache durch »unsaubere« Programmierung zu verschleiern, denn viele Compiler optimieren den Code meist so, dass auch in diesem Fall »sauberer« Assemblercode generiert wird.

Verwenden Sie in Ihrem Programm keine geheimen Schalter, die zu internen Testzwecken die Sicherheitsabfragen deaktivieren. Damit könnte durch eine einzige Änderung im Programmcode das gesamte Sicherheitssystem ausgeschaltet werden. Achten Sie auch darauf, dass die Programmversion, die Sie ausliefern, ohne Debug-Informationen erstellt wird.

Verwenden Sie keine einfachen Integer-Vergleiche

Nach dem Lesen einer Variable aus dem Speicher des Matrix-Dongles: Um den Inhalt dieser Variable zu prüfen, verwenden Sie komplexe Formeln anstelle von einfachen Integer-Vergleichen.

Sie möchten z.B. prüfen, ob die Variable den Wert 225 hat. Statt den einfachen Vergleich *if(MyVar == 225)....* zu machen, verwenden Sie eine komplexere Formel:

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

Durch Verwendung von Fließkomma-Operationen werden sehr aufwändige und komplexe Stukturen im Maschinencode erzeugt.

Ein Hacker wird viel mehr Aufwand haben, solche Code-Strukturen und deren Bedeutung zu verstehen.

Maßnahmen nach Erkennen eines Angriffes

Wenn Sie über eine der oben beschriebenen Methoden oder über Konsistenzchecks einen Angriff oder eine Veränderung Ihres Programms erkannt haben, gibt es eine Vielzahl von Möglichkeiten, das Programm unbrauchbar zu machen. Dabei sollten die getroffenen Maßnahmen so weit wie möglich kausal von der Erkennung des Angriffes »entkoppelt« werden. Dazu können die in den folgenden Abschnitten beschriebenen Maßnahmen verwendet werden.

Verspätete Reaktion

Zeitliche Verschiebung der Abwehrmaßnahmen: Erst nach einiger Zeit (von Sekunden bis zu Tagen bei Ausnutzung des Systemdatums) lässt sich das Programm nicht mehr starten.

Verschleierung

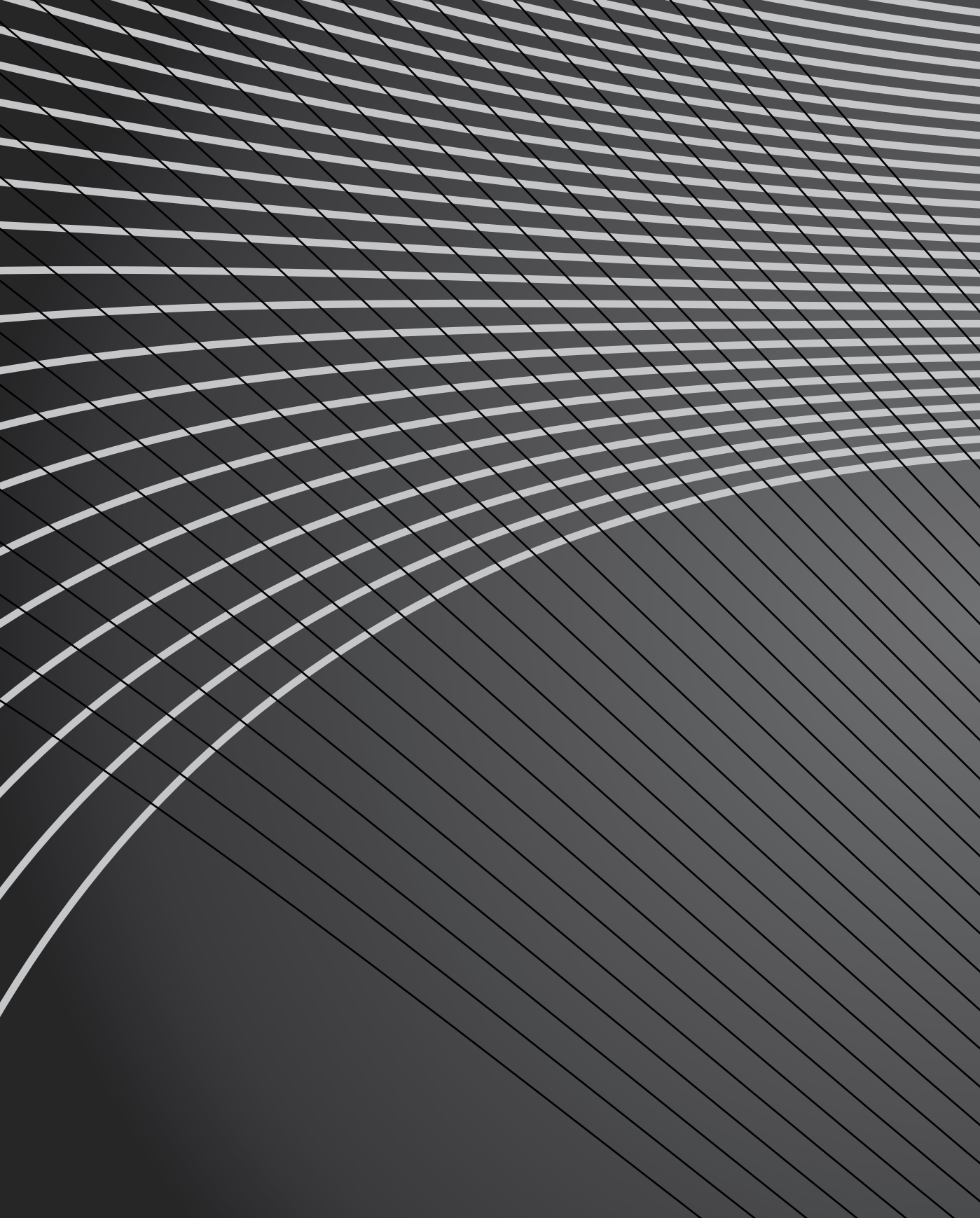
Verschleierung des kausalen Zusammenhangs: Eine kleine Änderung hat eine Veränderung von vielen anderen Werten zur Folge, von denen wiederum nur einer die eigentliche Maßnahme initiiert.

Verfälschung der Ergebnisse

Das Programm sollte nach dem Erkennen eines Angriffs nicht sofort beendet werden, sondern vielmehr weiter arbeiten und gegebenenfalls unsinnige Ergebnisse liefern.

Einschränkungen der Programmfunktionen

Gegenmaßnahmen des Programmherstellers können sich auch auf das Abschalten einzelner Programmfunktionen – wie z.B. Drucken, Speichern etc. – beschränken. Der Hacker merkt erst später, dass seine Bemühungen fruchtlos waren.



6

API-Funktionen Referenz

API-Funktionen Referenz

Übersicht der API-Funktionen

Funktionsname	Beschreibung	Verfügbar für LPT	USB	NET	Seite
Init_MatrixAPI	Startet die Matrix API	✔	✔	✔	94
Release_MatrixAPI	Beendet die Matrix API	✔	✔	✔	94
GetVersionAPI	Gibt die Versionsnummer der Matrix-API zurück	✔	✔	✔	95
GetVersionDRV	Gibt die Versionsnummer des LPT-Treibers zurück	✔			95
GetVersionDRV_USB	Gibt die Versionsnummer des USB-Treibers zurück		✔		96
SetW95Access	Kommunikation mit dem LPT- Dongle unter Win9x mit oder ohne VXD-Treiber	✔			97
GetPortAdr	Gibt die Adresse des LPT-Anschlusses zurück	✔			97
PausePrinterActivity	Hält den Windows Print Spooler an (nur unter Win 9.x/ME/NT/2000/XP/Vista)	✔			98
ResumePrinterActivity	Gibt den Windows Print Spooler wieder frei (nur unter Win 9.x/ME/NT/2000/XP/Vista)	✔			98
Dongle_Find	Sucht den Dongle und gibt den LPT/USB Port zurück, an dem dieser gefunden wurde	✔	✔	✔	99
Dongle_FindEx	Sucht alle LPT-Ports und Dongle und legt die Informationen im einem Datenpuffer ab	✔		✔	100
Dongle_Count	Gibt die Anzahl der vorhandenen Dongle an den LPT- oder USB-Port zurück	✔	✔	✔	101
Dongle_MemSize	Gibt die Speichergröße des Dongle in Bytes zurück	✔	✔	✔	103
Dongle_Model	Liest und gibt die Modellnummer des Matrix-Dongle zurück	✔	✔	✔	104
Dongle_Version	Liest und gibt die Versionsnummer des Matrix-Dongle zurück	✔	✔	✔	106
Dongle_ReadData	Liest die Daten ab Anfang des Speicher-bereiches aus dem Matrix-Dongle	✔	✔	✔	108

Funktionsname	Beschreibung	Verfügbar für LPT	USB	NET	Seite
Dongle_ReadDataEx	Liest die Daten ab einer bestimmten Speicheradresse aus dem Matrix-Dongle	✔	✔	✔	110
Dongle_WriteData	Daten ab Anfang des Speicherbereiches in den Dongle schreiben	✔	✔		112
Dongle_WriteDataEx	Daten ab einer bestimmten Speicher-adresse in den Dongle schreiben	✔	✔		114
Dongle_ReadSerNr	Liest und gibt die eindeutige Serien-nummer des Matrix-Dongles zurück	✔	✔	✔	116
Dongle_WriteKey	128-Bit XTEA-Schlüssel in den Dongle spei-chern	✔	✔		118
Dongle_GetKeyFlag	Prüft ob ein 128-Bit XTEA-Schlüssel ungleich Null in dem Dongle vorhanden ist	✔	✔		120
Dongle_EncryptData	8 Bytes langen Datenblock mit dem Dongle verschlüsseln	✔	✔		122
Dongle_DecryptData	8 Bytes verschlüsselter Datenblock mit dem Dongle entschlüsseln	✔	✔		124
Dongle_SetDriverFlag	USB-Betriebsart des Dongles ändern »HID-Mode« oder »Driver-Mode«		✔		126
Dongle_GetDriverFlag	Eingestellte Betriebsart des Dongles »HID-Mode« oder »Driver-Mode« ermitteln		✔		128
SetConfig_MatrixNet	Aktiviert oder deaktiviert den Netzwerkzugriff			✔	129
GetConfig_MatrixNet	Liest aus der Server-Datei Parameter, die im MxNet-Programm eingestellt wurden			✔	130
LogIn_MatrixNet	Meldet den Netzwerk-Client an und belegt/ aktualisiert den UserSlot in der Server-Datei			✔	131
LogOut_MatrixNet	Meldet den Netzwerk-Client ab und gibt den UserSlot in der ServerDatei wieder frei			✔	132

Beschreibung der API-Funktionen

Nachfolgend werden die API-Funktionen mit den jeweiligen Aufruf-Parametern beschrieben.

Init_MatrixAPI - Startet die Matrix API.

Syntax

```
short  Init_MatrixAPI( );
```

Die Matrix-API muss gestartet werden, bevor die API-Funktionen genutzt werden.

Rückgabe

0 Die API wurde erfolgreich gestartet.
-1 Die 32-Bit API (DLL) wurde nicht gefunden. Dieser Returncode wird nur von der 16-Bit API zurückgegeben.
-10 Die API ist gerade von einer anderen Applikation belegt. Dieser Rückgabewert kann nur dann auftreten, wenn die Option »Gemeinsamer Zugriff auf die parallele Schnittstelle« aktiviert wurde.
-20 Beim Zugriff auf den LPT-Treiber ist ein Fehler aufgetreten.
-21 Eine alte LPT-Treiberversion wurde gefunden, die API kann diesen LPT-Treiber nicht verwenden. Installieren Sie den neueren LPT-Treiber.
-22 Eine alte USB-Treiberversion wurde gefunden, die API kann diesen USB-Treiber nicht verwenden. Installieren Sie den neueren USB-Treiber.

Hinweis: Falls Sie nur USB Dongles verwenden und den LPT-Treiber nicht installieren, können Sie die LPT Error-Codes ignorieren.

Release_MatrixAPI - Beendet die Matrix API.

Syntax

```
short  Release_MatrixAPI( );
```

Die Matrix-API muss geschlossen werden, nachdem die API-Funktionen genutzt wurden, damit belegte Speicherbereiche wieder freigegeben werden.

Rückgabe

Diese Funktion hat keinen Rückgabewert.

GetVersionAPI - Gibt die Versionsnummer der Matrix-API zurück.

Syntax

```
long  GetVersionAPI( );
```

Unterstützt

LPT, USB, NET

Rückgabe

Es wird die Versionsnummer der Matrix-API zurückgegeben. Die höheren zwei Bytes sind die höhere Versionsnummer (HIWORD = MajorNumber). Die niedrigeren zwei Bytes sind die niedrige Versionsnummer (LOWORD =MinorNumber)

GetVersionDRV - Gibt die Versionsnummer des LPT-Treibers zurück.

Syntax

```
long  GetVersionDRV( );
```

Unterstützt

LPT

Rückgabe

Es wird die Versionsnummer des LPT-Treibers zurückgegeben. Die höheren zwei Bytes sind die höhere Versionsnummer (HIWORD = MajorNumber). Die niedrigeren zwei Bytes sind die niedrige Versionsnummer (LOWORD = MinorNumber).

GetVersionDRV_USB - Gibt die Versionsnummer des USB-Treibers zurück.

Syntax

```
long  GetVersionDRV_USB( );
```

Unterstützt

USB

Rückgabe

Es wird die Versionsnummer des USB-Treibers zurückgegeben. Die höheren zwei Bytes sind die höhere Versionsnummer (HIWORD = MajorNumber). Die niedrigeren zwei Bytes sind die niedrige Versionsnummer (LOWORD = MinorNumber).

Beispiel für die Ermittlung der Versionsnummer

In diesem Beispiel wird die Versionsnummer der Matrix-API ermittelt:

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
```

SetW95Access - Kommunikation mit dem LPT-Dongle unter Win95/98

Syntax

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

Unterstützt

LPT

Parameter

Mode Kann den Wert 1 oder 2 haben, oder die vordefinierten Werte:
IW_DRIVER / IW_NODRIVER

1 Die Kommunikation erfolgt über den VXD-Treiber.
2 Die Kommunikation erfolgt ohne VXD-Treiber.

Rückgabe

Diese Funktion hat keinen Rückgabewert.

GetPortAdr - Gibt die Adresse des LPT-Anschlusses zurück.

Syntax

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

Unterstützt

LPT

Parameter

LptNr Ist die Nummer des LPT-Anschlusses, z. B. 1 für LPT1, 2 für LPT2 oder 3 für LPT3. Die LPT-Nummer muss zwingend zwischen 1 und 3 liegen.

Rückgabe

Es wird die Adresse des LPT-Anschlusses zurückgegeben, z. B. 378 (Hex) für LPT1, oder 0 wenn der angegebene LPT-Port nicht vorhanden ist. Mit dieser Funktion kann die Anzahl der vorhandenen LPT-Ports in dem jeweiligen PC ermittelt werden.

Beispiel

In diesem Beispiel wird die Anzahl der vorhandenen LPT-Ports ermittelt:

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

PausePrinterActivity - Stops the Windows print-spooler. (nur Win 9.x/Win-NT/2000)

Syntax

```
short  PausePrinterActivity( );
```

Unterstützt

LPT

Rückgabe

0 Der Print-Spooler wurde angehalten.

-14 Der Print-Spooler konnte nicht angehalten werden.

Hinweis: Mit dieser Funktion kann der Windows Print-Spooler angehalten werden. So kann bei laufenden Druckaufträgen die LPT-Schnittstelle freigegeben werden, damit die Dongleabfrage durchgeführt werden kann. Anschließend, nach der Dongleabfrage, soll mit ResumePrinterActivity der Print-Spooler freigegeben werden, damit evtl. angehaltene Druckaufträge wieder fortgesetzt werden.

ResumePrinterActivity - Gibt den Windows Print Spooler wieder frei. (nur Win 9.x/Win-NT/2000)

Syntax

```
short  ResumePrinterActivity( );
```

Unterstützt

LPT

Rückgabe

0 Der Print-Spooler wurde freigegeben.

-14 Der Print-Spooler konnte nicht freigegeben werden.

Hinweis: Mit dieser Funktion wird der Windows Print-Spooler freigegeben. Somit werden evtl. angehaltene Druckaufträge wieder fortgesetzt.

Dongle_Find	- Sucht den Dongle und gibt den LPT-/USB-Port zurück, an dem dieser gefunden wurde.

Syntax

```
short  Dongle_Find( );
```

Unterstützt

LPT, USB, NET

Rückgabe

Es wird der Port zurückgegeben, an dem der Dongle gefunden wurde.
1 für LPT1, 2 für LPT 2, 3 für LPT3 oder 'U' (ASCII 85) für USB.

0	Kein Dongle an irgend einem LPT oder USB Anschluss gefunden.
-1	Es ist kein LPT-Port vorhanden.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-32	Die Server-Datei wurde nicht gefunden.
-33	Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34	Beim Zugriff auf die Server-Dateit ist ein Fehler aufgetreten.
-35	Das MxNet Serverprogramm ist inaktiv.

Hinweis: Diese Funktion liefert nur den ersten Anschluss, an dem ein Dongle gefunden wurde, unabhängig davon, ob noch weitere Dongle an anderen LPT oder USB-Ports vorhanden sind.

Verwenden Sie die Funktion *Dongle_FindEx* oder *Dongle_Count*, wenn Sie alle LPT-Ports nach der Existenz eines oder mehrerer Dongles untersuchen möchten.

Da die Funktion *Dongle_FindEx* den USB-Port nicht unterstützt, kann für USB-Dongles die Funktion *Dongle_Count* benutzt werden.

Dongle_FindEx - Sucht alle LPT-Ports und Dongle und legt diese Informationen in einem Datenpuffer ab.

Syntax

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

Unterstützt

LPT, NET

Parameter

xBuffer Ist ein Zeiger auf einen Datenpuffer, in dem die gelesenen Informationen abgelegt werden. Der Datenpuffer ist ein Array mit einer maximalen Größe von 3 und hat folgende Typenbeschreibung:

```
typedef struct {
   int LPT_Nr;    /* Nummer des LPT-Ports  */
   int LPT_Adr;   /* Adresse des LPT-Ports */
   int DNG_Cnt;   /* Anzahl der Dongles am LPT-Port
*/
} DNGINFO;
```

Rückgabe

Es wird die Anzahl der vorhandenen LPT-Ports zurückgegeben, oder:

0	Es ist kein LPT-Port vorhanden.
-5	Der LPT-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-32	Die Server-Datei wurde nicht gefunden.
-33	Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34	Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35	Das MxNet Serverprogramm ist inaktiv.

Hinweis: Da die Funktion Dongle_FindEx den USB-Port nicht unterstützt, kann für USB-Dongle die Funktion Dongle_Count benutzt werden.

Dongle_Count - **Gibt die Anzahl der vorhandenen Dongle an dem angegebenen LPT oder USB-Port zurück.**

Syntax

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

Unterstützt

LPT, USB, NET

Parameter

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Beispiel: 1 für LPT1, 2 für LPT2 oder 3 für LPT3. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.
Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe

Es wird die Anzahl der Dongle zurückgegeben, die an dem angegebenen Anschluss vorhanden sind. Die maximale Anzahl ist 99 für LPT-Dongle und 127 für USB.

0 Es ist kein Dongle an dem angegebenen Port vorhanden.
-1 Ein Kommunikationsfehler ist aufgetreten oder der mit ‚PortNr' angegebene Port wurde nicht gefunden.
-5 Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6 Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.
-29 keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).
-32 Die Server-Datei wurde nicht gefunden.
-33 Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34 Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35 Das MxNet Serverprogramm ist inaktiv.

Hinweis: Wenn an einem LPT-Port mehrere Dongle hintereinander gesteckt werden, dann werden diese vom PC aus zum Drucker rückwärts gezählt. Also bei z. B. drei Dongle, die hintereinander stecken, wäre der 3. Dongle direkt am PC und der 1. am Drucker.

Beispiel

In diesem Beispiel wird eine einfache Methode gezeigt, wie festgestellt werden kann, an welchem Port (LPT oder USB) der Dongle angeschlossen ist:

```
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
   /*--- Anzahl Dongles an dem ‚DNG_Port' Port ermitteln ---*/
   DNG_Count = Dongle_Count(DNG_Port);
   if(DNG_Count <= 0) continue;
   /*--- Den Dongle mit passendem UserCode herausfinden, ---*/
   /*--- falls mehrere Dongles vorhanden sind.           ---*/
   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;
}
/*-------------------------------------------------------*/
/* Ab jetzt ist bekannt, welcher der richtige Dongle ist:  */
/* ‚DNG_Nr' und ‚DNG_Port' identifizieren den Dongle.       */
/*-------------------------------------------------------*/
```

Dongle_MemSize - Gibt die Speichergröße des Dongle in Bytes zurück.

Syntax

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

Unterstützt

LPT, USB, NET

Parameter

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist bei LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.
Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe

Es wird die Speichergröße in Bytes zurückgegeben. z. B. wenn der ML-60 angeschlossen ist, dann ist der Rückgabewert 60. Bei einem ML-12 ist der Rückgabewert 12 usw.

0	Die Speichergröße konnte nicht gelesen werden.
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit ‚DngNr' angegebene Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26	Das USB-Gerät konnte nicht geöffnet werden.
-27	Das USB-Gerät ist ungültig.
-28	Das USB-Gerät konnte nicht konfiguriert werden.
-29	Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).
-32	Die Server-Datei wurde nicht gefunden.
-33	Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34	Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35	Das MxNet Serverprogramm ist inaktiv.

Hinweis: Mit diesem Rückgabewert kann auch die maximale Anzahl der Datenfelder im Dongle ermittelt werden.

Beispiel
Speichergröße und die maximale Anzahl der Datenfelder für den ersten Dongle an LPT2 ermitteln:

```
Speicher = Dongle_MemSize(1, 2)
MaxWerte = Speicher / 4
```

Ein Dongle mit einer Speichergröße von 60 Byte hat 15 Datenfelder (je 4 Byte).

Dongle_Model	**- Liest und gibt die Modellnummer des Matrix-Dongle zurück.**

Syntax

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

Unterstützt
LPT, USB, NET

Parameter

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert ‚U' (ASCII 85) enthalten.
Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe
Es wird die Modellnummer des Donglezurückgegeben, z. B. 257, 657, usw.

0 Die Modellnummer konnte nicht gelesen werden.
-1 Ein Kommunikationsfehler ist aufgetreten oder der mit ‚DngNr' angegebene

Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.

-5 Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.

-6 Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.

-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.

-26 Das USB-Gerät konnte nicht geöffnet werden.

-27 Das USB-Gerät ist ungültig.

-28 Das USB-Gerät konnte nicht konfiguriert werden.

-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

-32 Die Server-Datei wurde nicht gefunden.

-33 Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.

-34 Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.

-35 Das MxNet Serverprogramm ist inaktiv.

Hinweis: Die Versions- und die Modellnummer sollten Sie immer angeben, wenn Sie Fragen an unsere Hotline richten.

Dongle_Version - Liest und gibt die Versionsnummer des Matrix-Dongle zurück.

Syntax

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

Unterstützt

LPT, USB, NET

Parameter

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.
Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe

Es wird die Versionsnummer des Dongle zurückgegeben. Der Rückgabewert ist vom Typ »LONG« -> HIWORD = MajorNumber, LOWORD = MinorNumber.

0 Die Versionsnummer konnte nicht gelesen werden.
-1 Ein Kommunikationsfehler ist aufgetreten oder der mit ‚DngNr' angegebene Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.
-5 Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6 Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).
-32 Die Server-Datei wurde nicht gefunden.
-33 Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.

-34 Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35 Das MxNet Serverprogramm ist inaktiv.

Hinweis: Die Versions- und die Modellnummer sollten Sie immer angeben, wenn Sie Fragen an unsere Hotline richten.

Dongle_ ReadData - Liest die Daten ab Anfang des Speicherbereiches aus dem Matrix-Dongle.

Syntax

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

Unterstützt

LPT, USB, NET

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

Data Ist ein Zeiger auf einen Datenpuffer, in dem die gelesenen Werte abgelegt werden. Dieser Datenpuffer muss mindestens die Anzahl der Elemente haben, die in dem Parameter ‚Count' angegeben ist.

Count Ist die Anzahl der Werte, die aus dem Dongle gelesen werden sollen. Hier kann die maximale Anzahl der Werte angegeben werden (siehe Dongle_ MemSize).

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.
Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe
Ein Rückgabewert größer 0 ist die Anzahl der Datenfelder, die erfolgreich aus dem Dongle gelesen wurden.

0	Es wurden keine Daten aus dem Dongle gelesen.
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26	Das USB-Gerät konnte nicht geöffnet werden.
-27	Das USB-Gerät ist ungültig.
-28	Das USB-Gerät konnte nicht konfiguriert werden.
-29	Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).
-32	Die Server-Datei wurde nicht gefunden.
-33	Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34	Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35	Das MxNet Serverprogramm ist inaktiv.

Wichtig: Der Lesevorgang wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen. Verwenden Sie nicht Dongle_ReadData mehrmals hintereinander mit falschen und wechselnden »UserCodes«. Dadurch wird die »Anti-Hacker«-Sperre aktiviert. Der Dongle kann dann auch durch Angabe des richtigen »UserCodes« nicht mehr angesprochen werden. Die »Anti-Hacker«-Sperre kann nur mit einem MasterKey aufgehoben werden (siehe auch Dongle_ReadDataEx).

Dongle_ ReadDataEx - Liest die Daten ab einer bestimmten Speicheradresse aus dem Matrix-Dongle.

Syntax

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

Unterstützt

LPT, USB, NET

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

Data Ist ein Zeiger auf einen Datenpuffer, in dem die gelesenen Werte abgelegt werden. Dieser Datenpuffer muss mindestens die Anzahl der Elemente haben, die in dem Parameter »Count« angegeben ist.

Fpos Ist die Feldnummer, ab der gelesen werden soll.

Count Ist die Anzahl der Werte, die aus dem Dongle gelesen werden sollen. Hier kann die maximale Anzahl der Werte angegeben werden (siehe Dongle_MemSize).

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.
Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe
Ein Rückgabewert größer 0 ist die Anzahl der Datenfelder, die erfolgreich aus dem Dongle gelesen wurden.

0	Es wurden keine Daten aus dem Dongle gelesen.
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26	Das USB-Gerät konnte nicht geöffnet werden.
-27	Das USB-Gerät ist ungültig.
-28	Das USB-Gerät konnte nicht konfiguriert werden.
-29	Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).
-32	Die Server-Datei wurde nicht gefunden.
-33	Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34	Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35	Das MxNet Serverprogramm ist inaktiv.

Wichtig: Der Lesevorgang wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen. Verwenden Sie nicht Dongle_ReadDataEx mehrmals hintereinander mit falschen und wechselnden »UserCodes«. Dadurch wird die »Anti-Hacker«-Sperre aktiviert. Der Dongle kann dann auch durch Angabe des richtigen »UserCodes« nicht mehr angesprochen werden. Die »Anti-Hacker«-Sperre kann nur mit einem MasterKey aufgehoben werden.

Dongle_WriteData - Schreibt Daten in den Matrix-Dongle ab dem ersten Datenfeld.

Syntax

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

Unterstützt

LPT, USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

Data Ist ein Zeiger auf einen Datenpuffer mit den Werten, die im Dongle gespeichert werden sollen. Dieser Datenpuffer muss mindestens die Anzahl der Elemente haben, die in dem Parameter »Count« angegeben ist.

Count Ist die Anzahl der Werte, die Sie in den Dongle speichern wollen. Hier kann die maximale Anzahl der Werte angegeben werden (siehe Dongle_MemSize).

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.

Rückgabe
Ein Rückgabewert größer 0 ist die Anzahl jener Werte, die erfolgreich im Dongle gespeichert wurden.

0	Es wurden keine Daten in den Dongle gespeichert.
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-3	Der benötigte MasterKey-Dongle wurde nicht gefunden (nur bei der MK/MKU-Serie).
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26	Das USB-Gerät konnte nicht geöffnet werden.
-27	Das USB-Gerät ist ungültig.
-28	Das USB-Gerät konnte nicht konfiguriert werden.
-29	Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Wichtig: Der Schreibvorgang wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen. Verwenden Sie nicht Dongle_WriteData mehrmals hintereinander mit falschen und wechselnden »UserCodes«. Dadurch wird die »Anti-Hacker«-Sperre aktiviert. Der Dongle kann dann auch durch Angabe des richtigen »UserCodes« nicht mehr angesprochen werden. Die »Anti-Hacker«-Sperre kann nur noch mit einem MasterKey aufgehoben werden (siehe auch Dongle_WriteDataEx).

Dongle_WriteDataEx - Schreibt Daten in den Matrix-Dongle ab dem angegebenen Datenfeld.

Syntax

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

Unterstützt

LPT, USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

Data Ist ein Zeiger auf einen Datenpuffer mit den Werten, die im Dongle gespeichert werden sollen. Dieser Datenpuffer muss mindestens die Anzahl der Elemente haben, die in dem Parameter »Count« angegeben ist.

Fpos Ist die Feldnummer ab der gespeichert werden soll.

Count Ist die Anzahl der Werte, die Sie in den Dongle speichern wollen. Hier kann die maximale Anzahl der Werte angegeben werden (siehe Dongle_MemSize).

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.

Rückgabe
Ein Rückgabewert größer 0 ist die Anzahl jener Werte, die erfolgreich im Dongle gespeichert wurden.

0	Es wurden keine Daten in den Dongle gespeichert.
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle, an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-3	Der benötigte MasterKey-Dongle wurde nicht gefunden (nur bei der MK/MKU-Serie).
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26	Das USB-Gerät konnte nicht geöffnet werden.
-27	Das USB-Gerät ist ungültig.
-28	Das USB-Gerät konnte nicht konfiguriert werden.
-29	Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Wichtig: Der Schreibvorgang wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen. Verwenden Sie nicht Dongle_WriteDataEx mehrmals hintereinander mit falschen und wechselnden »UserCodes«. Dadurch wird die »Anti-Hacker«-Sperre aktiviert. Der Dongle kann dann auch durch Angabe des richtigen »UserCodes« nicht mehr angesprochen werden. Die »Anti-Hacker«-Sperre kann nur noch mit einem MasterKey aufgehoben werden.

Dongle_ReadSerNr - Liest und gibt die eindeutige Seriennummer des Dongle zurück.

Syntax

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

Unterstützt

LPT, USB, NET

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten. Beim Netzwerkzugriff hat dieser Parameter keine Bedeutung.

Rückgabe

Es wird die Seriennummer des Dongle zurückgegeben.

0	In diesem Dongle ist keine Seriennummer vorhanden.
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit ‚DngNr' angegebene Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.
-2	Der angegebene ‚UserCode' entspricht nicht dem UserCode aus dem Dongle.
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-7	Der Dongle unterstützt diese Funktion nicht. (Seriennummer wird nur ab Hardwareversion 2.3 unterstützt. Die Modelle ML-12/MK-12 unterstützen grundsätzlich keine Seriennummer).
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.

-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).
-32 Die Server-Datei wurde nicht gefunden.
-33 Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34 Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35 Das MxNet Serverprogramm ist inaktiv.

Wichtig: Der Lesevorgang wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen.

Dongle_WriteKey - 128 Bit XTEA-Schlüssel in den Dongle speichern.

Syntax

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

Unterstützt

LPT, USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

KeyData Ist ein Zeiger auf einen Datenpuffer mit den Key-Werten, die im Dongle gespeichert werden sollen.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.

Rückgabe

Bei erfolgreicher Speicherung muss die Funktion einen Wert größer 0 zurückgeben.

0 Es wurden keine Daten in den Dongle gespeichert.
-1 Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2 Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-3 Der benötigte MasterKey-Dongle wurde nicht gefunden (nur bei der MK/MKU-Serie).
-4 Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5 Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6 Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-7 Der Dongle unterstützt diese Funktion nicht. (Kryptographie wird nur ab

Hardwareversion 2.1 unterstützt. Die Modelle ML-12/MK-12 unterstützen grundsätzlich keine Kryptographie).

-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Wichtig: Die Speicherung wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen.

Dongle_GetKeyFlag - Prüft, ob ein 128 Bit XTEA-Schlüssel ungleich Null im Dongle vorhanden ist.

Syntax

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

Unterstützt

LPT, USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.

Rückgabe

Es wird 1 zurückgegeben, falls im Dongle ein Schlüssel ungleich Null vorhanden ist, oder 0, falls der Schlüssel Null ist (alle Schlüsselbytes = 0).

1	128-Bit Schlüssel ist im Dongle vorhanden
0	128-Bit Schlüssel ist im Dongle nicht vorhanden (Zero-Key)
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6	Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.
-7	Der Dongle unterstützt diese Funktion nicht. Kryptographie wird nur ab Hardwareversion 2.1 unterstützt. Die Modelle ML-12/MK-12 unterstützen grundsätzlich keine Kryptographie.

-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Hinweis: Da der 128-Bit Schlüssel nicht aus dem Dongle ausgelesen werden kann, bietet diese Funktion die Möglichkeit zur Überprüfung, ob ein 128-Bit Schlüssel im Dongle vorhanden ist. Ein »Zero-Key«, also mit allen Bytes=0 wird nicht als gültiger Schlüssel interpretiert. Der Lesevorgang wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen.

Dongle_EncryptData - 8 Bytes grossen Datenblock an Dongle senden. Dieser wird XTEA-verschlüsselt zurück gegeben.

Syntax

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

Unterstützt

LPT, USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

DataBlock Ist ein Zeiger auf einen 8 Bytes grossen Datenpuffer, der mit XTEA verschlüsselt werden soll.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.

Rückgabe

Bei erfolgreicher Verschlüsseleung wird ein Wert größer 0 zurückgeben.

-1 Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2 Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-4 Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5 Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6 Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.

-7 Der Dongle unterstützt diese Funktion nicht. Kryptographie wird nur ab Hardwareversion 2.1 unterstützt. Die Modelle ML-12/MK-12 unterstützen grundsätzlich keine Kryptographie.
-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Wichtig: Die Verschlüsselung wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen.

Dongle_DecryptData - Verschlüsselten 8 Bytes grossen Datenblock an Dongle senden, um diesen zu entschlüsseln.

Syntax

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

Unterstützt

LPT, USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

DataBlock Ist ein Zeiger auf einen 8 Bytes grossen Datenpuffer, der mit XTEA entschlüsselt werden soll.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 99 Dongle hintereinander an den LPT-Port und maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist beim LPT-Dongle die Nummer des LPT-Ports, an dem die Dongle angeschlossen sind. Die Port-Nummer muss beim LPT-Dongle zwischen 1 und 3 liegen.
Für USB muss dieser Parameter den Wert »U« (ASCII 85) enthalten.

Rückgabe

Bei erfolgreicher Entschlüsseleung wird ein Wert größer 0 zurückgeben.

-1 Ein Kommunikationsfehler ist aufgetreten oder der mit ‚DngNr' angegebene Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.
-2 Der angegebene ‚UserCode' entspricht nicht dem UserCode aus dem Dongle.
-4 Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5 Der LPT/USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-6 Beim Zugriff auf den LPT-Port ist ein Fehler aufgetreten.

-7 Der Dongle unterstützt diese Funktion nicht. Kryptographie wird nur ab Hardwareversion 2.1 unterstützt. Die Modelle ML-12/MK-12 unterstützen grundsätzlich keine Kryptographie.
-25 Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Wichtig: Die Entschlüsselung wird nur dann durchgeführt, wenn der von Ihnen übergebene »UserCode« und der »UserCode« aus dem Dongle übereinstimmen.

Dongle_SetDriverFlag - Ändert die USB-Betriebsart des Dongles auf »HID-Mode« oder »Driver-Mode«.

Syntax

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

Unterstützt

USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

Mode Kann den Wert 0 oder 1 haben:
0 - »Driver-Mode« - Betriebsart mit eigenem USB-Treiber.
1 - »HID-Mode« - Betriebsart ohne eigenen USB-Treiber.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist der Ports, an dem der Dongle angeschlossen ist. Dieser Parameter muss den Wert »U« (ASCII 85) enthalten.

Rückgabe

Bei erfolgreicher Änderung der Betriebsart wird der Wert 1 zurückgeben.

1	Die USB-Betriebsart wurde erfolgreich geändert
0	Die in »Mode« angegebene Betriebsart ist ungültig
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit ‚PortNr' angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-7	Der Dongle unterstützt diese Funktion nicht. (Unterschiedliche USB-Betriebsarten sind nur ab Hardwareversion 5.0 verfügbar).
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.

-26 Das USB-Gerät konnte nicht geöffnet werden.
-27 Das USB-Gerät ist ungültig.
-28 Das USB-Gerät konnte nicht konfiguriert werden.
-29 Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

Hinweis: Die eingestellte USB-Betriebsart wird vom System erst nach erneutem Anschliessen des Dongles erkannt.

Dongle_GetDriverFlag - Die eingestellte USB-Betriebsart des Dongle ermitteln: »HID-Mode« oder »Driver-Mode«.

Syntax

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

Unterstützt

USB

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.
Es können maximal 127 Dongle an den USB-Anschluss gesteckt werden.

PortNr Ist der Ports, an dem der Dongle angeschlossen ist. Dieser Parameter muss den Wert »U« (ASCII 85) enthalten.

Rückgabe

Es wird die eingestellte USB-Betriebsart des Dongle zurückgegeben.

1	Die Betriebsart des Dongles ist »HID-Mode« - ohne eigenen USB-Treiber
0	Die Betriebsart des Dongles ist »Driver-Mode« - mit eigenem USB-Treiber
-1	Ein Kommunikationsfehler ist aufgetreten oder der mit »DngNr« angegebene Dongle an dem mit »PortNr« angegebenen Port wurde nicht gefunden.
-2	Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-4	Der Dongle ist gesperrt; die »Anti-Hacker«-Sperre ist aktiv.
-5	Der USB-Port kann nicht belegt werden, da dieser bereits von anderen Geräten belegt ist.
-7	Der Dongle unterstützt diese Funktion nicht. Unterschiedliche USB-Betriebsarten sind nur ab Hardwareversion 5.0 verfügbar.
-25	Die Liste der USB-Geräte konnte nicht erzeugt werden.
-26	Das USB-Gerät konnte nicht geöffnet werden.
-27	Das USB-Gerät ist ungültig.
-28	Das USB-Gerät konnte nicht konfiguriert werden.
-29	Keine USB-Unterstützung in diesem Betriebssystem (z.B. Win-NT).

SetConfig_MatrixNet - Aktiviert oder deaktiviert den Netzwerkzugriff.

Syntax

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

Unterstützt

NET

Parameter

nAccess Kann den Wert 0 oder 1 haben:
0 - Der Dongle ist lokal angeschlossen.
1 - Aktiviert den MxNET-Netzwerkzugriff.

nFile In dieser Variable wird der Name der Server-Datei für den MxNet-Netzwerkzugriff angegeben. Die Server-Datei muss mit absolutem Pfad angegeben werden (z. B. \\servername\TEMP\MXFILE.NET).
Diese Variable wird ignoriert (kann den Wert NULL annehmen), falls der Zugriff für einen lokal angeschlossenen Dongle eingestellt wird mit nAccess = 0.

Rückgabe

0 Wird zurückgegeben, wenn der Netzwerk-Zugriff deaktiviert wurde (lokaler Zugriff).

1 Wird zurückgegeben, wenn der Netzwerk-Zugriff aktiviert wurde.

Hinweis: Diese Funktion muss immer vor den Funktionen LogIn_MatrixNet und LogOut_MatrixNet aufgerufen werden. Die Einstellung für den Netzwerk-Zugriff kann durch einen erneuten Aufruf der Funktion mit dem Wert nAccess = 0 deaktiviert werden. Wird der Netzwerk-Zugriff deaktiviert, dann greift die API auf einen lokal angeschlossenen Dongle zu.

GetConfig_MatrixNet - Liest aus der Server-Datei Parameter, die im MxNet-Serverprogramm eingestellt wurden

Syntax

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

Unterstützt

NET

Parameter

Category Diese Variable gibt an, welcher Parameter aus der Server-Datei gelesen werden soll. Mögliche Werte sind:

0 – die Versionsnummer der Server-Datei soll zurückgegeben werden.
Der Rückgabewert ist vom Typ Long mit dem Format:

```
HIWORD = MajorNumber
LOWORD = MinorNumber
```

1 – Die im MxNet-Serverprogramm eingestellte Anzahl Minuten für die Aktualisierung der Server-Datei (Refresh-Time) soll zurückgegeben werden.

2 – Die Anzahl der Minuten, die seit der letzten Aktualisierung (Refresh) der Server-Datei vergangen sind, soll zurückgegeben werden.

3 – Die im MxNet-Serverprogramm eingestellte Anzahl Minuten für den User-TimeOut soll zurückgegeben werden.

Rückgabe

Ein Rückgabewert größer oder gleich 0 ist der angeforderte Parameter.

-1	Die angeforderte Kategorie konnte nicht gelesen werden, oder der Netzwerk-Zugriff wurde noch nicht aktiviert (siehe SetConfig_MatrixNet).
-7	Die angegebene Kategorie ist unbekannt.
-32	Die Server-Datei wurde nicht gefunden.
-33	Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34	Beim Zugriff auf die Server-Datei ist ein Fehler aufgetreten.
-35	Das MxNet Serverprogramm ist inaktiv.

Hinweis: Die mit dieser Funktion angeforderten Parameter werden direkt aus der Server-Datei gelesen. Diese Parameter werden vom MxNet-Serverprogramm bei jedem Refresh der Server-Datei aktualisiert.

LogIn_MatrixNet - Meldet den Netzwerk-Client an und belegt/ aktualisiert den UserSlot in der Server-Datei.

Syntax

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

Unterstützt
NET

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

AppSlot Ist die Nummer jener Variable im Dongle, die die Anzahl der Nezwerk-Lizenzen beinhaltet (»Application-Slot«).

DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.

Rückgabe
Ein Rückgabewert größer oder gleich 0 wird zurückgegeben, wenn der LogIn erfolgreich durchgeführt wurde; zudem repräsentiert er dann die Anzahl der noch freien UserSlots.

-1 Der mit »DngNr« angegebene Dongle oder die mit »AppSlot« angegebene Feldnummer (»Application-Slot«) wurde nicht gefunden.
-2 Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-31 Es ist kein UserSlot mehr frei, der LogIn wurde nicht durchgeführt.
-32 Die Server-Datei wurde nicht gefunden.
-33 Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34 Beim Zugriff auf die Server-Datei-Port ist ein Fehler aufgetreten.
-35 Das MxNet Serverprogramm ist inaktiv.

Wichtig: Ihre Anwendung muss von Zeit zur Zeit deswegen die Funktion aufrufen, damit der UserSlot-Eintrag aktualisiert wird, bevor das TimeOut-Limit erreicht wird. (siehe »Einstellungen des MxNet-Server-Programms«).

LogOut_MatrixNet - Meldet den Netzwerk-Client ab und gibt den UserSlot in der Server-Datei wieder frei.

Syntax

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

Unterstützt

NET

Parameter

UserCode Ist der Kundencode. Die in UserCode übergebene Kundennummer muss mit dem internen UserCode aus dem Dongle übereinstimmen.

AppSlot Ist die Nummer jener Variable im Dongle, die die Anzahl der Nezwerk-Lizenzen beinhaltet (»Application-Slot«).

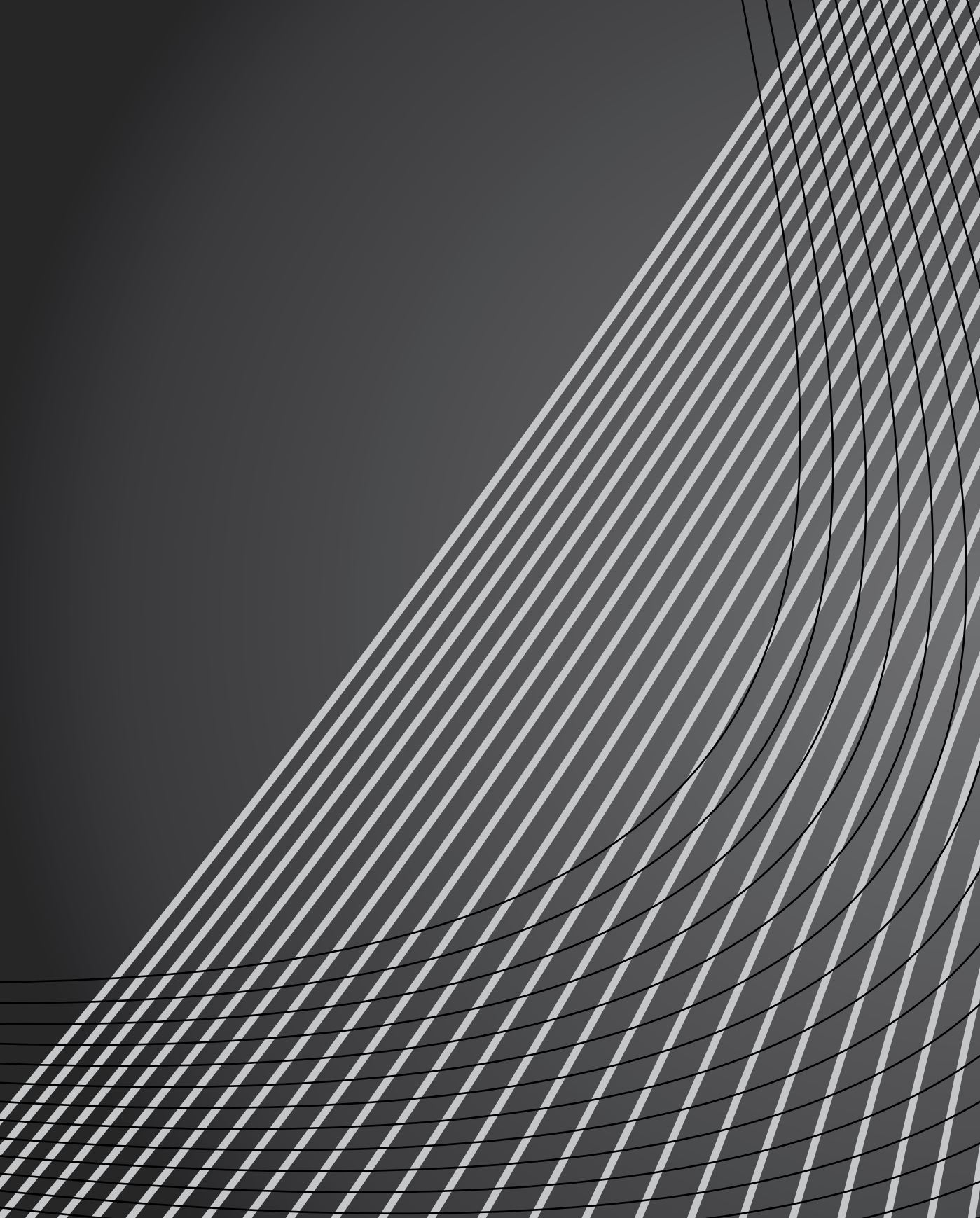
DongleNr Ist die Dongle-Nummer, wenn mehrere Dongle hintereinander gesteckt sind. Die maximale Anzahl der vorhandenen Dongle kann mit der Funktion Dongle_Count ermittelt werden.

Rückgabe

Ein Rückgabewert größer oder gleich o wird zurückgegeben, wenn der LogOut erfolgreich durchgeführt wurde und repräsentiert die Anzahl der noch freien UserSlots.

-1 Der mit »DngNr« angegebene Dongle oder die mit »AppSlot« angegebene Feldnummer (»Application-Slot«) wurde nicht gefunden.
-2 Der angegebene »UserCode« entspricht nicht dem UserCode aus dem Dongle.
-31 Für diesen User wurde keinen Eintrag in der ServerDatei gefunden. Der LogOut wurde nicht durchgeführt.
-32 Die Server-Datei wurde nicht gefunden.
-33 Die Server-Datei kann nicht belegt werden, da diese gerade von einer anderen Anwendung belegt ist.
-34 Beim Zugriff auf die Server-Datei-Port ist ein Fehler aufgetreten.

Wichtig: Es sollte immer ein LogOut durchgeführt werden, bevor die Anwendung beendet wird, um den UserSlot wieder frei zu geben.



7

Sonstiges

Sonstiges

Technische Daten

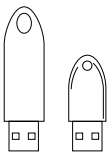
Anschluss	LPT/USB
Prozessor	RISC Auch einzeln für eigene Hardwareentwicklung lieferbar
Codierung	Fester, unveränderbarer Kundencode
Verfügbarer Speicher	Modellbezogen: LPT: ML-12 –> 12 Byte = 3 Variablen (3 x 32-Bit) ML-60 –> 60 Byte = 15 Variablen (15 x 32-Bit) ML-316 –> 316 Byte = 79 Variablen (79 x 32-Bit) USB: MLU-60 –> 60 Byte = 15 Variablen (15 x 32-Bit) MLU-316 –> 316 Byte = 79 Variablen (79 x 32-Bit) Größerer Speicher auf Anfrage
Datensicherheit	> 40 Jahre
Lese/Schreibzyklen	1,000,000 Zyklen garantiert
Anreihung	LPT: max. 99 Dongle hintereinander mit gezielter Adressierung jedes einzelnen Dongle im Stapel USB: max. 99 Dongle hintereinander mit gezielter Adressierung jedes einzelnen Dongle
Kommunikation	Verschlüsselt - Dongle <–> PC
Betriebssysteme	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	Programmiersoftware und Treiber im Lieferumfang für Windows 3.x, 95, 98, ME, NT, 2000, XP/Vista, Linux and Mac-OS X
Spannungsversorgung	Über den LPT-Anschluss - zwischen 2,5V und 5,5V Über den USB-Anschluss - zwischen 3,3V und 5,5V
Stromverbrauch	Ruhezustand: < 950µA Daten lesen: < 1,7 mA Daten schreiben: < 2,6 mA
Abmessungen	LPT: 43 x 54 x 14 mm (± 0,5 mm) USB: 55 x 16 x 8 mm (± 0,5 mm)
Sonstiges	Sonderlösungen auf Anfrage

Preise und Lieferkonditionen

Modell	Speicher	Crypt / Decrypt	Mengenstaffel (Euro)								
			1+	10+	100+	300+	500+	1K+	3K+	5K+	10K+
ML-12 LPT	12 Bytes (3 Datenfelder)	✔	28,-	18,-	16,-	15,-	14,-	13,-	12,-	11,-	10,50
ML-60 LPT	60 Bytes (15 Datenfelder)	✔	31,-	20,-	18,-	17,-	16,-	15,-	14,-	13,-	11,50
ML-316 LPT	316 Bytes (79 Datenfelder)	✔	32,-	22,-	20,-	19,-	18,-	17,-	16,-	15,-	13,50
MLU-60 USB	60 Bytes (15 Datenfelder)	✔	34,-	24,-	22,-	21,-	20,-	17,-	14,-	11,-	9,-
MLU-316 USB	316 Bytes (79 Datenfelder)	✔	36,-	26,-	24,-	23,-	21,-	18,-	15,-	12,-	9,50
MLU Piccolo USB*	12 Bytes (3 Datenfelder)	✔	Bulk - Mindestmenge 10K								5,87

Matrix Dongle mit größerem Speicher sind auf Anfrage erhältlich. Gehäusefarben und Logo-Imprints auf Anfrage.
Auf die Funktionalität werden 10 Jahre Garantie gewährleistet.
Bei fester Abnahmemenge können die Preise im Rahmen eines Vertrags vergünstigt werden.
Alle Preise sind in EUR zzgl. MwSt. und Versandkosten.

* MLU Piccolo - nur in Mindestmengen von 10K erhältlich (Bulk). Gehäuse-Farbe und Logo sind frei wählbar.



Alle USB-Dongle sind in zwei Gehäuse-Grössen verfügbar: Long und Short. Beide sind technisch und preislich gleich.

Alle Funktionen inklusive Netzwerkfähigkeit sowie sämtliche Software, Kundenanpassungen, Weiterentwicklungen und Support sind im Preis enthalten.

Für die MK/MKU-Serie gelten die Basispreise der ML-Reihe zzgl. EUR 1,50 pro Stück. Die Programmierung der Daten ist bei der MK-Serie nur mit dem entsprechenden MasterKey möglich. Der MasterKey-Dongle ist bei der ersten Bestellung im Lieferumfang enthalten (nur MK/MKU-Anwender).
Die Besteller der ML/MLU-Serie können optional einen MasterKey erwerben.

Allgemeine Geschäftsbedingungen

1. Allgemeines

a) Allen Lieferungen und Leistungen liegen ausschließlich die nachfolgenden Geschäftsbedingungen zugrunde. Abweichende Vereinbarungen bedürfen der Schriftform. Allen auch künftigen Liefergeschäften, liegen diese Geschäftsbedingungen zugrunde.
b) Sämtliche Angaben, hinsichtlich der von uns vertriebenen Geräte in Produktbeschreibungen, Prospekten o. ä. sind stets freibleibend, soweit sie nicht ausdrücklich verbindlich zugesichert sind. Dies gilt insbesondere für Änderungen, die dem technischen Fortschritt oder dem Erhalt der Lieferfähigkeit dienen.
c) Unsere Preise verstehen sich, soweit nicht gesondert schriftlich vereinbart, ohne gesondertes Zubehör, Aufrüstungen, Installation, Schulungen und sonstigen Nebenleistungen. Die Bestellung des Kunden kann innerhalb von 14 Tagen durch Zusendung der Auftragsbestätigung oder durch Auslieferung und Rechnungserteilung angenommen werden.

2. Liefertermine, Verzug, Gefahrenübergang

a) Verbindliche Liefertermine müssen schriftlich vereinbart werden. Sie setzen die Klärung von technischen Fragen sowie die Selbstbelieferung voraus. Geraten wir aus Gründen, die wir zu vertreten haben, in Lieferverzug, ist der Besteller berechtigt, entsprechend den gesetzlichen Bestimmungen vom Vertrag zurückzutreten, nachdem er erfolglos eine schriftliche Nachfrist von mindestens zwei Wochen gesetzt hat.
b) Die Haftung für gewöhnliche Fahrlässigkeit ist ausgeschlossen.
c) Wir sind zu Teillieferungen berechtigt. Soweit eine Versendung der Ware erfolgt, geschieht dies nach unserer freien Wahl, wobei die Übernahme von Fracht und Versendung durch uns nur bei besonderer schriftlicher Vereinbarung erfolgt. Wir sind berechtigt, jedoch nicht verpflichtet, die Ware auf Kosten des Bestellers zu versichern.

3. Zahlungsbedingungen, Abnahmeverzug

a) Unsere Rechnungen sind vorbehaltlich einer anderweitigen schriftlichen Vereinbarung sofort fällig und netto ohne jeden Abzug zu zahlen. Kommt der Käufer mit der Zahlung in Verzug, sind Verzugszinsen von 4% über dem Diskontsatz der Deutschen Bundesbank, mindestens jedoch 9% zu zahlen.
b) Nimmt der Käufer die verkaufte Ware nicht ab, können wir wahlweise auf Abnahme bestehen oder 25% der Kaufsumme als Schadensersatz verlangen, wobei der Nachweis, dass kein Schaden oder ein geringerer Schaden entstanden ist, dem Besteller verbleibt. Die Aufrechnung ist ausgeschlossen, außer die Gegenansprüche sind rechtskräftig festgestellt, unbestritten oder von uns anerkannt.

4. Verlängerter Eigentumsvorbehalt

Die gelieferten Waren bleiben bis zur Begleichung aller Verbindlichkeiten, bei Zahlung per Scheck oder Wechsel bis zu deren Einlösung, Eigentum der Firma TDi GmbH. Der Besteller ist berechtigt, die Ware im ordentlichen Geschäftsgang weiter zu veräußern, er tritt jedoch bereits jetzt alle Forderungen gegenüber seinen Abnehmern oder Dritten aus der Weiterveräußerung in Höhe des Faktura- Endbetrages (Einschließlich Mehrwertsteuer) ab. Im Falle vertragswidrigen Verhaltens des Bestellers, insbesondere bei Zahlungsverzug, sind wir berechtigt, die Kaufsache zurückzunehmen. In der Rücknahme ist jedoch kein Rücktritt vom Vertrag zu sehen, es sei denn, wir hätten dies ausdrücklich schriftlich erklärt.

5. Gefahrenübergang

Mit der Übergabe der Ware an den Besteller oder dessen Beauftragten, bei Versendung mit Übergabe an die Transportperson, geht die Gefahr auf den Käufer über, unabhängig von der Tatsache, wer die Transportkosten trägt.

6. Gewährleistung, Haftungsbeschränkung

a) Soweit ein von uns zu vertretender Mangel der Kaufsache vorliegt, sind wir nach unserer Wahl zur Mängelbeseitigung oder Ersatzleistung berechtigt. Offensichtliche Mängel sind innerhalb von 14 Tagen nach Erhalt der Ware zu rügen.
b) Die im kaufmännischen Rechtsverkehr geltenden §§ 377, 378 HGB bleiben unberührt. Erfolgt innerhalb von 10 Werktagen nach Eintreffen der Ware am Bestimmungsort keine Rüge, gilt die Ware als genehmigt.
c) Der Anspruch des Bestellers auf Gewährleistung erlischt bei Eingriffen, Reparaturen oder Reparaturversuchen des Käufers oder nicht autorisierter Dritter. Die Abtretung von Gewährleistungsansprüchen ist ausgeschlossen. Ersetzte Teile gehen in unser Eigentum über. Für einen erfolgten Austausch oder Reparaturen wird in gleicher Weise gewährleistet.
d) Sind wir zu einer Ersatzlieferung nicht bereit oder nicht in der Lage oder schlägt die Mängelbeseitigung mindestens dreimal fehl, ist der Besteller nach seiner Wahl berechtigt, vom Vertrag zurückzutreten oder eine entsprechende Minderung des Kaufpreises zu verlangen.
e) Es wird ausdrücklich festgestellt, dass aufgrund der Vielzahl unterschiedlichster Konfigurationen und der ständigen Weiterentwicklungen bei Druckern, Scannern und sonstigen Peripheriegeräten eine einwandfreie Funktion der Matrixmodule nicht in jedem Einzelfall zugesichert werden kann. Die mitgelieferte Software und die Datenträger werden auf Computerviren überprüft, eine Haftung für nicht entdeckte Vieren und für den dadurch entstandenen Schaden wird nicht übernommen.
f) Soweit sich nichts anderes ergibt, sind weitergehende Ansprüche des Bestellers - gleich, aus welchen Rechtsgründen, - ausgeschlossen. Für Schäden, die nicht im Liefergegenstand selbst entstanden sind, wird nicht gehaftet, insbesondere wird nicht für entgangenen Gewinn oder sonstige Vermögensschäden des Bestellers gehaftet. Vorstehende Haftungsbefreiung gilt nicht, wenn der Schaden durch Vorsatz, grobe Fahrlässigkeit oder Fehlens einer zugesicherten Eigenschaft, Verletzung vertragswesentlicher Pflichten, Leistungsverzug, Unmöglichkeit sowie Ansprüchen nach §§ 1,4 Produkthaftungsgesetz beruht. Soweit unsere Haftung ausgeschlossen oder beschränkt ist, gilt dies auch für unsere Angestellten, Mitarbeiter, Arbeitnehmer, Vertreter und Erfüllungsgehilfen.

7. Erfüllungsort

Sofern sich aus der Auftragsbestätigung nichts anderes ergibt, ist unser Geschäftssitz Erfüllungsort für Zahlung und Lieferung.

8. Gerichtsstand

Sofern der Besteller Vollkaufmann ist, ist unser Geschäftssitz Gerichtsstand.
TDi GmbH ist jedoch berechtigt, den Besteller auch an dem für seinen Wohnsitz zuständigen Gericht zu verklagen.

WWW.TDI-MATRIX.DE