Datamatrix Encode&Decode SDK v2.5

USER MANUAL

AIPSYS Software Laboratory

http://www.aipsys.com

Last Updated 21st March 2013

1. Introduction

Datamatrix Barcode introduction

About Datamatrix

This code is part of 2 dimentional code family, it can encode up to 2335 characters on a very small surface. The encoding is done in two stages : first the datas are converted to 8 bits "codeword" (High level encoding) then those are converted to small black and white squares. (Low level encoding) Moreover an error correction system is included, it allows to reconstitute badly printed, erased, fuzzy or torn off datas. In the continuation of this talk, the word "codeword" will be shortened into CW.

Symbol Structure

The symbol is a square or rectangular array made with rows and columns. Each cell is a small square black for a bit set to 1 and white for a bit set to 0. The dimension of the square is named the module. The colors can be inverted : white on black.

Extended Channel Interpretation (ECI) protocol provides a method to specify particular interpretations on byte values or to identify a particular page code.

The default ECI code is 000003 which designate the Latin alphabet ISO 8859-1.

There are two datamatrix standard : ECC 000-140 and ECC 200. Only ECC 200 can be used for a new project. This study is only dedicated to the ECC 200.

A symbol consists of one or several data regions. Each region has a one module wide perimeter. Independently of the number of region, there is one, and only one, mapping matrix. Le size of the matrix is : "region size" x "number of region"

Example for the 36x36 symbol : "16x16" x "2x2" ---> matrix size is 32x32

Second example for the 16x48 symbol : "14x22" x "1x2" ---> matrix size is 14x44

The number of rows and the number of columns (including the perimeter) are always even (Odd for ECC 000-140 !)

If necessary a mechanism allows to distribute more datas on several symbols. (Up to 16)

The error correction mechanism is based on Reed-Solomon codes.

For square symbol of 48 x 48 and less, Reed-Solomon codes are append after the datas; for other symbols they are interleaved : datas are divided in blocks.

Each symbol size has its own number of Reed-Solomon code.

The total number of CW per symbol is equal to the number of cells in the matrix divided by 8 (Without decimal part)

The 8 bits of each CW are placed in the region in order from left to right and top to bottom; certain CW are split in order to fill the matrix.

A quiet zone from 1 module (minimum) is required on the 4 sides.

Low level encoding

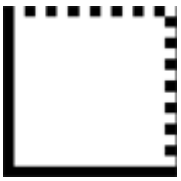
Thereafter we'll use operators : + --> addition, x --> multiplication, \ --> integer division, MOD --> remainder of the integer division.

There are 24 sizes of square symbol and 6 sizes of rectangular symbol. The following array give basic values for each symbol size.

Symbol size Rows x columns	Number of data region (H x V)	Number of Reed Solomon CW	Number of block
Square symbols			
10x10	1	5	1
12x12	1	7	1
14x14	1	10	1
16x16	1	12	1
18x18	1	14	1
20x20	1	18	1
22x22	1	20	1
24x24	1	24	1
26x26	1	28	1
32x32	2x2	36	1
36x36	2x2	42	1
40x40	2x2	48	1
44x44	2x2	56	1
48x48	2x2	68	1
52x52	2x2	2 x 42	2
64x64	4x4	2 x 56	2
72x72	4x4	4 x 36	4
80x80	4x4	4 x 48	4
88x88	4x4	4 x 56	4
96x96	4x4	4 x 68	4
104x104	4x4	6 x 56	6
120x120	6x6	6 x 68	6
132x132	6x6	8 x 62	8
144x144	6x6	10 x 62	8
Rectangular symbols			
8x18	1	7	1
8x32	2	11	1
12x26	1	14	1
12x36	1x2	18	1

16x36	1x2	24	1
16x48	1x2	28	1

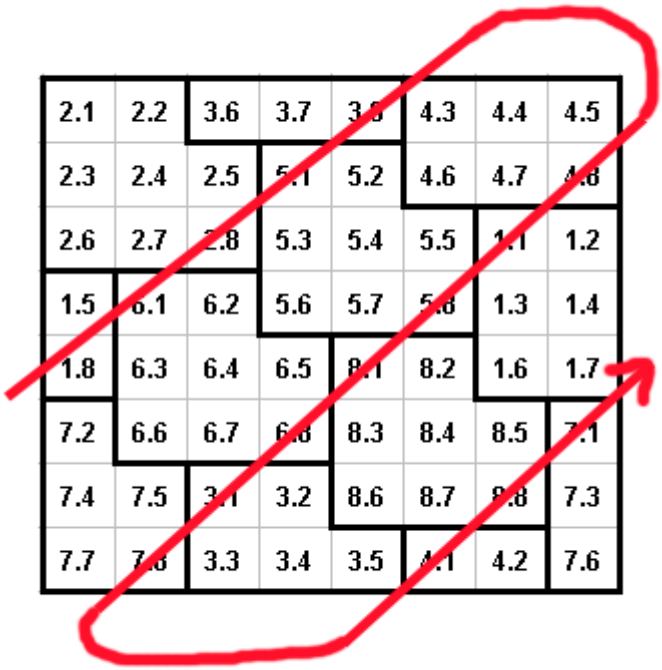
Each region has a one module wide perimeter. Left and lower sides are entirely black, right and top sides are made up of alternating black and white squares.



Each CW is placed in the matrix (If there are several regions, they are assembled to form an unique matrix) on 45 degree parallel diagonal lines and the left top corner is always as shown below

2.1	2.2	3.6	3.7	3.8	4.3	4.4	4.5
2.3	2.4	2.5	5.1	5.2	4.6	4.7	4.8
2.6	2.7	2.8	5.3	5.4	5.5		
1.x	6.1	6.2	5.6	5.7	5.8		
1.y	6.3	6.4	6.5				
	6.6	6.7	6.8				

In this image, we can remark than CW nr. 2, 5 and 6 have a regular shape. CW nr. 1, 3, 4 are truncated and the remain of these CW is reported on the other side of the symbol. Here is the entire placement of the 8 x 8 matrix :



You can remark on this image that the bit 8 of each CW is under the 45 degree parallel diagonal lines. Corner and border conditions are very intricate and different for each matrix size, fortunately Datamatrix standard give us an algorithm in order to make the placement.

High level encoding.

The hight level encoding support 6 compaction mode, ASCII mode is divided in 3 sub-mode :

Compaction mode	*Datas to encode*	*Rate compaction*
ASCII	ASCII character 0 to 127	1 byte per CW
ASCII extended	ASCII character 128 to 255	0.5 byte per CW
ASCII numeric	ASCII digits	2 byte per CW
C40	Upper-case alphanumeric	1.5 byte per CW
TEXT	Lower-case alphanumeric	1.5 byte per CW
X12	ANSI X12	1.5 byte per CW
EDIFACT	ASCII character 32 to 94	1.33 bytet per CW
BASE 256	ASCII character 0 to 255	1 byte per CW

The default character encodation method is ASCII. Some special CWs allow to switch between the encoding methods

Codeword	*Data or function*
1 to 128	ASCII datas
129	Padding
130 to 229	Pair of digits : 00 to 999
230	Switch to C40 method
231	Switch to Base 256 method
232	FNC1 character
233	Structure of several symbols
234	Reader programming
235	Shift to extended ASCII for one character
236	Macro
237	Macro
238	Switch to ANSI X12 method
239	Switch to TEXT method
240	Switch to EDIFACT method
241	Extended Channel Interpretation character
254	If ASCII method is in force : End of datas, next CWs are pads CW If other method is in force : Switch back to ASCII method or indicate end of datas

If the symbol is not full, pad CWs are required. After the last data CW, the 254 CW indicates the end of the datas or the return to ASCII method. First padding CW is 129 and next padding CWs are computed with the 253-state algorithm.

The ASCII mode. This mode has 3 ways to encode character :

- ASCII character in the range 0 to 127
  CW = "ASCII value" + 1
- Extended ASCII character in the range 128 to 255
  A first CW with the value 235 and a second CW with the value : "ASCII value" - 127
- Pair of digits 00, 01, 02 99
  CW = "Pair of digits numerical value" + 130

C40, TEXT and X12 modes

C40 and TEXT modes are similar : only uppercase and lowercase characters are inverted.

In these modes 3 data characters are compacted in 2 CWs. In C40 and TEXT modes 3 shift characters allow to indicate an other character set for the next character.

The 16 bits value of a CW pair is computed as following :

Value = C1 * 1600 + C2 * 40 + C3 + 1 with C1, C2 and C3 the 3 character values to compact.

254 CW indicate a return to the ASCII method exept if this mode allows to fill completely the symbol.

In C40 and TEXT mode a pad character with 0 value can be added at the 2 last characters in order to form a pair of CW.

If it remains to encode only one character in C40 or TEXT mode or 2 character in X12 mode; it(they) must be encoded with ASCII method but if a single free CW remain in the symbol before data correction CWs, it is assumed that this CW is encoded using ASCII method without using the 254 CW.

"Upper Shift" character enable to encode extended ASCII character..

Extended characters are encoded as follows :

- Generate code "1" to switch to set 2, then the code 30 which is the "upper shift" code.
- Substract 128 from the ASCII value of the character to encode; we obtains a not- extended character.
- Encode normally this character with changing the set if necessary.

EDIFACT mode

In this mode 4 data characters are compacted in 3 CWs. Each EDIFACT character is coded with 6 bits which are the 6 last bits of the ASCII value.

EDIFACT value	*ASCII value character*	*Comment*
0 to 30	64 to 94	EDIFACT value = ASCII value - 64
31		End of datas, return to ASCII mode
32 to 63	32 to 63	EDIFACT value = ASCII value

"Base 256" mode.

This mode can encode any byte.

After the 231 CW which switch to "base 256" mode, there is a length field. This field is build with 1 or 2 bytes.

Let N the number of data to encode :

If N < 250 a single byte is used, its value is N (from 0 to 249)

If N >= 250 two bytes are used, the value of the first one is : (N \ 250) + 249 (Values from 250 to 255) and the value of the second one is N MOD 250 (Values from 0 to 249).

If N finishes the filling of the symbol: the value of the byte is 0.

Moreover each CW (including the length field) must be computed with the 255-state algorithm.

Errors detection and correction.

The correction system is based on "Reed Solomon" codes which enjoy the math students and terrify others ...

The number of correction CWs depend of the matrix size, more exactly it depend of the bloc size.

Reed Solomon codes are based on a polynomial equation where x power is the number of error correction CWs used. For sample with the 8 x 8 matrix we use an equation like this : x5 + ax4 + bx3 + cx2 + dx + e. The numbers a, b, c, d and e are the factors of the polynomial equation.

For information the equation is : (x - 2)(x - 22)(x - 23).....(x - 2k) We develop the polynomial equation with Galois arithmetic on each factor...

There is 16 Reed Solomon block size (See table) : 5, 7, 10, 11, 12, 14, 18, 20, 24, 28, 36, 42, 48, 56, 62, 68. The factors of these 16 polynomial equations have been pre-computed. You can see the factors file.

Rather than to draw the algorithm used to compute the correction CWs, I prefer to provide it to you in Basic.

Let k the number of correction CWs, a the factors array, m the number of data CWs, d the data CWs array and c the correction CWs array. We'll use a temporary variable t.

c and t are inited with 0. And let's go with the math fiddle :

```
For i = 0 To m - 1
  t = (d(i) Xor c(k - 1))
  For j = k - 1 To 0 Step -1
    If t = 0 Then
      c(j) = 0
    Else
      c(j) = Mult(t, a(j))
    End If
      If j > 0 Then c(j) = c(j - 1) Xor c(j)
  Next
Next
```

Mult is the special Galois field multiplication.

1.2. License

AIPSYS SOFTWARE LICENSE AGREEMENT

READ THE TERMS OF THIS SOFTWARE LICENSE AGREEMENT (HEREINAFTER THE "AGREEMENT") CAREFULLY. BY DOWNLOADING, INSTALLING, IMPLEMENTING OR USING THIS SOFTWARE PRODUCT, YOU AGREE TO THE TERMS AND CONDITIONS OF THIS AGREEMENT. YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE AS ANY WRITTEN AGREEMENT NEGOTIATED AND SIGNED BY YOU AND AIPSYS.COM INCORPORATED (HEREINAFTER "**AIPSYS SOFTWARE**"). IF YOU ARE ACCESSING SOFTWARE ELECTRONICALLY, INDICATE YOUR ACCEPTANCE OF THESE TERMS BY SELECTING THE "ACCEPT" (OR EQUIVALENT) BUTTON. IF YOU DO NOT AGREE TO ALL OF THE TERMS, PROMPTLY RETURN THE UNUSED SOFTWARE TO YOUR PLACE OF PURCHASE FOR A REFUND OR, IF SOFTWARE IS ACCESSED ELECTRONICALLY, SELECT THE "DECLINE" (OR EQUIVALENT) BUTTON.

NOW, THEREFORE, IN CONSIDERATION OF THE MUTUAL PROMISES SET FORTH HEREIN, AIPSYS SOFTWARE AND YOU HEREBY AGREE AS FOLLOWS:

DEFINITIONS:

(a) "**You**" shall mean the individual using, implementing, downloading, or installing the underlying Software. In the event You are using, implementing, downloading, or installing the underlying Software on behalf of an Organization, all liability for a breach of this agreement shall be the responsibility of said Organization.
(b) "**Licensee**" shall mean You together with any Organization You may be representing, or any related agent, employee, or representative of You that has downloaded, used, installed, or implemented the software package on Your behalf.
(a) "**Software**" shall mean any and all computer programs produced, created, developed, or provided by AIPSYS, including, but not limited to, applicable programs, fonts, components, hosted services, source code, modules, corresponding documentation, updates, upgrades, or modifications thereto.
(b) "**Developer**" shall mean an individual who has a primary job function of developing software applications.
(c) "**Server**" shall mean a computer system that multiple users access or make use of, including but not limited to, terminal servers, file servers, application servers or web servers.
(d) "**Source Code Agreement**" shall mean a separate written instrument governing the use and rights to the underlying Software.
(e) "**Effective Users**" shall mean the number of users that are effective for software licensing, which is determined by the following method that returns the greatest number: (1) The number of users that have access to the Software, (2) The number of computers on which the Software is installed, (3) The number of printers that are being printed to with the Software, or (4) Where the Software is used on a Server or run from a Server, the number of users per week that have access to the Software on the Server, or (5) the number of users per week that have access to programs making use of the Software on the server.

(f) "**Affiliate Program**" shall mean the automated sales referral program described at affiliates program.
(g) "**Organization**" shall mean a single company, business unit, entity or individual. In this Agreement, each subsidiary of a company or business unit with a separate Tax ID is considered a separate Organization.
(h) "**User**" shall mean a single person that is making use of the Software.

TERMS:

1. License Grant

In consideration for the license fee paid, and other good and valuable consideration, AIPSYS grants to Licensee only, unless otherwise limited by the license purchased or granted, the nonexclusive, nontransferable, perpetual, world-wide right to use the Software in accordance with this Agreement and the license defined herein that Licensee purchases ("**License**"). If You are installing, accessing or using this Software for Your employer, this Agreement also includes Your employer. Licensee may only use the Software according to the License purchased or granted by AIPSYS. AIPSYS offers several license types to meet the needs of different Organizations and implementations. Particular Licenses are offered for each product depending on the intended use of the Software. AIPSYS offers some Licenses that are granted to Licensee by this Agreement and not purchased; these include the Optional Integration License, Evaluation License, Free License and the Beta License.

A. Site License - allows use of the Software for all users at a single site within a single Organization. Because of the discounts associated with this license, technical support is provided to a single technical contact at Licensee's Organization instead of to each individual user.

B. Multi Site License - allows use of the Software for an unlimited number of users at an unlimited number of sites within a single Organization. Because of the discounts associated with this license, technical support is provided to a single technical contact at Licensee's Organization instead of to each individual user.

C. Developer Licenses

Developer Licenses offer royalty free use of the Software internally (within the same Organization) and externally (outside the Organization bundled with an application) according to the Developer License Distribution Terms. This license type is licensed by the number of Developers that will be using or working with the Software. The following types of Developer Licenses are available:

(1). One Developer License

The One Developer License (“1DL”) allows royalty-free distribution and use of the Software internally (in the same Organization) and externally (outside the Organization) for a single Developer and up to 10,000 user licenses according to Effective Users, provided Licensee adheres to the Developer License Distribution Terms.

(2). Five Developer License
The Five Developer License ("5DL") allows royalty-free distribution and use of the Software internally (in the same Organization) and externally (outside the Organization) for up to five Developers and up to 20,000 user licenses according to Effective Users, provided Licensee adheres to the Developer License Distribution Terms. This license is also granted if two 1DLs are purchased.

(3). Unlimited Developer License
The Unlimited Developer License ("UDL") allows complete royalty-free distribution and use of the Software internally (in the same Organization) and externally (outside the Organization) for an unlimited number of developers, servers and other user licenses, provided Licensee adheres to the Developer License Distribution Terms.

(4). Small Company Developer Licenses
The Small Company Developer License ("SCDL") grants all rights of the applicable Developer License to all Organizations with a gross annual revenue or funding of less than 2 million U.S. Dollars (or equivalent amount in a foreign currency) with a signed Small Company Agreement. All rights of the Five Developer License are granted if 2 SCDLs are purchased and all rights of the Unlimited Developer License are granted if 3 SCDLs are purchased.

D. Single User License
The Single User License ("SUL") allows use of the Software for one User in Licensee's Organization according to Effective Users.
The SUL shall not be used in connection with: (1) A high speed printer that prints over 55 Pages Per Minute, or (2) A system (including all hardware, printer and software) having a cost totaling over 50,000 USD or equivalent amount in a foreign currency. Such use requires a Developer License, Site License or Multi Site License.
The more single user licenses that are purchased, the more users that are allowed:
1 Single User License = 1 licensed user
2 Single User Licenses = 5 licensed users
3 Single User Licenses = 25 licensed users
4 Single User Licenses = 50 licensed users
5 Single User Licenses = 100 licensed users
6 Single User Licenses = Developer License Granted for 10,000 licensed users

E. Multiple User Licensing
Multiple user licenses grant the rights of the Single User License for a particular number of Users. For example, a 5 User License grants the rights for 5 Single User Licenses. These licenses may be combined; for example, 1 Single User License and a 10 User License = 11 licensed users.

F. Developer use with the Single User License
Developer use with the Single User License requires at least a 5 User License to be purchased unless the Developer and end User are the same person. A Developer may

integrate the Software into an application if the Developer is not the end User, provided the Developer uses one License and the end User uses another License. If more than one Developer uses the Software, Licensee must purchase a Developer License for each additional Developer**.**

G. Single Server License
The Single Server License (“SSL”) allows use of the Software on one (1) server in Licensee's Organization, where a single Server may have only 1 CPU core and up to 100 unique User accesses to the Software from the Server per day. A SSL is required for each additional Server, or CPU core. Additional SSLs may also be obtained for the same server to increase the requirements. For example, 2 SSLs allow 2 CPU cores and up to 200 unique User accesses to the Software per day on the same Server. If 4 SSLs are purchased for the same Software, the rights of the Developer License are granted. If the Software is not used on a server, the licensing options of the Single User License may be used where 1 SSL equals 1 Single User License. The SSL shall not be used in connection with: (1) A high speed printer that prints over 55 Pages Per Minute, or (2) A system (including all hardware, printer and software) having a cost totaling over 50,000 USD or equivalent in a foreign currency. Such use requires a Developer License, Site License or Multi Site License.

ANY OTHER USE REQUIRES A PURCHASE FOR THE ASSOCIATED PRODUCT LICENSE, AFTER A PERIOD OF 30 DAYS, WHICH IS GRANTED TO LICENSEE FOR EVALUATION PURPOSES ONLY.

H. Evaluation License
Software that is distributed as shareware or a demo version may only be used for testing and evaluation purposes only for a period of 30 days.

J. Beta License
Software that is distributed as a beta version may be used during the beta testing period and up to 30 days after the official release is available.

K. Developer License Distribution Terms
As used in this section, the term ("**User Licenses**") shall mean the number of Users that Licensee's License allows according to the definition of Effective Users. The Developer License allows 10,000 Effective Users, The 5 Developer License allows 20,000 Effective Users and the Unlimited Developer License allows an unlimited number of Effective Users.

(a). Internal Distribution:
Allows use of the Software in Licensee’s Organization, provided Licensee does not exceed its User Licenses and Licensee adheres to the following terms:
(1) If distribution of Licensee’s application exceeds its User Licenses, additional Developer Licenses are required; each Developer License purchased will allow distribution of an additional 10,000 Effective Users. Royalty-free, unlimited

distribution is granted after purchasing three Developer Licenses or the Unlimited Developer License.
(2) If more than one Developer is developing with the Software, an additional Developer License is required. Up to 5 Developers may use or develop with the Software after purchasing an additional Developer License or the 5 Developer License. Unlimited Developer use is granted after purchasing three Developer Licenses or the Unlimited Developer License.
(3) The Software may be used on any number of Servers, provided that the number of users accessing all of the servers does not exceed Licensee's User Licenses.

(b). External Distribution:
Allows Licensee to rent, lease or distribute the Software outside its Organization bundled with an application, provided Licensee does not exceed its User Licenses and Licensee adheres to the following terms:
(1) If more than one Developer is developing with the Software, an additional Developer License is required. Up to 5 developers may develop with the Software after purchasing an additional Developer License or the 5 Developer License. Unlimited Developer use is granted after purchasing three Developer Licenses or the Unlimited Developer License.
(2) Licensee may not resell, rent, lease or distribute the Software alone. The Software must be distributed as a component of an application and bundled with an application or with the application's installation files. The Software may only be used as part of, and in connection with, the bundled application. If the Unlimited Developer License is purchased, Licensee may embed the Software into Licensee's firmware, provided a copyright notice is added in the firmware or documentation as detailed in number 5 of this section.
(3) Licensee may not resell, rent, lease, distribute or otherwise use the Software for the License that was purchased, in any way that would compete with AIPSYS. If it is determined by AIPSYS or Licensee that Licensee's distribution or use of the Software competes with AIPSYS, a reasonable royalty fee for Licensee's distribution or use of the Software must be negotiated and agreed to by Licensee and AIPSYS and paid to AIPSYS each quarter or another agreed upon interval of time.
(4) If Licensee uses the Software internally within its Organization, Licensee shall deduct the quantity of its User Licenses used within its Organization from the total number of its User Licenses that are distributed outside its Organization. For example, if Licensee has a One Developer License and uses 4,000 User Licenses internally, it may only distribute up to 6,000 User Licenses outside its Organization.
(5) A valid copyright notice must be provided within the user documentation, start-up screen or in the help-about section of Licensee's application that specifies AIPSYS as the provider of the Software bundled with it's application, for example: "<<your application name>> contains barcode components licensed from AIPSYS.com, Inc. These products may only be used as part of and in connection with <<your application name>>."
(6) Licensee's User Licenses are counted by the number of users the Licensee's

bundled product is licensed for, with the exception that if its product is licensed for more than 500 users per copy, only 500 licenses of the Developer License are used per copy distributed. For example, if Licensee distributes 1 application licensed for 25 users, then that distribution used 25 User Licenses. If Licensee distributes an application that is licensed to be used on a server or host system in such a way that 900 users use it, then that application only uses 500 of its User Licenses.

2. Registration

If Licensee purchases the License directly from AIPSYS, registration is automatic. If Licensee purchases the License from a reseller, Licensee must register the License at www.aipsys.com/register/ before technical support or upgrades for the Software can be made available.

3. Copyright

By downloading, installing, using, or implementing this Software, Licensee acknowledges the validity and enforceability of AIPSYS's copyright in the underlying software and code. The Software and the accompanying materials are licensed, not sold, to Licensee. AIPSYS maintains ownership of all copyright interests in the Software, including any derivative works based upon the Software. Licensee may not rent, lease, display or distribute copies of the Software to others except under the conditions of this Agreement. Unauthorized copying of the Software or accompanying materials even if modified, merged, or included with other software, or of the written materials, is expressly forbidden. Licensee may be held legally responsible for any infringement of intellectual property rights that is caused or encouraged by Licensees failure to abide by the terms of this Agreement. Licensee may make copies of the Software as needed for development and use provided that the number of copies made do not exceed the number of users allowed by the License purchased. Licensee may also make a reasonable number of archival copies of the Software for backup and recovery purposes. In any case, when a copy is created, any copyright notices included in the Software must be reproduced in their entirety on the copy.

4. Software Modifications

If the Unlimited Developer License is purchased, Licensee may modify any portions of the Software as needed, provided that copyright notices are not removed, including but not limited to the height, width and tables of any fonts provided.

5. Agreement Duration and Termination

Subject to the terms and conditions of this Agreement, this Agreement begins when the Software is downloaded, installed, used or when a License for Software is purchased or granted and is perpetual unless terminated. When the Agreement begins, this Agreement shall supersede all older versions of this Agreement including any older Agreements that may be embedded in the Software. This Agreement shall inure to the benefit of

and be binding upon AIPSYS and Licensee. Licensee may terminate this Agreement at any time by returning the Software to AIPSYS and destroying all copies thereof. This Agreement shall terminate upon notice from AIPSYS if Licensee fails to comply with any provision contained herein or if the funds paid for the license are refunded or are not received, and such failure or breach is not cured within thirty (30) days of such notice. Upon termination, Licensee must destroy the Software and all copies (in part and in whole, including modified copies, if any) in its possession or control. AIPSYS reserves the right to terminate this Agreement if the use of Software by Licensee causes a loss of revenue for AIPSYS that exceeds ten (10) times the amount Licensee paid for the License. Termination of this Agreement shall not affect the Software bundled and distributed with an application under the Developer License by Licensee prior to termination, provided Licensee has purchased a Developer License for the Software, the bundled application does not compete with AIPSYS in any way, and funds for the License were received and not returned or refunded in any way. All restrictions prohibiting Licensee's use of the Software and intellectual property provisions relating to Software to the benefit of AIPSYS shall survive termination of this Agreement.

6. Warranty and Limitation of Liability

Although efforts have been made to assure that the Software is date compliant, correct, reliable, technically accurate and will perform in accordance with the documentation, the Software is licensed to Licensee as is and without warranties as to performance of merchantability, fitness for a particular purpose or use, or any other warranties whether expressed or implied. Licensee, its Organization, and all users of the Software, assume all risks when using it. To the maximum extent permitted by applicable law, in no event shall AIPSYS be liable for any consequential, incidental, indirect, punitive or special damages arising out of the use of or inability to use the Software or the provision of or failure to provide support services or hosted services, even if AIPSYS has been advised of the possibility of such damages. In any case, AIPSYS's entire liability under any provision of this Agreement shall be limited to ten (10) times the amount actually paid by Licensee for the License or $5.00 USD if no license was purchased.

7. Technical Support and Product Upgrades

Unless otherwise indicated in the documentation of the Software, AIPSYS offers a free Priority Support and Product Upgrade Subscription for a period of thirty (30) days from the date of purchase on all licensed Software. When Licensee's Priority Support is active, Licensee may contact AIPSYS by phone, email and through the Online Priority Support Request Form. Priority Support and Product Upgrades may be provided beyond

thirty (30) days if the Priority Support and Upgrade Subscription is purchased. Support may be provided to the appropriate individual that (a) ordered the License; (b) is integrating the Software; (c) a Developer; or (d) the end user if each end user has a separate License for the Software. If one Developer License is purchased, technical support is provided for only one Developer. If the 5 Developer License is purchased, technical support is provided for up to 5 Developers. If the Unlimited Developer License is purchased, technical support is provided for an unlimited number of Developers. The Developers responsibilities may be transferred to another Developer within the Organization as necessary provided no more than 2 transfers occur within any ninety (90) day period. If Licensee's Priority Support and Product Upgrade Subscription expires, Licensee may obtain free technical support by referring to support documents at the website or by renewing the Priority Support and Product Upgrade Subscription.

Whenever any Software update, upgrade, or revision is provided to Licensee or Licensee purchases an additional License, all related Software from AIPSYS (including any Software that was acquired previously) shall be covered by the latest version of the Agreement that exists at the time the most recent update was provided to Licensee.

1.3. About trial version

With 2D barcode encoder and decoder SDK, some of the input element will be replaced with char '*' before encoding, and some of the output element will be replaced with '*' after decoding.

With 1D linear barcode encoder and decoder, some of the input element will be replaced with char '0' before encoding, and some of the output element will be replaced with '0' after decoding.

The Trial version have 30 days' evaluation time, you must remove it from your computer and your application after expiration.

We will mail the licensed version or register serial no to you after you order it.

2. Encoder SDK

2.1. Static link library

2.1.1 Constants

Size pattern constant of DataMatrix encoding

```
#define MAPSIZE10X10      0
#define MAPSIZE12X12      1
#define MAPSIZE14X14      2
#define MAPSIZE16X16      3
#define MAPSIZE18X18      4
#define MAPSIZE20X20      5
#define MAPSIZE22X22      6
#define MAPSIZE24X24      7
#define MAPSIZE26X26      8
#define MAPSIZE32X32      9
#define MAPSIZE36X36      10
#define MAPSIZE40X40      11
#define MAPSIZE44X44      12
#define MAPSIZE48X48      13
#define MAPSIZE52X52      14
#define MAPSIZE64X64      15
#define MAPSIZE72X72      16
#define MAPSIZE80X80      17
#define MAPSIZE88X88      18
#define MAPSIZE96X96      19
#define MAPSIZE104X104    20
#define MAPSIZE120X120    21
#define MAPSIZE132X132    22
#define MAPSIZE144X144    23
#define MAPSIZE8X18       24
#define MAPSIZE8X32       25
#define MAPSIZE12X26      26
#define MAPSIZE12X36      27
#define MAPSIZE16X36      28
#define MAPSIZE16X48      29
#define SQUARESIZEAUTO          -1
#define RECTANGLESIZEAUTO       -2
```

2.1.2 Data structure

The following data structure define the properties of the DataMatrix barcode, it can be transfered into function as parameter.

```
typedef struct _tagDATAMATRIXCONTEXT
{
    char code[3300];      //define the data to be encoded
    int size;             //define the data size of the data to be encoded
    int encoding;         //coding pattern
                          //0: EncodeAscii;
                          //1: EncodeC40;
                          //2: EncodeText;
                          //3: EncodeBase256;
                          //4: EncodeX12;
                          //5: EncodeEdifact
    int dotPixel;         //module size
    int sizePattern;      // size pattern of rectangle and square datamatrix
    int nMargin;          // define white margin of the output image
    COLORREF clBackGround;  //define the fore ground color of output image
    COLORREF clForeGround;  //define the fore ground color of output image
} _DATAMATRIXCONTEXT,*_LPDATAMATRIXCONTEXT;
```

2.1.3. Function or procedure

2.1.3.1. _InitDataMatrixContext

The _InitDataMatrixContext function initilize the environment of DataMatrix encoding with default value.

void __stdcall _InitDataMatrixContext(_DATAMATRIXCONTEXT *pDmCtx);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _PDATAMATRIXCONTEXT

Return values

None

2.1.3.2. _DataMatrixEncode2File

The DataMatrixEncode2File function encode the data inputted with the defined attributes and save the barcode to an image file

BOOL __stdcall _DataMatrixEncode2File(_DATAMATRIXCONTEXT *pDmCtx,
LPCTSTR lpImageFile);

Parameters

pDmCtx
[in] define the DataMatrix attributes for encoding, refer structure type _DATAMATRIXCONTEXT

lpImageFile
[in] define the image file outputted, currently bitmap image supported

Return values

If the function succeeds, the return value is TRUE,otherwise , return FALSE.

2.1.3.3. _DataMatrixEncode2Bitmap

The **DataMatrixEncode2Bitmap** function encode the data inputted with the defined attributes and return the bitmap handle of the DataMatrix barcode image

HBITMAP __stdcall _DataMatrixEncode2Bitmap(
_DATAMATRIXCONTEXT *pDmCtx);

Parameters

pDmCtx
[in] define the DataMatrix attributes for encoding, refer structure type _DATAMATRIXCONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of DATAMATRIX Barcode, otherwise , return NULL.

2.1.3.4. _ _FreeDataMatrixContext

The _FreeDataMatrixContext function free environment of the DataMatrix encoding

BOOL __stdcall _ _FreeDataMatrixContext ();

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.1.3.5. _DataMatrixEncode2SMSImage

The _DataMatrixEncode2SMSImage function encode the data to specilized size bitmap (72X28 or 32X32) , it can be send as SMS message

```
BOOL __stdcall _DataMatrixEncode2SMSImage (char *pText,
                                           LPCTSTR pFile,int nWidth,int nHeight);
```

Parameters

pText

[in] define the data encoded

pFile

[in] Image file name of output, only support bitmap format.

nWidth

[in] define image width of output

nHeight

[in] define image height of output

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.1.4. Example for Microsoft visual C++

Example1

```
#include "DataMatrixEncodeLIB.h"

....
    _tagDATAMATRIXCONTEXT tDmCtx;
    _InitDataMatrixContext(&tDmCtx);
    tDmCtx.encoding = 0;
    tDmCtx.dotPixel = 4;
    tDmCtx.sizePattern = -1;    //auto size pattern
    tDmCtx.nMargin = 10;
    tDmCtx.clBackGround = RGB(255,255,255);
    tDmCtx.clForeGround = RGB(255,0,0);
    sprintf(tDmCtx.code,"http://www.aipsys.com");
    tDmCtx.size = strlen(tDmCtx.code);
```

```
    _DataMatrixEncode2File(&tDmCtx, "c:\\dm.bmp");
    _FreeDataMatrixContext();
.....
```

LIBRARY for linking

DataMatrixEncodeLIB.lib

2.2. Dynamic link library

2.2.1. Data structure

The following data structure define the properties of the DataMatrix barcode, it can be transfer into function as parameter.

```
typedef struct   tagDATAMATRIXCONTEXT
{
    char code[3300];      //define the data to be encoded
    int size;             //define the data size of the data to be encoded
    int encoding;         //coding pattern
                          //0: EncodeAscii;
                          //1: EncodeC40;
                          //2: EncodeText;
                          //3: EncodeBase256;
                          //4: EncodeX12;
                          //5: EncodeEdifact
    int dotPixel;         //module size
    int sizePattern;      // size pattern of rectangle and square datamatrix
    int nMargin;          // define white margin of the output image
    COLORREF clBackGround;  //define the fore ground color of output image
    COLORREF clForeGround;  //define the fore ground color of output image
} DATAMATRIXCONTEXT,*LPDATAMATRIXCONTEXT;
```

Size pattern constant of DataMatrix encoding

```
#define MAPSIZE10X10    0
#define MAPSIZE12X12    1
#define MAPSIZE14X14    2
#define MAPSIZE16X16    3
#define MAPSIZE18X18    4
#define MAPSIZE20X20    5
#define MAPSIZE22X22    6
#define MAPSIZE24X24    7
#define MAPSIZE26X26    8
```

```
#define MAPSIZE32X32      9
#define MAPSIZE36X36      10
#define MAPSIZE40X40      11
#define MAPSIZE44X44      12
#define MAPSIZE48X48      13
#define MAPSIZE52X52      14
#define MAPSIZE64X64      15
#define MAPSIZE72X72      16
#define MAPSIZE80X80      17
#define MAPSIZE88X88      18
#define MAPSIZE96X96      19
#define MAPSIZE104X104    20
#define MAPSIZE120X120    21
#define MAPSIZE132X132    22
#define MAPSIZE144X144    23
#define MAPSIZE8X18       24
#define MAPSIZE8X32       25
#define MAPSIZE12X26      26
#define MAPSIZE12X36      27
#define MAPSIZE16X36      28
#define MAPSIZE16X48      29
#define SQUARESIZEAUTO          -1
#define RECTANGLESIZEAUTO       -2
```

2.2.2. Function or procedure

2.2.2.1. InitWorkSpace

The _InitWorkSpace function initilize the environment of DataMatrix encoding with default value.

void __stdcall _InitWorkSpace(LPDATAMATRIXCONTEXT pDmCtx);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type LPPDATAMATRIXCONTEXT

Return values

None

2.2.2.2. DataMatrixEncode2File

The DataMatrixEncode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall DataMatrixEncode2File(LPDATAMATRIXCONTEXT pDmCtx,
LPCTSTR lpImageFile);

Parameters

pDmCtx
[in] define the DataMatrix attributes for encoding, refer structure type LPDATAMATRIXCONTEXT

lpImageFile
[in] define the image file outputted, currently bitmap image supported

Return values

If the function succeeds, the return value is TRUE,otherwise , return FALSE.

2.2.2.3. DataMatrixEncode2Bitmap

The **DataMatrixEncode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the DataMatrix barcode image

HBITMAP __stdcall DataMatrixEncode2Bitmap(
LPDATAMATRIXCONTEXT pDmCtx);

Parameters

pDmCtx
[in] define the DataMatrix attributes for encoding, refer structure type LPDATAMATRIXCONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of DATAMATRIX Barcode, otherwise , return NULL.

2.2.2.4. FreeWorkSpace

The FreeWorkSpace function free environment of the DataMatrix encoding

BOOL __stdcall FreeWorkSpace();

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.2.2.5. DataMatrixEncode2SMSImage

The DataMatrixEncode2SMSImage function encode the data to specilized size bitmap (72X28 or 32X32) , it can be send as SMS message

```
BOOL __stdcall    DataMatrixEncode2SMSImage (char *pText,
                                             LPCTSTR pFile,int nWidth,int nHeight);
```

Parameters

pText

[in] define the data encoded

pFile

[in] Image file name of output, only support bitmap format.

nWidth

[in] define image width of output

nHeight

[in] define image height of output

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.2.2.6. EncodeDataMatrix2File

The **EncodeDataMatrix2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
BOOL   __stdcall EncodeDataMatrix2File(char *pBuf,
            int nSize,int nScheme,int nPixelSize,
            int nSizePattern,int nMargin,COLORREF clBack,
            COLORREF clFore, LPCTSTR lpImageFile);
```

Parameters

pBuf

[in] define the data encoded

nSize

[in] data size input

nScheme

[in] encode pattern such as ascii, text, base256 and etc.

nPizelSize
[in] Module size of barcode image
nSizePattern
[in] define size pattern and form of barcode
nMargin
[in] define margin of output barcode image
clFore
[in] define foreground color
clBack
[in] define background color
lpImageFile
[in] bitmap image output

Return values
If the function succeeds, the return TRUE, otherwise , return FALSE.

2.2.2.7. EncodeDataMatrix2Bitmap

The **EncodeDataMatrix2Bitmap** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
HBITMAP __stdcall EncodeDataMatrix2Bitmap(char *pBuf,
        int nSize,int nScheme,int nPixelSize,
        int nSizePattern,int nMargin,COLORREF clBack,
        COLORREF clFore);
```

Parameters

pBuf
[in] define the data encoded
nSize
[in] data size input
nScheme
[in] encode pattern such as ascii, text, base256 and etc.
nPizelSize
[in] Module size of barcode image
nSizePattern
[in] define size pattern and form of barcode
nMargin
[in] define margin of output barcode image
clFore
[in] define foreground color
clBack
[in] define background color

Return values

If the function succeeds, the return Image Handle, otherwise , return NULL.

2.2.3. Example for Microsoft visual C++

Example1

```
#include "DataMatrixEncodeDLL.h"

....
    DATAMATRIXCONTEXT tDmCtx;
    InitWorkSpace(&tDmCtx);
    tDmCtx.encoding = 0;
    tDmCtx.dotPixel = 4;
    tDmCtx.sizePattern=-1;
    tDmCtx.nMargin = 10;
    tDmCtx.clBackGround = RGB(255,255,255);
    tDmCtx.clForeGround = RGB(255,0,0);
    sprintf(tDmCtx.code,"http://www.aipsys.com");
    tDmCtx.size=strlen(tDmCtx.code);
    DataMatrixEncode2File(&tDmCtx, "c:\\dm.bmp");
    FreeWorkSpace();

.....


LIBRARY for linking
      DataMatrix EncodeDLL.lib
Runtime library
      DataMatrixEncodeDLL.DLL
```

2.2.4. Example for Borland Delphi

2.2.4.1. Redeclaration of the data type and function

```
type
LPDATAMATRIXCONTEXT = ^TDATAMATRIXCONTEXT;
TDATAMATRIXCONTEXT = record
    code : array [1..3000] of char;
    size : integer;
```



```
    encoding : integer;
    dotPixel : integer;
   sizePattern : integer;
    nMargin : integer;
    clForeGround : TColor;
   clBackGround : TColor;
end;

procedure InitWorkSpace(pDmCtx : LPDATAMATRIXCONTEXT ); stdcall;
                external 'DATAMATRIXENCODEDLL.DLL';

function   DataMatrixEncode2File(pDmCtx :
              LPDATAMATRIXCONTEXT;lpImageFile :PChar) :
              boolean; stdcall;external 'DATAMATRIXENCODEDLL.DLL';

function   DataMatrixEncode2Bitmap(pDmCtx : LPDATAMATRIXCONTEXT) : HBITMAP;
                stdcall;external 'DATAMATRIXENCODEDLL.DLL';

function   FreeWorkSpace : boolean;stdcall external 'DATAMATRIXENCODEDLL.DLL';

function   EncodeDataMatrix2File (pBuf: PChar;nSize : Integer; nScheme : Integer;
          nPizelSize:Integer; nSizePattern:Integer; nMargin:integer; clBack : TColor; clFore :
          TColor , lpImageFile : PChar):   Boolean ;stdcall;external
          'DATAMATRIXENCODEDLL.DLL';

function   EncodeDataMatrix2Bitmap (pBuf: PChar;nSize : Integer; nScheme : Integer;
          nPizelSize:Integer; nSizePattern:Integer; nMargin:integer; clBack : TColor; clFore :
          TColor ):   HBITMAP ;stdcall;external 'DATAMATRIXENCODEDLL.DLL';
```

2.2.4.2. Example

Example1

```
var
ctx : TDATAMATRIXCONTEXT;
s : string;
i : Integer;
pCtx : LPDATAMATRIXCONTEXT;
begin
    pCtx := @ctx;
    InitWorkSpace(pCtx);
    ctx.sizePattern := -1;
    ctx.encoding := 0;
```



```
        ctx.nMargin := 20;
        ctx. dotPixel := 2;
        ctx.clForeGround := RGB(255,0,0);
        ctx.clBackGround := RGB(255,255,255);
        Strcopy(ctx.cData,PChar('edMemo.Text'));
        ctx.nSize := 11;
        DataMatrixEncode2File(pCtx,PChar('c:\test.bmp'));
        FreeWorkSpace();
    end;
```

Example2

```
    if  EncodeDataMatrix2File(PChar('1234567890'),10  ,0,2,-1,10,  RGB(255,0,0),
          RGB(255,255,255), PChar('c:\test.bmp')) then
    begin
        ShowMessage('Encode success');
    end;
```

Example3

```
    hBarcode : HBITMAP;
    hBarcode:=    EncodeDataMatrix2Bitmap(PChar('1234567890'),10    ,0,2,-1,10,
                          RGB(255,0,0), RGB(255,255,255));
    …..
    DeleteObject(hBarcode);
```

2.2.5. Example for Microsoft visual Basic

2.2.5.1. Redeclaration of the data type and function

```
Private Declare Function DataMatrixEncode2SMSImage Lib
"DataMatrixEncodeDll.dll" (ByVal pBuf As String, ByVal ImgFile As String, ByVal
nWidth As Long, ByVal nHeight As Long) As Boolean

Private Declare Function DataMatrixDecodeImageFile Lib
"DataMatrixDecodeDll.dll" (ByVal pFile As String, ByVal pBuf As Any, ByVal
pSize As Long) As Boolean
```

2.2.5.2. Example

Example1

```
….
  If (DataMatrixEncode2SMSImage("ABCDEFGHIJKLMNOP", "c:\pic4.bmp", 72, 28)) Then
```


MsgBox ("SUccess, Trial version will replace some char of input with '*' randomly")
Else
MsgBox ("failed")
End If…….

2.3. ActiveX

2.3.1. Properties

2.3.1.1. sizePattern

The property set the sizePattern of DataMatrix barcode

short size Pattern

2.3.1.2. encodingMode

The property set the encoding pattern of DataMatrix barcode

short encoding

//0: EncodeAscii;

//1: EncodeC40;

//2: EncodeText;

//3: EncodeBase256;

//4: EncodeX12;

//5: EncodeEdifact

2.3.1.3. dotSize

The property set the module size of DataMatrix barcode

short PixelSize

2.3.1.4. Margin

The property set the margin of DataMatrix barcode

short Margin

2.3.1.5. ForeGroundColor

The property set the Foreground color of DataMatrix barcode

OLE_COLOR ForeGroundColor

2.3.1.6. BackGroundColor

The property set the Background color of DataMatrix barcode

OLE_COLOR BackGroundColor

2.3.1.7. TextData

The property set the data to be encoded

```
BSTR  TextData
```

2.3.2. Methods

2.3.2.1. Encode2ImageFile

The method Encode2ImageFile encode the data inputed and save the barcode image to file.

boolean Encode2ImageFile(BSTR lpImageFile);

Parameters

lpImageFile

[in] specify the barcode image file to be saved

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.3.3. Register activeX component

Regsvr32 DataMatrixEncodeOcx.OCX

2.3.4. Example for Microsoft visual C++

To refer it in VC:

Run Visual C++;

Select menu project, click add to project item, then click components and controls

Select Registered ActiveX controls then select DataMatrixEncodeOcx.Ocx

```
#include “DataMatrixEncodeOcx.h”

    CDataMatrixEncodeOcx    objDataMatrix;
    objDataMatrix.TextData = "http://www.aipsys.com";
    objDataMatrix.sizePattern = -1;
    objDataMatrix.encodingMode = 0;
    objDataMatrix.ForeGroundColor = RGB(255,0,0);
    objDataMatrix.Margin = 10;
    objDataMatrix.Encode2ImageFile("c:\\pdf.gif");
```

2.3.5. Example for Borland Delphi

To install it to Delphi:

Run Delphi

Select menu-> component, click Import ActiveX Control item,

Select DataMatrixEncodeOcx ActiveX module when dialog shows,

Install it. You can find the component in the Active Page

```
Uses
    …   DataMatrixEncodeOcx_TLB;
objDM :   TDataMatrixEncodeOcx;
begin
    objDM.TextData := ‘http://www.aipsys.com’
    objDM.sizePattern   := -1
    objDM.encodingMode   := 0
    objDM.ForeGroundColor   := &HFF00FF
    objDM.Margin   := 10
    objDM.Encode2ImageFile(‘c:\dm.gif’);
end;
```

2.3.6. Example for Microsoft visual Basic

```
Private Sub Command1_Click()
    DataMatrixEncodeOCX1.TextData = "http://www.aipsys.com"
    DataMatrixEncodeOCX1.sizePattern = -1
    DataMatrixEncodeOCX1.encodingMode = 0
    DataMatrixEncodeOCX1.ForeGroundColor = &HFF00FF
    DataMatrixEncodeOCX1.Margin = 10
    DataMatrixEncodeOCX1.Encode2ImageFile("c:\dm.gif");
End Sub
```

2.4. ASP Control for server side

2.4.1. Properties

2.4.1.1. sizePattern

The property set the sizePattern of DataMatrix barcode

short nSizePattern

2.4.1.2. nPixelSize

The property set the module width of DataMatrix barcode

short nPixelWidth

2.4.1.3. nEncodeMode

The property set the encoding mode of DataMatrix barcode

short nEncodeMde

//0: EncodeAscii;

//1: EncodeC40;

//2: EncodeText;

//3: EncodeBase256;

//4: EncodeX12;

//5: EncodeEdifact

2.4.1.4. nMargin

The property set the margin of DataMatrix barcode

short nMargin

2.4.1.5. clForeGround

The property set the Foreground color of DataMatrix barcode

OLE_COLOR clForeGround

2.4.1.6. clBackGround

The property set the Background color of DataMatrix barcode

OLE_COLOR clBackGround

2.4.1.7. strText

The property set the data to be encoded

BSTR strText

2.4.2. Methods

2.4.2.1. InitWorkspace

The method InitWorkspace initialize the working environment

BOOL InitWorkspace().

Parameters

none

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.4.2.2. FreeWorkspace

The method FreeWorkspace destroy the working environment

BOOL FreeWorkspace().

Parameters

none

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.3.2.3. Encode2File

The method Encode2File encode the data inputed and save the barcode image to file.

boolean Encode2File(BSTR strImageFile);

Parameters

strImageFile

[in] specify the barcode image file to be saved

Return values

If the function succeeds, the return TRUE, otherwise , return FALSE.

2.4.3. Register the ASP server component

Regsvr32 DataMatrixEncodeASP.DLL

2.4.4. Example for ASP

```
<%
    set obj = Server.CreateObject("DataMatrixEncodeCOM.EncodeService")
    obj.InitWorkspace()
    obj.nSizePattern = -1           '  to select size automatically according text length
    obj.nMargin = 5           '  Margin
    obj.nPixelSize = 2          '  PixelSize
    obj.nEncodeMode = 0    '    ascii
    obj.strText = "Http://www.aipsys.com"
```



```
    obj.Encode2File("C:\1.gif")
    obj.FreeWorkspace()
    response.Write("<img src='1.gif'>")
    response.Write("<br>")
    response.Write("Trial Version randomly change element of input with * <br>")

%>
```

2.5. Library for IOS

2.5.1 Constants

Size pattern constant of DataMatrix encoding

```
//constant for sizePattern of datamatrix image.
#define SQUAREAUTOSIZE          -1
#define RECTANGLEAUTOSIZE       -2

#define MAPSIZE10X10        0
#define MAPSIZE12X12        1
#define MAPSIZE14X14        2
#define MAPSIZE16X16        3
#define MAPSIZE18X18        4
#define MAPSIZE20X20        5
#define MAPSIZE22X22        6
#define MAPSIZE24X24        7
#define MAPSIZE26X26        8
#define MAPSIZE32X32        9
#define MAPSIZE36X36        10
#define MAPSIZE40X40        11
#define MAPSIZE44X44        12
#define MAPSIZE48X48        13
#define MAPSIZE52X52        14
#define MAPSIZE64X64        15
#define MAPSIZE72X72        16
#define MAPSIZE80X80        17
#define MAPSIZE88X88        18
#define MAPSIZE96X96        19
#define MAPSIZE104X104      20
#define MAPSIZE120X120      21
#define MAPSIZE132X132      22
#define MAPSIZE144X144      23
#define MAPSIZE8X18         24
#define MAPSIZE8X32         25
```



```
#define MAPSIZE12X26      26
#define MAPSIZE12X36      27
#define MAPSIZE16X36      28
#define MAPSIZE16X48      29
```

2.5.2 Data structure

The following data structure define the properties of the DataMatrix barcode, it can be transfered into function as parameter.

```
typedef unsigned int COLORREF;
typedef unsigned char BYTE;

typedef struct _tagDATAMATRIXCONTEXT
{
    char code[3300];
    int size;
    int encoding;
            //0: EncodeAscii;
            //1: EncodeC40;
            //2: EncodeText;
            //3: EncodeBase256;
            //4: EncodeX12;
            //5: EncodeEdifact
    int dotPixel;
    int sizePattern;
    int nMargin;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _DATAMATRIXCONTEXT,*_LPDATAMATRIXCONTEXT;
```

2.5.3. Function or procedure

2.5.3.1. _InitDataMatrixContext

The _InitDataMatrixContext function initilize the environment of DataMatrix encoding with default value.

void __stdcall _InitDataMatrixContext(_DATAMATRIXCONTEXT *pDmCtx);

Parameters

pDmCtx
[in] define the DataMatrix attributes for encoding, refer structure type _PDATAMATRIXCONTEXT

Return values

None

2.5.3.2. _DataMatrixEncode2Bitmap

The DataMatrixEncode2Bitmap function encode the data inputed with the defined attributes and save the barcode to an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _DataMatrixEncode2Bitmap(_LPDATAMATRIXCONTEXT pDmCtx, int *pWidth, int *pHeight);

Parameters

pDmCtx
[in] define the DataMatrix attributes for encoding, refer structure type _DATAMATRIXCONTEXT
pWidth
[in/out] return the width of the bitmap buffer output
pHeight
[in/out] return the height of the bitmap buffer output

Return values

If the function succeeds, the return value are RGB bitmap buffer of DataMatrix barcode , pWidth \pHeight; otherwise , return NULL.

2.5.3.3. _DataMatrixEncodeRegister

The _DataMatrixEncodeRegister function initilize the environment of DataMatrix encoding with default value.

bool _DataMatrixEncodeRegister(char *pMailBox,char *pRegCode);

Parameters

pMailBox: Mail box used to generate the regcode
pRegCode: regcode generated with mail box string

Return values

Return TRUE if register the product successfully, otherwise return FALSE .

2.5.4. Example for Object C

Example1

```
#include "DataMatrixEncodeLib.h"

....
    _tagDATAMATRIXCONTEXT tDmCtx;
    _InitDataMatrixContext(&tDmCtx);
    tDmCtx.encoding = 0;
    tDmCtx.dotPixel = 4;
    tDmCtx.sizePattern = -1;    //auto size pattern
    tDmCtx.nMargin = 10;
    tDmCtx.clBackGround = RGB(255,255,255);
    tDmCtx.clForeGround = RGB(255,0,0);
    sprintf(tDmCtx.code,"http://www.aipsys.com");
    tDmCtx.size = strlen(tDmCtx.code);
    _DataMatrixEncode2Bitmap(&tDmCtx, "c:\\dm.bmp");
    _FreeDataMatrixContext();
.....
```

LIBRARY for linking

DataMatrixEncodeLIBIOS.lib

2.6. Library for Linux

2.6.1 Constants

Size pattern constant of DataMatrix encoding

```
//constant for sizePattern of datamatrix image.
#define SQUAREAUTOSIZE          -1
#define RECTANGLEAUTOSIZE       -2

#define MAPSIZE10X10        0
#define MAPSIZE12X12        1
#define MAPSIZE14X14        2
#define MAPSIZE16X16        3
#define MAPSIZE18X18        4
```

```
#define MAPSIZE20X20      5
#define MAPSIZE22X22      6
#define MAPSIZE24X24      7
#define MAPSIZE26X26      8
#define MAPSIZE32X32      9
#define MAPSIZE36X36      10
#define MAPSIZE40X40      11
#define MAPSIZE44X44      12
#define MAPSIZE48X48      13
#define MAPSIZE52X52      14
#define MAPSIZE64X64      15
#define MAPSIZE72X72      16
#define MAPSIZE80X80      17
#define MAPSIZE88X88      18
#define MAPSIZE96X96      19
#define MAPSIZE104X104    20
#define MAPSIZE120X120    21
#define MAPSIZE132X132    22
#define MAPSIZE144X144    23
#define MAPSIZE8X18       24
#define MAPSIZE8X32       25
#define MAPSIZE12X26      26
#define MAPSIZE12X36      27
#define MAPSIZE16X36      28
#define MAPSIZE16X48      29
```

2.6.2 Data structure

The following data structure define the properties of the DataMatrix barcode, it can be transfered into function as parameter.

```
typedef unsigned int COLORREF;
typedef unsigned char BYTE;

typedef struct _tagDATAMATRIXCONTEXT
{
    char code[3300];
    int size;
    int encoding;
            //0: EncodeAscii;
            //1: EncodeC40;
            //2: EncodeText;
            //3: EncodeBase256;
            //4: EncodeX12;
```



```
            //5: EncodeEdifact
    int dotPixel;
    int sizePattern;
    int nMargin;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _DATAMATRIXCONTEXT,*_LPDATAMATRIXCONTEXT;
```

2.6.3. Function or procedure

2.6.3.1. _InitDataMatrixContext

The _InitDataMatrixContext function initilize the environment of DataMatrix encoding with default value.

void __stdcall _InitDataMatrixContext(_DATAMATRIXCONTEXT *pDmCtx);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _PDATAMATRIXCONTEXT

Return values

None

2.6.3.2. _DataMatrixEncode2Bitmap

The DataMatrixEncode2Bitmap function encode the data inputed with the defined attributes and save the barcode to an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _DataMatrixEncode2Bitmap(_LPDATAMATRIXCONTEXT pDmCtx, int *pWidth, int *pHeight);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _DATAMATRIXCONTEXT

pWidth

[in/out] return the width of the bitmap buffer output

pHeight

[in/out] return the height of the bitmap buffer output

Return values

If the function succeeds, the return value are RGB bitmap buffer of DataMatrix barcode , pWidth \pHeight; otherwise , return NULL.

2.6.3.3. _DataMatrixEncodeRegister

The _DataMatrixEncodeRegister function initilize the environment of DataMatrix encoding with default value.

bool _DataMatrixEncodeRegister(char *pMailBox,char *pRegCode);

Parameters

pMailBox: Mail box used to generate the regcode

pRegCode: regcode generated with mail box string

Return values

Return TRUE if register the product successfully, otherwise return FALSE .

2.6.4. Example for C/C++

2.6.4.1 Example1

In this example, there we declared a function named save_png_to_file() which inputs the bitmap-buffer we get before and outputs the right result to a file named “fruit.png”

```
/* Given "bitmap", this returns the pixel of bitmap at the point

   ("x", "y"). */

static pixel_t * pixel_at (unsigned char *pRgb,int width, int x, int y)

{

    return (pixel_t*)(pRgb + width * y * 3 + x * 3);

}

/* Write "bitmap" to a PNG file specified by "path"; returns 0 on
```



```
    success, non-zero on error. */
static int save_png_to_file (unsigned char *pRgb,int width,int height, const char *path)
{
    FILE * fp;
    png_structp png_ptr = NULL;
    png_infop info_ptr = NULL;
    size_t x, y;
    png_byte ** row_pointers = NULL;
    /* "status" contains the return value of this function. At first
       it is set to a value which means 'failure'. When the routine
       has finished its work, it is set to a value which means
       'success'. */
    int status = -1;
    /* The following number is set by trial and error only. I cannot
       see where it it is documented in the libpng manual.
    */
    int pixel_size = 3;
    int depth = 8;
    fp = fopen (path, "wb");
    if (! fp) {
        goto fopen_failed;
    }
    png_ptr = png_create_write_struct (PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
```



```
    if (png_ptr == NULL) {
        goto png_create_write_struct_failed;
    }
    info_ptr = png_create_info_struct (png_ptr);
    if (info_ptr == NULL) {
        goto png_create_info_struct_failed;
    }
    /* Set up error handling. */
    if (setjmp (png_jmpbuf (png_ptr))) {
        goto png_failure;
    }
    /* Set image attributes. */
    png_set_IHDR (png_ptr, info_ptr,width, height, depth, PNG_COLOR_TYPE_RGB,
PNG_INTERLACE_NONE, PNG_COMPRESSION_TYPE_DEFAULT,
PNG_FILTER_TYPE_DEFAULT);
    /* Initialize rows of PNG. */
    row_pointers = (png_byte **)png_malloc (png_ptr, height * sizeof (png_byte *));
    for (y = 0; y < height; ++y) {
        png_byte *row =(png_byte *)
            png_malloc (png_ptr, sizeof (uint8_t) * width * pixel_size);
        row_pointers[y] = row;
        for (x = 0; x < width; ++x) {
            pixel_t * pixel = pixel_at(pRgb,width, x, y);
            *row++ = pixel->red;
            *row++ = pixel->green;
```



```
                *row++ = pixel->blue;
            }
        }
        /* Write the image data to "fp". */
        png_init_io (png_ptr, fp);
        png_set_rows (png_ptr, info_ptr, row_pointers);
        png_write_png (png_ptr, info_ptr, PNG_TRANSFORM_IDENTITY, NULL);
        /* The routine has successfully written the file, so we set
           "status" to a value which indicates success. */
        status = 0;
        for (y = 0; y < height; y++) {
            png_free (png_ptr, row_pointers[y]);
        }
        png_free (png_ptr, row_pointers);
    png_failure:
     png_create_info_struct_failed:
        png_destroy_write_struct (&png_ptr, &info_ptr);
     png_create_write_struct_failed:
        fclose (fp);
     fopen_failed:
        return status;
    }
```


2.6.4.2 Source code

```
#include "DataMatrixEncodeLib.h"

....
    _tagDATAMATRIXCONTEXT tDmCtx;

    int width,height;

    unsigned char *pMap;


    _InitDataMatrixContext(&tDmCtx);
    tDmCtx.encoding = 0;
    tDmCtx.dotPixel = 4;
    tDmCtx.sizePattern = -1;    //auto size pattern
    tDmCtx.nMargin = 10;
    tDmCtx.clBackGround = RGB(255,255,255);
    tDmCtx.clForeGround = RGB(255,0,0);
    sprintf(tDmCtx.code,"http://www.aipsys.com");
    tDmCtx.size = strlen(tDmCtx.code);
    pMap =_DataMatrixEncode2Bitmap (&tDmCtx, "c:\\dm.bmp");

    if(!pMap) return;

    /* Write the image to a file 'fruit.png' by the function we declared. */

    save_png_to_file (pMap,width,height, "fruit.png");

    free(pMap);
    _FreeDataMatrixContext();
.....
```

LIBRARY for linking

DataMatrixEncodeLIBLinux.lib

2.7. Library for Linux ARM

2.7.1 Constants

Size pattern constant of DataMatrix encoding

//constant for sizePattern of datamatrix image.

```
#define SQUAREAUTOSIZE          -1
#define RECTANGLEAUTOSIZE       -2

#define MAPSIZE10X10        0
#define MAPSIZE12X12        1
#define MAPSIZE14X14        2
#define MAPSIZE16X16        3
#define MAPSIZE18X18        4
#define MAPSIZE20X20        5
#define MAPSIZE22X22        6
#define MAPSIZE24X24        7
#define MAPSIZE26X26        8
#define MAPSIZE32X32        9
#define MAPSIZE36X36        10
#define MAPSIZE40X40        11
#define MAPSIZE44X44        12
#define MAPSIZE48X48        13
#define MAPSIZE52X52        14
#define MAPSIZE64X64        15
#define MAPSIZE72X72        16
#define MAPSIZE80X80        17
#define MAPSIZE88X88        18
#define MAPSIZE96X96        19
#define MAPSIZE104X104      20
#define MAPSIZE120X120      21
#define MAPSIZE132X132      22
#define MAPSIZE144X144      23
#define MAPSIZE8X18         24
#define MAPSIZE8X32         25
#define MAPSIZE12X26        26
#define MAPSIZE12X36        27
#define MAPSIZE16X36        28
#define MAPSIZE16X48        29
```

2.7.2 Data structure

The following data structure define the properties of the DataMatrix barcode, it can be transfered into function as parameter.

```
typedef unsigned int COLORREF;
typedef unsigned char BYTE;

typedef struct _tagDATAMATRIXCONTEXT
{
```



```
    char code[3300];
    int size;
    int encoding;
            //0: EncodeAscii;
            //1: EncodeC40;
            //2: EncodeText;
            //3: EncodeBase256;
            //4: EncodeX12;
            //5: EncodeEdifact
    int dotPixel;
    int sizePattern;
    int nMargin;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _DATAMATRIXCONTEXT,*_LPDATAMATRIXCONTEXT;
```

2.7.3. Function or procedure

2.7.3.1. _InitDataMatrixContext

The _InitDataMatrixContext function initilize the environment of DataMatrix encoding with default value.

void __stdcall _InitDataMatrixContext(_DATAMATRIXCONTEXT *pDmCtx);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _PDATAMATRIXCONTEXT

Return values

None

2.7.3.2. _DataMatrixEncode2Bitmap

The DataMatrixEncode2Bitmap function encode the data inputed with the defined attributes and save the barcode to an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _DataMatrixEncode2Bitmap(_LPDATAMATRIXCONTEXT pDmCtx, int *pWidth, int *pHeight);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _DATAMATRIXCONTEXT

pWidth

[in/out] return the width of the bitmap buffer output

pHeight

[in/out] return the height of the bitmap buffer output

Return values

If the function succeeds, the return value are RGB bitmap buffer of DataMatrix barcode , pWidth \pHeight; otherwise , return NULL.

2.7.3.3. _DataMatrixEncodeRegister

The _DataMatrixEncodeRegister function initilize the environment of DataMatrix encoding with default value.

bool _DataMatrixEncodeRegister(char *pMailBox,char *pRegCode);

Parameters

pMailBox: Mail box used to generate the regcode

pRegCode: regcode generated with mail box string

Return values

Return TRUE if register the product successfully, otherwise return FALSE .

2.7.4. Example for C/C++

2.7.4.1 Example1

In this example, there we declared a function named save_png_to_file() which inputs the bitmap-buffer we get before and outputs the right result to a file named “fruit.png”

```
/* Given "bitmap", this returns the pixel of bitmap at the point

   ("x", "y"). */
```



```
static pixel_t * pixel_at (unsigned char *pRgb,int width, int x, int y)
{
    return (pixel_t*)(pRgb + width * y * 3 + x * 3);
}
/* Write "bitmap" to a PNG file specified by "path"; returns 0 on
   success, non-zero on error. */
static int save_png_to_file (unsigned char *pRgb,int width,int height, const char *path)
{
    FILE * fp;
    png_structp png_ptr = NULL;
    png_infop info_ptr = NULL;
    size_t x, y;
    png_byte ** row_pointers = NULL;
    /* "status" contains the return value of this function. At first
       it is set to a value which means 'failure'. When the routine
       has finished its work, it is set to a value which means
       'success'. */
    int status = -1;
    /* The following number is set by trial and error only. I cannot
       see where it it is documented in the libpng manual.
    */
    int pixel_size = 3;
    int depth = 8;
    fp = fopen (path, "wb");
```



```
    if (! fp) {
        goto fopen_failed;
    }
    png_ptr = png_create_write_struct (PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
    if (png_ptr == NULL) {
        goto png_create_write_struct_failed;
    }
    info_ptr = png_create_info_struct (png_ptr);
    if (info_ptr == NULL) {
        goto png_create_info_struct_failed;
    }
    /* Set up error handling. */
    if (setjmp (png_jmpbuf (png_ptr))) {
        goto png_failure;
    }
    /* Set image attributes. */
    png_set_IHDR (png_ptr, info_ptr,width, height, depth, PNG_COLOR_TYPE_RGB,
PNG_INTERLACE_NONE, PNG_COMPRESSION_TYPE_DEFAULT,
PNG_FILTER_TYPE_DEFAULT);
    /* Initialize rows of PNG. */
    row_pointers = (png_byte **)png_malloc (png_ptr, height * sizeof (png_byte *));
    for (y = 0; y < height; ++y) {
        png_byte *row =(png_byte *)
            png_malloc (png_ptr, sizeof (uint8_t) * width * pixel_size);
```



```
            row_pointers[y] = row;
            for (x = 0; x < width; ++x) {
                pixel_t * pixel = pixel_at(pRgb,width, x, y);
                *row++ = pixel->red;
                *row++ = pixel->green;
                *row++ = pixel->blue;
            }
        }
        /* Write the image data to "fp". */
        png_init_io (png_ptr, fp);
        png_set_rows (png_ptr, info_ptr, row_pointers);
        png_write_png (png_ptr, info_ptr, PNG_TRANSFORM_IDENTITY, NULL);
        /* The routine has successfully written the file, so we set
           "status" to a value which indicates success. */
        status = 0;
        for (y = 0; y < height; y++) {
            png_free (png_ptr, row_pointers[y]);
        }
        png_free (png_ptr, row_pointers);
    png_failure:
     png_create_info_struct_failed:
        png_destroy_write_struct (&png_ptr, &info_ptr);
     png_create_write_struct_failed:
        fclose (fp);
```



```
 fopen_failed:

    return status;

}
```

2.7.4.2 Source code

```
#include "DataMatrixEncodeLib.h"

....
    _tagDATAMATRIXCONTEXT tDmCtx;

    int width,height;

    unsigned char *pMap;


    _InitDataMatrixContext(&tDmCtx);
    tDmCtx.encoding = 0;
    tDmCtx.dotPixel = 4;
    tDmCtx.sizePattern = -1;    //auto size pattern
    tDmCtx.nMargin = 10;
    tDmCtx.clBackGround = RGB(255,255,255);
    tDmCtx.clForeGround = RGB(255,0,0);
    sprintf(tDmCtx.code,"http://www.aipsys.com");
    tDmCtx.size = strlen(tDmCtx.code);
    pMap =_DataMatrixEncode2Bitmap (&tDmCtx, "c:\\dm.bmp");

    if(!pMap) return;

    /* Write the image to a file 'fruit.png' by the function we declared. */

    save_png_to_file (pMap,width,height, "fruit.png");

    free(pMap);
    _FreeDataMatrixContext();
.....
```

LIBRARY for linking

DataMatrixEncodeLIBLinuxARM.lib

2.8. Library for MAC

2.8.1 Constants

Size pattern constant of DataMatrix encoding

```
//constant for sizePattern of datamatrix image.
#define SQUAREAUTOSIZE          -1
#define RECTANGLEAUTOSIZE       -2

#define MAPSIZE10X10        0
#define MAPSIZE12X12        1
#define MAPSIZE14X14        2
#define MAPSIZE16X16        3
#define MAPSIZE18X18        4
#define MAPSIZE20X20        5
#define MAPSIZE22X22        6
#define MAPSIZE24X24        7
#define MAPSIZE26X26        8
#define MAPSIZE32X32        9
#define MAPSIZE36X36        10
#define MAPSIZE40X40        11
#define MAPSIZE44X44        12
#define MAPSIZE48X48        13
#define MAPSIZE52X52        14
#define MAPSIZE64X64        15
#define MAPSIZE72X72        16
#define MAPSIZE80X80        17
#define MAPSIZE88X88        18
#define MAPSIZE96X96        19
#define MAPSIZE104X104      20
#define MAPSIZE120X120      21
#define MAPSIZE132X132      22
#define MAPSIZE144X144      23
#define MAPSIZE8X18         24
#define MAPSIZE8X32         25
#define MAPSIZE12X26        26
#define MAPSIZE12X36        27
#define MAPSIZE16X36        28
#define MAPSIZE16X48        29
```


2.8.2 Data structure

The following data structure define the properties of the DataMatrix barcode, it can be transfered into function as parameter.

```
typedef unsigned int COLORREF;
typedef unsigned char BYTE;

typedef struct _tagDATAMATRIXCONTEXT
{
    char code[3300];
    int size;
    int encoding;
            //0: EncodeAscii;
            //1: EncodeC40;
            //2: EncodeText;
            //3: EncodeBase256;
            //4: EncodeX12;
            //5: EncodeEdifact
    int dotPixel;
    int sizePattern;
    int nMargin;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _DATAMATRIXCONTEXT,*_LPDATAMATRIXCONTEXT;
```

2.8.3. Function or procedure

2.8.3.1. _InitDataMatrixContext

The _InitDataMatrixContext function initilize the environment of DataMatrix encoding with default value.

void __stdcall _InitDataMatrixContext(_DATAMATRIXCONTEXT *pDmCtx);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _PDATAMATRIXCONTEXT

Return values

None

2.8.3.2. _DataMatrixEncode2Bitmap

The DataMatrixEncode2Bitmap function encode the data inputed with the defined attributes and save the barcode to an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _DataMatrixEncode2Bitmap(_LPDATAMATRIXCONTEXT pDmCtx, int *pWidth, int *pHeight);

Parameters

pDmCtx

[in] define the DataMatrix attributes for encoding, refer structure type _DATAMATRIXCONTEXT

pWidth

[in/out] return the width of the bitmap buffer output

pHeight

[in/out] return the height of the bitmap buffer output

Return values

If the function succeeds, the return value are RGB bitmap buffer of DataMatrix barcode , pWidth \pHeight; otherwise , return NULL.

2.8.3.3. _DataMatrixEncodeRegister

The _DataMatrixEncodeRegister function initilize the environment of DataMatrix encoding with default value.

bool _DataMatrixEncodeRegister(char *pMailBox,char *pRegCode);

Parameters

pMailBox: Mail box used to generate the regcode

pRegCode: regcode generated with mail box string

Return values

Return TRUE if register the product successfully, otherwise return FALSE .

2.8.4. Example for Object C

2.8.4.1 Example1

In this example, there we declared a function named save_png_to_file() which inputs the bitmap-buffer we get before and outputs the right result to a file named “fruit.png”

```
/* Given "bitmap", this returns the pixel of bitmap at the point
    ("x", "y"). */
static pixel_t * pixel_at (unsigned char *pRgb,int width, int x, int y)
{
     return (pixel_t*)(pRgb + width * y * 3 + x * 3);
}
/* Write "bitmap" to a PNG file specified by "path"; returns 0 on
    success, non-zero on error. */
static int save_png_to_file (unsigned char *pRgb,int width,int height, const char *path)
{
     FILE * fp;
     png_structp png_ptr = NULL;
     png_infop info_ptr = NULL;
     size_t x, y;
     png_byte ** row_pointers = NULL;
     /* "status" contains the return value of this function. At first
         it is set to a value which means 'failure'. When the routine
         has finished its work, it is set to a value which means
         'success'. */
```



```
    int status = -1;
    /* The following number is set by trial and error only. I cannot
       see where it it is documented in the libpng manual.
    */
    int pixel_size = 3;
    int depth = 8;
    fp = fopen (path, "wb");
    if (! fp) {
        goto fopen_failed;
    }
    png_ptr = png_create_write_struct (PNG_LIBPNG_VER_STRING, NULL, NULL,
NULL);
    if (png_ptr == NULL) {
        goto png_create_write_struct_failed;
    }
    info_ptr = png_create_info_struct (png_ptr);
    if (info_ptr == NULL) {
        goto png_create_info_struct_failed;
    }
    /* Set up error handling. */
    if (setjmp (png_jmpbuf (png_ptr))) {
        goto png_failure;
    }
    /* Set image attributes. */
```



```
    png_set_IHDR (png_ptr, info_ptr,width, height, depth, PNG_COLOR_TYPE_RGB,
PNG_INTERLACE_NONE, PNG_COMPRESSION_TYPE_DEFAULT,
PNG_FILTER_TYPE_DEFAULT);

    /* Initialize rows of PNG. */

    row_pointers = (png_byte **)png_malloc (png_ptr, height * sizeof (png_byte *));

    for (y = 0; y < height; ++y) {

        png_byte *row =(png_byte *)

            png_malloc (png_ptr, sizeof (uint8_t) * width * pixel_size);

        row_pointers[y] = row;

        for (x = 0; x < width; ++x) {

            pixel_t * pixel = pixel_at(pRgb,width, x, y);

            *row++ = pixel->red;

            *row++ = pixel->green;

            *row++ = pixel->blue;

        }

    }

    /* Write the image data to "fp". */

    png_init_io (png_ptr, fp);

    png_set_rows (png_ptr, info_ptr, row_pointers);

    png_write_png (png_ptr, info_ptr, PNG_TRANSFORM_IDENTITY, NULL);

    /* The routine has successfully written the file, so we set

       "status" to a value which indicates success. */

    status = 0;

    for (y = 0; y < height; y++) {

        png_free (png_ptr, row_pointers[y]);
```



```
    }

    png_free (png_ptr, row_pointers);

png_failure:

 png_create_info_struct_failed:

    png_destroy_write_struct (&png_ptr, &info_ptr);

 png_create_write_struct_failed:

    fclose (fp);

 fopen_failed:

    return status;

}
```

2.8.4.2 Source code

```
#include "DataMatrixEncodeLib.h"

....
    _tagDATAMATRIXCONTEXT tDmCtx;

    int width,height;

    unsigned char *pMap;


    _InitDataMatrixContext(&tDmCtx);
    tDmCtx.encoding = 0;
    tDmCtx.dotPixel = 4;
    tDmCtx.sizePattern = -1;    //auto size pattern
    tDmCtx.nMargin = 10;
    tDmCtx.clBackGround = RGB(255,255,255);
    tDmCtx.clForeGround = RGB(255,0,0);
    sprintf(tDmCtx.code,"http://www.aipsys.com");
    tDmCtx.size = strlen(tDmCtx.code);
    pMap =_DataMatrixEncode2Bitmap (&tDmCtx, "c:\\dm.bmp");

    if(!pMap) return;
```



```
        /* Write the image to a file 'fruit.png' by the function we declared. */

        save_png_to_file (pMap,width,height, "fruit.png");

        free(pMap);
        _FreeDataMatrixContext();
    .....
```

LIBRARY for linking

DataMatrixEncodeLIBLinux.lib

3. Decoder SDK

3.1. Static link library

3.1.1. Function or procedure

3.1.1.1. _DataMatrixDecodeImageFile

The _DataMatrixDecodeImageFile function read DataMatrix figure from image and decode it to text or binary data.

Result* __stdcall _DataMatrixDecodeImageFile(LPCTSTR lpImageFile);

Parameters

lpImageFile LPCTSTR

[in] the image file containing DataMatrix figure, it can be BMP,GIF,PNG,JPG or TIF formats.

Return values

Result * when decode success, the decoded data written in Result structure and return.

NULL decode failure.

3.1.1.3. _DataMatrixDecodeBitmap

The _DataMatrixDecodeBitmap function read DataMatrix figure from image opened and decode it to text or binary data.

Result* __stdcall _ _DataMatrixDecodeBitmap (HBITMAP hImage);

Parameters

hImage HBITMAP
[in] the bitmap handle containing DataMatrix figure,

Return values

Result * when decode success, the decoded data written in Result structure and return.
NULL decode failure.

3.1.1.2. _DataMatrixDecodeGrayImage

The _DataMatrixDecodeGrayImage function read DataMatrix figure from gray image buffer and decode it to text or binary data.

Result* __stdcall _DataMatrixDecodeGrayImage (BYTE *pGray, int width,int height);

Parameters

BYTE *pGray
[in] the gray image buffer containing DataMatrix figure,
int width
[in] width of image
int height,
[in] height of image

Return values

Result * when decode success, the decoded data written in Result structure and return.
NULL decode failure.

3.1.1.4. _DatamatrixFree

The _DatamatrixFree function free the result buffer after decoding barcode sucessfully.

void __stdcall _DatamatrixFree(Result *r);

Parameters

Result *r

[in] the result buffer.

Return values

void

3.1.1.5. _DMReaderRegister

The _DMReaderRegister function can register your license.

BOOL __stdcall _DMReaderRegister(char *pMailBox,char *pRegCode);

Parameters

char *pMailBox

[in] user mailbox or domain.

char *pRegCode

[in] the register code

. **Return values**

TRUE if license is OK

FALSE **if the license is invalid**

3.1.2. Samples

3.1.2.1 Example for Microsoft visual C++

```
#include "stdafx.h"
#include "DatamatrixDecodeLib.h"
```



```
int main(int argc, char* argv[])
{
    Result *r = _DataMatrixDecodeImageFile("c:\\test.bmp");
    if (r)
    {
        printf("%s \n",r->pData);
        _DatamatrixFree(r);
    }
    return 0;
}
```

Static library linked:

DataMatrixDecodeLib.LIB

3.2. Dynamic link library

3.2.1. Function or procedure

3.2.1.1. DataMatrixDecodeImageFile

The DataMatrixDecodeImageFile function read DataMatrix figure from image and decode it to text or binary data.

Result* __stdcall DataMatrixDecodeImageFile(LPCTSTR lpImageFile);

Parameters

lpImageFile LPCTSTR

[in] the image file containing DataMatrix figure, it can be BMP,GIF,PNG,JPG or TIF formats.

Return values

Result * when decode success, the decoded data written in Result structure and return.

NULL decode failure.

3.2.1.2. _DataMatrixDecodeBitmap

The DataMatrixDecodeBitmap function read DataMatrix figure from image opened and decode it to text or binary data.

Result* __stdcall DataMatrixDecodeBitmap (HBITMAP hImage);

Parameters

hImage HBITMAP
[in] the bitmap handle containing DataMatrix figure,

Return values

Result * when decode success, the decoded data written in Result structure and return.
NULL decode failure.

3.2.1.3. DataMatrixDecodeGrayImage

The DataMatrixDecodeGrayImage function read DataMatrix figure from gray image buffer and decode it to text or binary data.

Result* __stdcall DataMatrixDecodeGrayImage (BYTE *pGray, int width,int height);

Parameters

BYTE *pGray
[in] the gray image buffer containing DataMatrix figure,
int width
[in] width of image
int height,
[in] height of image

Return values

Result * when decode success, the decoded data written in Result structure and return.
NULL decode failure.

3.2.1.4. DatamatrixFree

The _DatamatrixFree function free the result buffer after decoding barcode sucessfully.

void __stdcall _DatamatrixFree(Result *r);

Parameters

Result *r

[in] the result buffer.

Return values

void

3.2.1.5. DMReaderRegister

The DMReaderRegister function can register your license.

BOOL __stdcall DMReaderRegister(char *pMailBox,char *pRegCode);

Parameters

char *pMailBox

[in] user mailbox or domain.

char *pRegCode

[in] the register code

. **Return values**

TRUE if license is OK

FALSE **if the license is invalid**

3.2.2. Samples

3.2.2.1 Example for Microsoft visual C++

```
#include "stdafx.h"
#include "DatamatrixDecodeDll.h"

int main(int argc, char* argv[])
{
     Result *r = DataMatrixDecodeImageFile("c:\\test.bmp");
     if (r)
     {
```



```
            printf("%s \n",r->pData);
            _DatamatrixFree(r);
        }
        return 0;
}
```

static library linked:

DataMatrixDecodeDLL.LIB

Dynamic library at runtime:

DataMatrixDecodeDLL.DLL

3.3. SDK for Windows Phone7

3.3.1 Interface

```
public class Result
{
    public override System.String ToString();
    virtual public int X
    virtual public int Y
    virtual public sbyte[] Value
}

public class Scanner
{
        public bool RegisterDatamatrixReader(String strMail, String strRegCode);
        public Result DecodeFromImageFile(System.String file);
        public Result DecodeFromImage(BitmapImage im);
        public Result DecodeFromBitmap(sbyte[][] map);
}
```

3.3.2 Sample

```
using aipsys.barcode.datamatrix.decoder;
.....

                Uri uri = new Uri("/ImageTest;component/2.JPG", UriKind.Relative);
```



```
BitmapImage bitmapImage = new BitmapImage();
bitmapImage.CreateOptions = BitmapCreateOptions.None;
bitmapImage.UriSource = uri;

WriteableBitmap bmp = new WriteableBitmap(bitmapImage);
Scanner scanner = new Scanner();
Result r = scanner.DecodeFromImage(bitmapImage);
if (r != null )
    textBox1.Text = r.ToString();
else
     textBox1.Text = "Can't detected";
```

.....

3.3.3. Library

DatamatrixDecodeWP7.dll

3.4. SDK for Android

3.4.1. Interface

```
public class DatamatrixReader {

  static {

    System.loadLibrary("DatamatrixReader");

    Boolean bool =  RegisterDecoder( "yourmailbox", "yuor register code");

  }

    public native static byte[] decodeRgbImage(int[]rgb,int width,int height,int
nTimeOut);

    public native static byte[] decodeSubGrayImage(byte []gray,int width,int
height,int x1,int y1,int x2,int y2,int nTimeOut);

    //where x1,y1,x2,y2 is the clip rectangle area.
```



```
    public native static byte[] decodeGrayImage(byte []gray,,int width,int
height,int nTimeOut)
    public native static boolean RegisterDecoder(String mailBox, String
regCode);
 }
```

3.4.2. Sample

```
#import com.aipsys.DatamatrixReader;

DatamatrixReader nativeLib;
…..
nativeLib. decodeRgbImage (rgb, 480, 320, 20000);
….
```

3.4.3. Library

libDatamatrixReader.so

3.5. SDK for iPhone platform

3.5.1. Interface

```
  typedef struct tagPoint
  {
    int x;
    int y;
  }POINT;
```



```
typedef struct tagResult
{
  char sBarcodeType[32];
  unsigned char *pData;
  int nSize;
  POINT ptsBarcode[4];
}Result;
Result *DecodeGrayImage(char *rgb,int width,int height,int nTimeOut);
Result *DecodeSubGrayImage(char *rgb,int width,int height,int x1,int y1,int
x2,int y2,int nTimeOut);
Result *DecodeRgbImage(char *rgb,int width,int height,int nTimeOut);
void   FreeResult(Result *pResult, int nCount);
int RegisterDecoder(char *mailBox, char *regCode);
```

3.5.2. Samples

```
#include “DatamatrixReader.h”
…….
int nTimeout = 1000;

Result *pResult = DecodeGrayImage(subsetData,subsetWidth,subsetHeight,nTimeout);

if(pResult)
{
      char *temp = (char *)malloc(pResult->nSize+1);
      memcpy(temp, pResult->pData, pResult->nSize);
      ….
      Free(temp);
```



```
}
```

3.5.3. Library

libDatamatrixReader.a

3.6. SDK for Blackberry

3.6.1. Interface

```
public class Result
{
   public Result(byte[] v, int x, int y)
   public final byte[] getValue()
   public final int getX()
   public final int getY()
   public final String toString()
}
public class Scanner
{
        public final Result DecodeFromImageFile(String file)
        public final Result DecodeFromImage(Bitmap im)
        public final Result DecodeFromImageBitmap(byte map[][])
}
```

3.6.2. Samples

```
import aipsys.barcode.datamatrix.decoder.Result;
import aipsys.barcode.datamatrix.decoder.Scanner;
……..
private Bitmap bitmap;
……….
Result result = scanner.DecodeFromImage(bitmap);
```

3.6.3. Library

dmreader.rar

4. Order Information

Our sales department was outsourced to Shareit Inc. Ordering online is secure, safe, and guaranteed. We provide first-rate, global service to you.

We accept Visa, MasterCard, American Express, JCB, Diners Club, Switch and Solo. Credit card payments are processed within seconds, and clients receive their product or licensing information without delay.

- The *Developer License* allows one developer royalty-free distribution up to 10,000 user licenses.
- The *5 Developer License* grants the rights of the Developer License for up to five (5) developers and 20,000 user licenses.
- The *Unlimited Developer License* grants the rights of the Developer License for an unlimited number of developers and an unlimited number of user licenses.
- The *Small Company Developer License* grants the rights of the Developer License to organizations which a gross annual revenue or funding of less than 2 million U.S. Dollars.
- The *Single User License* allows use of the Software for one (1) user in your organization

Packages	Trial Dwonload	Single User	Small Company Developer	1 Developer	5 Developer	Unlimited Developer	Version
1D Barcode Encode SDK							
Static Library	Free Trial		$495	$990	$2,180	$3,099	1.2
Dynamic Library	Free Trial	$125	$179	$379	$1,090	$2,199	1.2
ActiveX	Free Trial	$125	$179	$379	$1,090	$2,199	1.2
ASP Component	Free Trial		$179	$379	$1,090	$2,199	1.2
QRCode Encode SDK							
Static Library	Free Trial		$495	$990	$2,180	$3,099	1.2
Dynamic Library	Free Trial	$125	$179	$379	$949	$2,199	1.2
ActiveX	Free Trial	$125	$179	$379	$949	$2,199	1.2

ASP Component	Free Trial		$179	$379	$949	$2,199	1.2
DataMatrix Encode SDK							
Static Library	Free Trial		$495	$990	$2,180	$3,099	1.2
Dynamic Library	Free Trial	$125	$179	$379	$949	$2,199	1.2
ActiveX	Free Trial	$125	$179	$379	$949	$2,199	1.2
ASP Component	Free Trial		$179	$379	$949	$2,199	1.2
PDF417 Encode SDK							
Static Library	Free Trial		$495	$990	$2,180	$3,099	1.2
Dynamic Library	Free Trial	$125	$179	$379	$949	$2,199	1.2
ActiveX	Free Trial	$125	$179	$379	$949	$2,199	1.2
ASP Component	Free Trial		$179	$379	$949	$2,199	1.2
Aztec Encode SDK							
Static Library	Free Trial		$495	$990	$2,180	$3,099	1.2
Dynamic Library	Free Trial	$125	$179	$379	$949	$2,199	1.2
ActiveX	Free Trial	$125	$179	$379	$949	$2,199	1.2
ASP Component	Free Trial		$179	$379	$949	$2,199	1.2

5. Affiliate program

You will receive a 10%~40% commission on all orders placed by referrals from your website. More sales higher commissions!

AIPSYS affiliate program allows publishers, resellers, and web site owners to advertise AIPSYS.com products to their users and visitors, and earn 30%. Signup is simple and free, and you can spread the word about these fine products and earn commissions in the process.

What You Can Earn

AIPSYS products range in price from $49.95 to $4000. Many users buy multiple products at a time, and since many users also order additional voices at the time of purchase, the average order size is over $100 per order. You will earn 30% on each order you enable through your advertising of AIPSYS.com Products. Our experience suggests that focused target audiences lead to higher sales, with many customers buying up front, and 1% to 3% of downloaders will purchase.

Step by step instruction

Becoming a AIPSYS.com Affiliate is quick and easy. Our affiliate program is managed by Shareit, the established leader in software affiliate programs.

¤To Get Started, simply click Sign Up as a Shareit affiliate.

¤Complete the form, read the agreement and FAQ;

¤Share*it will check your entry, send us confirmation of your application and, subject to final review, we will activate your affiliate account

¤We'll send you an e-mail containing your affiliate ID – the ID is used on your site to inform Share*it that the order is coming from your affiliate account, e.g.http://www.shareit.com/product.html?cart=1&productid=YYYYYY&languageid=1&affiliateid=XXXXX

(XXXXX should be changed to your ID,YYYYYYshould be product id you affiliate,you can select from the table below);

¤You can combine this link on your site with the logo from our site

¤Please note that you or your customers can choose from 1, 2 or even 3 years support contracts, and up to 99 licenses can be purchased online (prices are available after clicking on "Display volume discount prices" button).

¤When the affiliate relationship is established, you will get an affiliate link. Any sales originating from your link will be credited to you, and you will receive the 10~40% commission. Please note currently we will grant 20% commission rate for new affiliates at ShareIt in order to avoid fraud and chargeback. But we are more than happy to increase your commission rate. Please contact us via sales@aipsys.com directly.

PRODUCT ID TABLE

Packages	Single User	Small Company Developer	1 Developer	5 Developer	Unlimited Developer	Version
1D Barcode Encode SDK						
Static Library		300222295	300222296	300222297	300222298	1.2
Dynamic Library	300222332	300222333	300222334	300222335	300222336	1.2
ActiveX	300222367	300222368	300222369	300222370	300222371	1.2
ASP Component		300222388	300222389	300222390	300222391	1.2
QRCode Encode SDK						
Static Library		300222413	300222284	300222285	300222286	1.2
Dynamic Library	300222309	300222312	300222314	300222317	300222320	1.2

ActiveX	300222343	300222344	300222347	300222350	300222355	1.2
ASP Component		300222376	300222377	300222377	300222379	1.2
DataMatrix Encode SDK						
Static Library		300222291	300222292	300222293	300222294	1.2
Dynamic Library	300222321	300222322	300222324	300222325	300222326	1.2
ActiveX	300222357	300222358	300222359	300222360	300222361	1.2
ASP Component		300222380	300222381	300222382	300222383	1.2
PDF417 Encode SDK						
Static Library		300222280	300222281	300222282	300222283	1.2
Dynamic Library	300222299	300222300	300222301	300222303	300222305	1.2
ActiveX	300222337	300222338	300222339	300222340	300222341	1.2
ASP Component		300222372	300222373	300222374	300222375	1.2
Aztec Encode SDK						
Static Library		300222288	300222414	300222289	300222290	1.2
Dynamic Library	300222323	300222327	300222329	300222330	300222331	1.2
ActiveX	300222362	300222363	300222364	300222365	300222366	1.2
ASP Component		300222384	300222385	300222386	300222387	1.2

6. Support Information

Sales information

If you purchase online please check the Current versions list to ensure you have the latest version. The latest versions are those available on the downloads menu above.

Before you buy, you can download it for evaluation, To buy it please refer price list and place an order, price in other currency shown in order form.

Having other question or requirement about sale, please contact sales service.

Having some technical question or new requirement, please contact our technical support service.

Barcode resources reference information

Introduction to common barcode types

RSS barcodes renamed GS1-DataBar

Recommended sizes for barcodes

Barcode specifications & Standards

- American National Standards Institute
- Automatic Identification Manufacturer's Association
- Automotive Industry Action Group
- British Standards Institution (BSI)
- GS1 (formerly EAN International)
- GS1 UK (formerly the e-Centre)
- GS1 US (formerly UCC - Uniform Code Council)
- Health Industry Barcode Standards
- ISO - International Standards Organisation

7. Product Information Link

. QRCode encoder SDK

. PDF417 encoder SDK

. DataMatrix encoder SDK

. Aztec encoder SDK

. Linear 1D barcode encoder SDK