PDF417 Encode SDK
v2.5

USER MANUAL

AIPSYS Software Laboratory

http://www.aipsys.com

Last Updated 18th June 2008

1. Introduction

1.1. PDF417 Barcode introduction

About PDF417

PDF417 s a multi-row, variable-length symbology offering high data capacity and error-correction capability. PDF417 can be scanned by linear PDF417 barcode scanners, rastering laser scanners, or two-dimensional imaging devices. One PDF417 symbol is capable of encoding more than 1100 bytes, 1800 ASCII characters, or 2700 digits, depending on the selected data compaction mode.

Every PDF417 symbol is composed of stack of rows, from a minimum of 3 to a maximum of 90 rows. Each PDF417 row contains start and stop patterns, left and right row indicators, and from one to thirty data symbol characters. Since both the number of rows and their length are selectable when printed, the aspect ratio of a PDF417 symbol can be varied to suit spatial requirements for printing.

A PDF417 symbol character consists of seventeen modules arranged into four bars and four spaces. The entire set of symbol characters is divided into three mutually exclusive encodation sets, or "clusters". Each cluster encodes the 929 available PDF417 symbol character values, or codewords, with distinct bar-spaces patterns so that one cluster cannot be confused with another. Because any two adjacent rows use different clusters, the decoder can utilize data from scans that cross rows while decoding a PDF417 symbol.

By adding error correction codewords to the data message, PDF417 supports correction of lost or missing data. Each PDF417 symbol requires two error correction codewords for error detection. In addition, up to 510 codewords of error correction can be added when printing the symbol. This allows for decoding security that mathematically is many orders of magnitude stronger than that of bar codes with simple check characters.

PDF417 offers three data compaction modes. Each mode defines a conversation or mapping between codeword sequences and byte sequences. The three modes are Text Compaction mode, Byte Compaction mode, and Numeric Compaction mode.

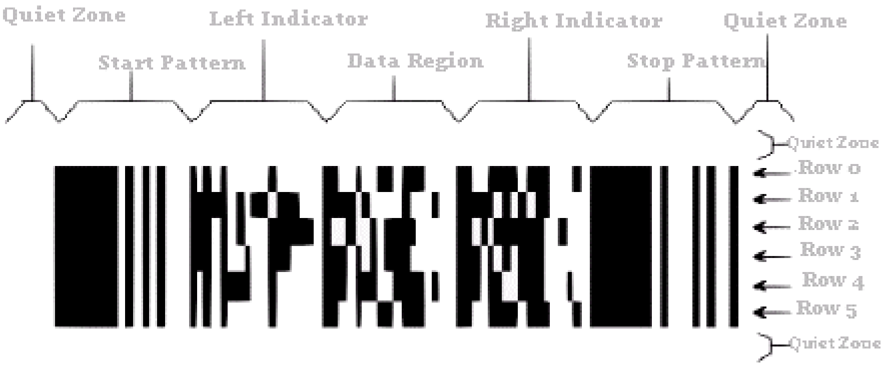
The interpretation of the byte sequences encoded by a compaction mode is determined by the Global Label Identifier (GLI). A GLI is a special codeword sequence which activates a set of interpretations. The implementation of GLIs enables PDF417 to encode international character sets, and industry - and user-defined character sets, as well as ASCII.

Marco PDF417 provides a means for creating a distributed representation of files too large to be represented by a single PDF417 symbol. Marco PDF417 symbols differ from ordinary PDF417 symbols in that they contain additional control information used to support this distributed representation. This allows a decoder to make use of this information to correctly reconstruct and verify the file, independent of the symbol scanning order.

In a relatively clean environment in which damage to the label is unlikely, Truncated PDF417 can be used. This version omits the right row indicator and simplifies the stop pattern into a single module width bar. This reduces the non-data overhead, with some trade-off in robustness, or the ability to withstand degradation. Truncated PDF417 is fully reader-compatible with standard PDF417.

Characteristics of PDF417	
Encode Character Set	All 128 ASCII characters All 128 Extended ASCII Characters 8-Bit Binary Data Up to 811,800 Different Character Sets or interpretations
Code Type	Continuous, Multi-row
Character Self Checking	Yes
Symbol Size:	
Height	Variable (3 to 90 Rows)
Width	Variable (90X to 583X)
Bidirectional Decoding	Yes
Error Correction Characters	2 to 512
Maximum Data Characters per Symbol (at error correction level 0)	1850 texts 2710 digits 1108 bytes
Additional Features	Selectable Levels of Error Correction Can Utilize Scans that Cross Rows
Additional Options	Macro PDF417 Global Label Identifiers Truncated PDF417

Symbol Structure



Above Figure shows a typical PDF417 symbol and its structure

Every PDF417 symbol contains a minimum or 3 to a maximum of 90 rows. Each row consists of:

a. Leading quiet zone
b. Start pattern
c. Left row indicator symbol character
d. One to thirty data symbol characters
e. Right row indicator symbol character
f. Stop pattern
g. Trailing quiet zone

A symbol character consists of seventeen modules arranged into four bars and four spaces. each symbol character represents a value in the range of 0 to 928; within this documents, these symbol character value are referred to as "codewords"

Because the number of row is variable, and rows are variable in length (i.e., in the number of symbol character "columns"), the height/width proportion, or aspect ration, of a PDF417 symbol can be varied to suit spatial requirements for printing. However, the number of symbol characters in all rows of a given symbol must be the same.

The data region of a PDF417 symbol is the central area of codeword columns between the left row indicator column and the right row indicator column. The first (upper left) codeword of the data region is the symbol length descriptor. Its value indicates the total number of codewords in the data region, including the symbol length descriptor itself, but excluding the error correction codewords. The remaining codewords in the data region (including the data codewords, pad codewords, and error correction code words, in that order) are arranged with the most significant codeword adjacent to the symbol length descriptor and are read from left to right, top row to bottom. The number of codewords in the data region of a single PDF417 symbol cannot exceed 928.

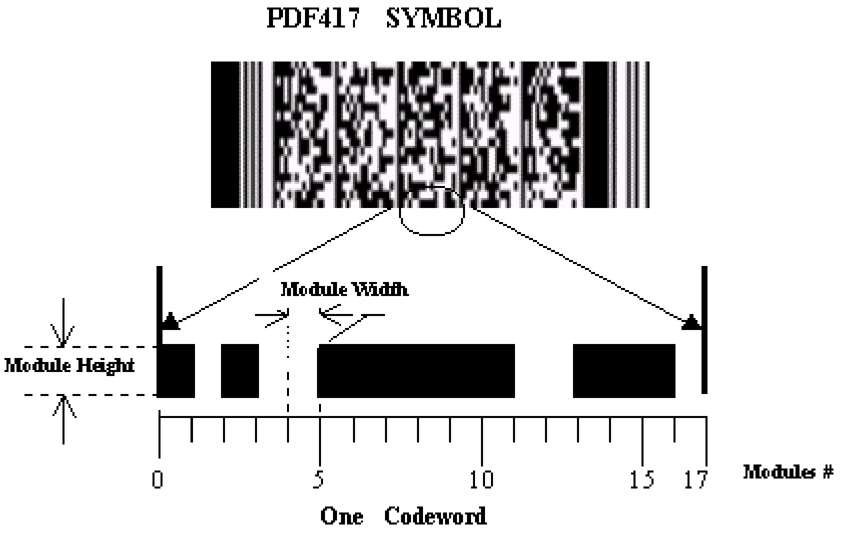
Symbol Character Encodation

Structure

A symbol character consists of four bars and four spaces; each bar or space contains one to six modules. In all cases, the four bars and four spaces of any symbol character measure 17 modules in total. The width of one module is the X dimension of that symbol.

Clusters and Symbol Character Definitions

The entire set of PDF417 symbol characters is divided into three mutually exclusive encodation sets, or "clusters". Each cluster encodes all 929 defined PDF417 codewords with distinct bar and space patterns. Within each cluster, each symbol character is associated with a unique value in the range of 0 to 928; this value is called the symbol character value or codeword.



1.3. MicroPDF417 Barcode introduction

Micro PDF-417 is a 2D, multi-row symbology, derived from and closely based on PDF-417. Micro PDF-417 is designed for applications with a need for improved area efficiency, but without the requirement for PDF-417's maximum data capacity. A limited set of symbol sizes is available, together with a fixed level of error correction for each symbol.

Reduced version of PDF417, smaller, but smaller capacity. Number of error correction codewiords is fixed for each available row/column combination, from 7 to 50 codewords pre symbol.

Symbol size:

- Number of rows : 4 to 44
- Number of cols : 1,2,3 or 4

Data encoding mode	**Maximum capacity**
Text compaction	250 characters
Numeric compaction	366 characters
Byte compaction	150 characters

CW : Codeword, EC : Error correction codewords, RW : Number of non error codewords-really used codewords without error correction codewords

MicroPDF417 characteristics

Number of data columns	Number of rows	Total CW	Number of EC	% EC/CW	Max.data bytes	Max.aplha chars	Max. digits

1	11	11	7	64	3	6	8
1	14	14	7	50	7	12	17
1	17	17	7	41	10	18	26
1	20	20	8	40	13	22	32
1	24	24	8	33	18	30	44
1	28	28	8	29	22	38	55
2	8	16	8	50	8	14	20
2	11	22	9	41	14	24	35
2	14	28	9	32	21	36	52
2	17	34	10	29	27	46	67
2	20	40	11	28	33	56	82
2	23	46	13	28	38	64	93
2	26	52	15	29	43	72	105
3	6	18	12	67	6	10	14
3	8	24	14	58	10	18	26
3	10	30	16	53	15	26	38
3	12	36	18	50	20	34	49
3	15	45	21	47	27	46	67
3	20	60	26	43	39	66	96
3	26	78	32	41	54	90	132
3	32	96	38	40	68	114	167
3	38	114	44	39	82	138	202
3	44	132	50	38	97	162	237
4	4	16	8	50	8	14	20
4	6	24	12	50	13	22	32
4	8	32	14	44	20	34	49
4	10	40	16	40	27	46	67
4	12	48	18	38	34	58	85
4	15	60	21	35	45	76	111
4	20	80	26	33	63	106	155
4	26	104	32	31	85	142	208
4	32	128	38	30	106	178	261
4	38	152	44	29	128	214	331
4	44	176	50	28	150	250	366

1.4. 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. SDK

2.1. Static link library

2.1.1 Data structure

The following data structure define the properties of the PDF417 barcode, it can be transfer into function as parameter.

```
typedef struct _tagPDF417CONTEXT
{
      int nRows;                //define rows of PDF417 barcode
      int nColumns;             // define columns of PDF417 barcode
      int nModuleWidth;         // define module width of PDF417 barcode
      int nModuleHeight;        // define module height of PDF417 barcode
      int nErrorCorrectLevel;// define error correction level of PDF417 barcode
      int nMargin;              // define white margin of the output image
      BOOL bMacroPDFEnabled; // switch for macro pdf417 encoding
      char cFileID[256];        // file id for macr pdf417
      int   nSegmentIndex;    //segment index for macro pdf417
      int   nSegmentCount;    //Total segments
       COLORREF clForeGround;    //define the fore ground color of output image
      COLORREF clBackGround;    //define the fore ground color of output image
      char cData[7000];         //define the data to be encoded
      int   nSize;              //define the data size of the data to be encoded, set it
                                //when the data is binary type
}_PDF417CONTEXT;

typedef struct _tagMICRODPDF417CONTEXT
{
      int nModuleHeight;        //Pixel height to build barcode
      int nModuleWidth;         //Pixel Width to build barcode
      int nRows;                //barcode rows    4~44
      int nColumns;             //barcode columns    1~4
      int nMargin;             //margin size
      COLORREF clForeGround;      //fore ground color
      COLORREF clBackGround;      //back ground color
      char cEncodedData[45][180];    //output of the encoding
      char errtxt[100];        // error message when encoding failed
      int nSize;               // data size
      char cData[368];         // data inputed
```

}_MICROPDF417CONTEXT;

2.1.2. Function or procedure

2.1.2.1. _InitPDF417Context

The _InitWorkSpace function initilize the environment of PDF417 encoding with default value.

void __stdcall _InitPDF417Context **(_PDF417CONTEXT *pPdf417Ctx);**

Parameters

pPdf417Ctx
[in] define the PDF417 attributes for encoding, refer structure type
_PDF417CONTEXT

Return values

None

2.1.2.2. _PDF417Encode2File

The PDF417Encode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall _PDF417Encode2File(_PDF417CONTEXT *pPdf417Ctx,
LPCTSTR lpImageFile);

Parameters

pPdf417Ctx
[in] define the PDF417 attributes for encoding, refer structure type
_PDF417CONTEXT
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.

2.1.2.3. _PDF417Encode2Bitmap

The **PDF417Encode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the PDF417 barcode image

HBITMAP __stdcall _PDF417Encode2Bitmap(
_PDF417CONTEXT *pPdf417Ctx);

Parameters

pPdf417Ctx
[in] define the PDF417 attributes for encoding, refer structure type
_PDF417CONTEXT

Return values
If the function succeeds, the return value is BITMAP handle of PDF417 Barcode, otherwise , return NULL.

2.1.2.4. _ FreePDF417Context

The _FreePDF417Context function free environment of the PDF417 encoding

BOOL __stdcall _ FreePDF417Context **();**

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

2.1.2.5. _PDF417EncodeBinary2File

The _PDF417EncodeBinary2File function encode the binary data inputed with the defined attributes and return save barcode bitmap into specified file

BOOL __stdcall _PDF417EncodeBinary2File(char *pBuf,int nLen,
char *pOutImage, int nRows, int nColumns,
int nECLevel, int nModuleWidth,
int nModuleHeight, int nMargin,
BOOL bMacroPDFEnabled,
char *pFileID,int nSegmentIndex,int nSegmentCount,
COLORREF clFore,COLORREF clBack);

Parameters

pBuf

[in] define the data encoded

nLen

[in] length of data to be encoded

pOutImage

[in] define output file name

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in]define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

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

2.1.2.6. _PDF417EncodeDigit2File

The **_PDF417EncodeDigit2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
BOOL __stdcall _PDF417EncodeDigit2File(
        char *pBuf,char *pOutImage,int nRows, int nColumns,
        int nECLevel,    int nModuleWidth, int nModuleHeight,
        int nMargin, BOOL bMacroPDFEnabled,
        char *pFileID,int nSegmentIndex,int nSegmentCount,
        COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
pOutImage
[in] define output file name
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background color

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

2.1.2.7. _PDF417EncodeHybrid2File

The **_PDF417EncodeDigit2File** function encode the hybrid data composed of char,digit,binary inputed with the defined attributes and return save barcode bitmap into specified file

BOOL __stdcall _PDF417EncodeHybrid2File(char *pBuf,int nLen,
char *pOutImage,int nRows, int nColumns, int nECLevel,
int nModuleWidth, int nModuleHeight,int nMargin,

```
BOOL bMacroPDFEnabled,
char *pFileID,int nSegmentIndex,int nSegmentCount,
 COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
nLen
[in] length of data to be encoded
pOutImage
[in] define output file name
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf 417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define b ackground color

Return values

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

2.1.2.8. _PDF417EncodeText2File

The _**PDF417EncodeText2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
BOOL __stdcall _PDF417EncodeText2File(char *pBuf,
            char *pOutImage,int nRows, int nColumns, int nECLevel,
            int nModuleWidth, int nModuleHeight,
            int nMargin, BOOL bMacroPDFEnabled,
            char *pFileID,
            int nSegmentIndex,int nSegmentCount,
            COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
pOutImage
[in] define output file name
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background col or

Return values

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

2.1.2.9. _PDF417EncodeBinary2Bitmap

The PDF417EncodeBinary2File function encode the binary data inputed with the defined attributes and return bitmap handle of barcode image

```
HBITMAP __stdcall _PDF417EncodeBinary2Bitmap(char *pBuf,
                    int nLen, int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin, BOOL bMacroPDFEnabled,
                    char *pFileID,
                    int nSegmentIndex,int nSegmentCount,
                    COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
nLen
[in] length of data to be encoded
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode , otherwise , return NULL.

2.1.2.10. _PDF417EncodeDigit2Bitmap

The **_PDF417EncodeDigit2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
HBITMAP __stdcall _PDF417EncodeDigit2Bitmap(char *pBuf,
                    int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin, BOOL bMacroPDFEnabled,
                    char *pFileID,
                    int nSegmentIndex,int nSegmentCount,
                    COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf

[in] define the data encoded

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode, otherwise , return NULL.

2.1.2.11. _PDF417EncodeHybrid2Bitmap

The **PDF417EncodeDigit2File** function encode the hybrid data composed of char,digit,binary inputed with the defined attributes and return bitmap handle of barcode image.

```
HBITMAP __stdcall _PDF417EncodeHybrid2Bitmap(char *pBuf,
                    int nLen,int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin, BOOL bMacroPDFEnabled,
                    char *pFileID,
                    int nSegmentIndex,int nSegmentCount,
                    COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf

[in] define the data encoded

nLen

[in] length of data to be encoded

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode ma cro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode image , otherwise , return NULL.

2.1.2.12. _PDF417EncodeText2Bitmap

The **_PDF417EncodeText2File** function encode the text string inputed with the defined attributes and return bitmap handle of barcode image.

HBITMAP __stdcall _PDF417EncodeText2Bitmap(char *pBuf,
int nRows, int nColumns, int nECLevel,
int nModuleWidth, int nModuleHeight,
int nMargin, BOOL bMacroPDFEnabled,
char *pFileID,
int nSegmentIndex,int nSegmentCount,
COLORREF clFore,COLORREF clBack);

Parameters

pBuf

[in] define the data encoded

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode, otherwise , return NULL.

2.1.2.13. _InitMicroPDF417Context

The _InitWorkSpace function initilize the environment of PDF417 encoding with default value.

void __stdcall _InitMicroPDF417Context
(_MICROPDF417CONTEXT *pMicro**Pdf417Ctx);**

Parameters

pMicroPdf417Ctx
[in] define the MicroPDF417 attributes for encoding, refer structure type
_**MICRO**PDF417CONTEXT

Return values

None

2.1.2.14. _MicroPDF417Encode2File

The PDF417Encode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall _MicroPDF417Encode2File
(_MICROPDF417CONTEXT *pMicroPdf417Ctx,
LPCTSTR lpImageFile);

Parameters

pMicroPdf417Ctx
[in] define the MicroPDF417 attributes for encoding, refer structure type
_MICROPDF417CONTEXT
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.

2.1.2.15. _MicroPDF417Encode2Bitmap

The **_MicroPDF417Encode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the MicroPDF417 barcode image

HBITMAP __stdcall _MicroPDF417Encode2Bitmap(
_MICROPDF417CONTEXT *pMicroPdf417Ctx);

Parameters

pPdf417Ctx
[in] define the MicroPDF417 attributes for encoding, refer structure type **_MICRO**PDF417CONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of MicroPDF417 Barcode, otherwise , return NULL.

2.1.2.16. _ FreeMicroPDF417Context

The _FreeMicroPDF417Context function free environment of the MicroPDF417 encoding

BOOL __stdcall _ FreeMicroPDF417Context **();**

Return values

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

2.1.3. Example for Microsoft visual C++

Example1

```
#include "PDF417EncodeLIB.h"

....
_tagPDF417CONTEXT tPdf417Ctx;
_InitPDF417Context(&tPdf417Ctx);
```

```
tPdf417Ctx.nColumns = 10;   //set barcode co;umn 0~n
tPdf417Ctx.nRows = 10;   //set barcode co;umn 0~n
 //set error correction level 0~8
tPdf417Ctx.nErrorCorrectLevel = 0;
tPdf417Ctx.nModuleWidth = 2;
tPdf417Ctx.nModuleHeight = 7;
 tPdf417Ctx.nMargin = 10;   //set margin when generate bitmap
tPdf417Ctx.clForeGround = RGB(255,0,0); //set fore ground color
 // set encoded text
 sprintf(tPdf417Ctx.cData,"http://www.aipsys.com");
 tPdf417Ctx.nSize = 21;
 _PDF417Encode2File(&tPdf417Ctx,"c:\\pdf417.bmp");
 _FreePDF417Context();
.....
```

LIBRARY for linking
PDF417EncodeLIB.lib

Example2

```
#include "PDF417EncodeLIB.h"

    ....
    char sData[1024];
    Memcpy(sData,"\1\2\3\4\5\6\7\8\9\0",10);
    _PDF417EncodeBinary2File(sData,10, "c:\\test.bmp"
                     12, 10, 3,7,2, 20,RGB(255,0,0),RGB(255,255,255));
    ....
```

LIBRARY for linking
PDF417EncodeLIB.lib

Example3

```
#include "PDF417EncodeLIB.h"

    ....
    char sData[1024];
    Memcpy(sData,"\1\2\3\4\5\6\7\8\9\0",10);
    hBarcode   = _PDF417EncodeBinary2Bitmap(sData,10,
                     12, 10, 3,7,2, 20, FALSE,"",0,0,
                     RGB(255,0,0),RGB(255,255,255));
    ....
```

LIBRARY for linking
PDF417EncodeLIB.lib

2.2. Dynamic link library

2.2.1. Data structure

```
typedef struct tagPDF417CONTEXT
{
    int nRows;
    int nColumns;
    int nModuleWidth;
    int nModuleHeight;
    int nErrorCorrectLevel;
    int nMargin;
    COLORREF clForeGround;
    COLORREF clBackGround;
    BOOL bMacroPDFEnabled;
    char cFileID[256];
    int    nSegmentIndex;
    int    nSegmentCount;
    char cData[7000];
    int    nSize;
}PDF417CONTEXT;

typedef struct tagMICRODPDF417CONTEXT
{
    int nModuleHeight;      //Pixel height to build barcode
    int nModuleWidth;      //Pixel Width to build barcode
    int nRows;                  //barcode rows    4~44
    int nColumns;              //barcode columns    1~4
    int nMargin;               //margin size
    COLORREF clForeGround;      //fore ground color
    COLORREF clBackGround;      //back ground color
    char cEncodedData[45][180];    //output of the encoding
    char errtxt[100];        // error message when encoding failed
    int nSize;                  // data size
    char cData[368];        // data inputed
}MICROPDF417CONTEXT;
```

2.2.2. Function or procedure

2.2.2.1. InitWorkSpace

The InitWorkSpace function initilize the environment of PDF417 encoding with default value.

void __stdcall InitWorkSpace(PDF417CONTEXT *pPdf417Ctx);

Parameters

pPdf417Ctx
[in] define the PDF417 attributes for encoding, refer structure type
_PDF417CONTEXT

Return values

None

2.2.2.2. PDF417Encode2File

The PDF417Encode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall PDF417Encode2File(PDF417CONTEXT *pPdf417Ctx,
LPCTSTR lpImageFile);

Parameters

pPdf417Ctx
[in] define the PDF417 attributes for encoding, refer structure type
_PDF417CONTEXT
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. PDF417Encode2Bitmap

The **PDF417Encode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the PDF417 barcode image

HBITMAP __stdcall PDF417Encode2Bitmap(
PDF417CONTEXT *pPdf417Ctx);

Parameters

pPdf417Ctx

[in] define the PDF417 attributes for encoding, refer structure type _PDF417CONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of PDF417 Barcode, otherwise , return NULL.

2.2.2.4. FreeWorkSpace

The FreeWorkSpace function free environment of the PDF417 encoding

BOOL __stdcall FreeWorkSpace();

Return values

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

2.2.25 . PDF417EncodeBinary2File

The PDF417EncodeBinary2File function encode the binary data inputed with the defined attributes and return save barcode bitmap into specified file

BOOL __stdcall PDF417EncodeBinary2File(char *pBuf,int nLen,
char *pOutImage, int nRows, int nColumns,
int nECLevel, int nModuleWidth,
int nModuleHeight, int nMargin,
BOOL bMacroPDFEnabled,
char *pFileID,
int nSegmentIndex,int nSegmentCount,
COLORREF clFore,COLORREF clBack);

Parameters

pBuf

[in] define the data encoded

nLen

[in] length of data to be encoded

pOutImage

[in] define output file name

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

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

2.2.26 . PDF417EncodeDigit2File

The **PDF417EncodeDigit2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
BOOL __stdcall PDF417EncodeDigit2File(
        char *pBuf,char *pOutImage,int nRows, int nColumns,
        int nECLevel,   int nModuleWidth, int nModuleHeight,
        int nMargin, BOOL bMacroPDFEnabled,
        char *pFileID,
        int nSegmentIndex,int nSegmentCount,
        COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf

[in] define the data encoded

pOutImage

[in] define output file name

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barco de image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

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

2.2.27 . PDF417EncodeHybrid2File

The **PDF417EncodeDigit2File** function encode the hybrid data composed of char,digit,binary inputed with the defined attributes and return save barcode bitmap into specified file

BOOL __stdcall PDF417EncodeHybrid2File(char *pBuf,int nLen,
char *pOutImage,int nRows, int nColumns, int nECLevel,
int nModuleWidth, int nModuleHeight,int nMargin,
BOOL bMacroPDFEnabled,
char *pFileID,
int nSegmentIndex,int nSegmentCount,

COLORREF clFore,COLORREF clBack);

Parameters

pBuf
[in] define the data encoded
nLen
[in] length of data to be encoded
pOutImage
[in] define output file name
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background color

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

2.2.28 . PDF417EncodeText2File

The **PDF417EncodeText2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

BOOL __stdcall PDF417EncodeText2File(char *pBuf,

```
                    char *pOutImage,int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin, BOOL bMacroPDFEnabled,
                    char *pFileID,
                     int nSegmentIndex,int nSegmentCount,
                    COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
pOutImage
[in] define output file name
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background color

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

2.2.29 . PDF417EncodeBinary2Bitmap

The PDF417EncodeBinary2File function encode the binary data inputed with the defined attributes and return bitmap handle of barcode image

```
HBITMAP __stdcall PDF417EncodeBinary2Bitmap(char *pBuf,
                    int nLen, int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin, BOOL bMacroPDFEnabled,
                    char *pFileID,
                    int nSegmentIndex,int nSegmentCount,
                    COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
nLen
[in] length of data to be encoded
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in] define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]define foreground color
clBack
[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode , otherwise , return NULL.

2.2.210 . PDF417EncodeDigit2Bitmap

The **PDF417EncodeDigit2File** function encode the digital string inputed with the defined attributes and return save barcode bitmap into specified file

```
HBITMAP __stdcall PDF417EncodeDigit2Bitmap(char *pBuf,
                     int nRows, int nColumns, int nECLevel,
                     int nModuleWidth, int nModuleHeight,
                     int nMargin, BOOL bMacroPDFEnabled,
                     char *pFileID,
                     int nSegmentIndex,int nSegmentCount,
                     COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf
[in] define the data encoded
nRows
[in] define rows of PDF417 barcode
nColumns
[in] define columns of PDF417 barcode
nECLevel
[in] define error correction level of PDF417 barcode
nModuleWidth
[in] define module width of PDF417 barcode
nModuleHeight
[in] define module height of PDF417 barcode
nMargin
[in]define margin of output barcode image
bMacroPDFEnabled
[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417
pFileID
[in] file id for macro PDF417 barcode
nSegmentIndex
[in] Segment index for macro PDF417 barcode
nSegmentCount
[in] Total segments
clFore
[in]defin e foreground color
clBack
[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode, otherwise , return NULL.

2.2.211 . PDF417EncodeHybrid2Bitmap

The **PDF417EncodeDigit2File** function encode the hybrid data composed of char,digit,binary inputed with the defined attributes and return bitmap handle of barcode image.

```
HBITMAP __stdcall PDF417EncodeHybrid2Bitmap(char *pBuf,
                    int nLen,int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin, BOOL bMacroPDFEnabled,
                    char *pFileID,
                    int nSegmentIndex,int nSegmentCount,
                    COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf

[in] define the data encoded

nLen

[in] length of data to be encoded

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode image , otherwise , return NULL.

2.2.212 . PDF417EncodeText2Bitmap

The **PDF417EncodeText2File** function encode the text string inputed with the defined attributes and return bitmap handle of barcode image.

```
HBITMAP __stdcall PDF417EncodeText2Bitmap(char *pBuf,
                    int nRows, int nColumns, int nECLevel,
                    int nModuleWidth, int nModuleHeight,
                    int nMargin,COLORREF clFore,COLORREF clBack);
```

Parameters

pBuf

[in] define the data encoded

nRows

[in] define rows of PDF417 barcode

nColumns

[in] define columns of PDF417 barcode

nECLevel

[in] define error correction level of PDF417 barcode

nModuleWidth

[in] define module width of PDF417 barcode

nModuleHeight

[in] define module height of PDF417 barcode

nMargin

[in]define margin of output barcode image

bMacroPDFEnabled

[in]switch for macro pdf417, set it t true if hoping to encode macro pdf417

pFileID

[in] file id for macro PDF417 barcode

nSegmentIndex

[in] Segment index for macro PDF417 barcode

nSegmentCount

[in] Total segments

clFore

[in]define foreground color

clBack

[in]define background color

Return values

If the function succeeds, the return bitmap handle of barcode, otherwise , return NULL.

2.2.2.13. InitMicroPDF417Context

The **InitMicroPDF417Context** function initilize the environment of MicroPDF417 encoding with default value.

void __stdcall InitMicroPDF417Context
(MICROPDF417CONTEXT *pMicroPdf417Ctx);

Parameters

pMicroPdf417Ctx
[in] define the MicroPDF417 attributes for encoding, refer structure type
_MICROPDF417CONTEXT

Return values

None

2.2.214 . MicroPDF417Encode2File

The MicroPDF417Encode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall MicroPDF417Encode2File
(MICROPDF417CONTEXT *pMicroPdf417Ctx,
LPCTSTR lpImageFile);

Parameters

pMicroPdf417Ctx
[in] define the MicroPDF417 attributes for encoding, refer structure type
_MICROPDF417CONTEXT
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.

2.2.215 . MicroPDF417Encode2Bitmap

The **MicroPDF417Encode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the MicroPDF417 barcode image

HBITMAP __stdcall MicroPDF417Encode2Bitmap(
MICROPDF417CONTEXT *pMicroPdf417Ctx);

Parameters

pMicroPdf417Ctx
[in] define the MicroPDF417 attributes for encoding, refer structure type _MICROPDF417CONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of MicroPDF417 Barcode, otherwise , return NULL.

2.2.216 . FreeMicroPDF417Context

The FreeMicroPDF417Context function free environment of the PDF417 encoding

BOOL __stdcall FreeMicroPDF417Context ();

Return values

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

2.2.3. Example for Microsoft visual C++

Example1

```
#include "PDF417EncodeDLL.h"

....
tagPDF417CONTEXT tPdf417Ctx;
InitPDF417Context(&tPdf417Ctx);
tPdf417Ctx.nColumns = 10;   //set barcode co;umn 0~n
tPdf417Ctx.nRows = 10;   //set barcode co;umn 0~n
 //set error correction level 0~8
tPdf417Ctx.nErrorCorrectLevel = 0;
tPdf417Ctx.nModuleWidth = 2;
tPdf417Ctx.nModuleHeight = 7;
```

```
 tPdf417Ctx.nMargin = 10;    //set margin when generate bitmap
tPdf417Ctx.clForeGround = RGB(255,0,0); //set fore ground color
 // set encoded text
 sprintf(tPdf417Ctx.cData,"http://www.aipsys.com");
 tPdf417Ctx.nSize = 21;
 PDF417Encode2File(&tPdf417Ctx,"c:\\pdf417.bmp");
 FreePDF417Context();
.....
```

LIBRARY for linking

PDF417EncodeDLL.lib

Runtime library

PDF417EncodeDLL.DLL

Example2

```
#include "PDF417EncodeDLL.h"

    ....
    char sData[1024];
    Memcpy(sData,"\1\2\3\4\5\6\7\8\9\0",10);
    PDF417EncodeBinary2File(sData,10, "c:\\test.bmp"
                    12, 10, 3,7,2, 20,FALSE,"",0,0
                    RGB(255,0,0),RGB(255,255,255));
    ....
```

LIBRARY for linking

PDF417EncodeLIB.lib

Runtime library

PDF417EncodeDLL.DLL

Example3

```
#include "PDF417EncodeDLL.h"

    ....
    char sData[1024];
    Memcpy(sData,"\1\2\3\4\5\6\7\8\9\0",10);
    hBarcode   =   PDF417EncodeBinary2Bitmap(sData,10,
                    12, 10, 3,7,2, 20,FALSE,"",0,0,
                    RGB(255,0,0),RGB(255,255,255));
    ....
```

LIBRARY for linking

PDF417EncodeLIB.lib

Runtime library
PDF417EncodeDLL.DLL

2.2.4. Example for Borland Delphi

2.2.4.1. Redeclaration of the data type and function

```
type
LPPDF417CONTEXT = ^TPDF417CONTEXT;
TPDF417CONTEXT = record
    nRows : integer;
    nColumns : integer;
    nModuleWidth : integer;
   nModuleHeight : integer;
    nErrorCorrectLevel : integer;
    nMargin : integer;
    clForeGround : TColor;
   clBackGround : TColor;
    bMacroPDFEnabled: boolean; // switch for macro pdf417 encoding
    cFileID: array[1..256] of char;        // file id for macr pdf417
    nSegmentIndex:Integer;    //segment index for macro pdf417
    nSegmentCount:Integer;    //Total segments
    cData : array [0..7000] of char;
   nSize : integer;
end;

procedure InitWorkSpace(pPdf417Ctx : LPPDF417CONTEXT ); stdcall;
                  external 'PDF417ENCODEDLL.DLL';

function    PDF417Encode2File(pPdf417Ctx : LPPDF417CONTEXT;lpImageFile :PChar) :
                  boolean; stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417Encode2Bitmap(pPdf417Ctx : LPPDF417CONTEXT) : HBITMAP;
                  stdcall;external 'PDF417ENCODEDLL.DLL';

function    FreeWorkSpace : boolean;stdcall external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeBinary2File(pBuf: PChar;nLen : Integer; pOutImage : PChar;
        nRows : Integer; nColumns:Integer; nECLevel:Integer; nModuleWidth : Integer;
        nModuleHeight:Integer;nMargin:integer;bMacroPDFEnabled: Boolean;
        pFileID:PChar;nSegmentIndex:Integer;nSegmentCount; clFore : TColor; clBack :
```

```
      TColor ):
      boolean;stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeDigit2File(pBuf: PChar; pOutImage : PChar; nRows : Integer;
      nColumns:Integer; nECLevel:Integer; nModuleWidth : Integer;
      nModuleHeight:Integer;nMargin:integer; bMacroPDFEnabled: Boolean;
      pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
      boolean;stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeHybrid2File(pBuf: PChar;nLen : Integer; pOutImage : PChar;
      nRows : Integer; nColumns:Integer; nECLevel:Integer; nModuleWidth : Integer;
      nModuleHeight:Integer;nMargin:integer; bMacroPDFEnabled: Boolean;
     pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
     boolean;
      stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeText2File(pBuf: PChar; pOutImage : PChar; nRows : Integer;
       nColumns:Integer; nECLevel:Integer; nModuleWidth : Integer;
       nModuleHeight:Integer;nMargin:integer; bMacroPDFEnabled: Boolean;
      pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
      boolean;
       stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeBinary2Bitmap(pBuf: PChar;nLen : Integer; nRows : Integer;
       nColumns:Integer; nECLevel:Integer; nModuleWidth : Integer;
       nModuleHeight:Integer;nMargin:integer; bMacroPDFEnabled: Boolean;
      pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
      HBITMAP;
       stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeDigit2Bitmap(pBuf: PChar; nRows : Integer; nColumns:Integer;
       nECLevel:Integer; nModuleWidth : Integer; nModuleHeight:Integer;nMargin:integer;
       bMacroPDFEnabled: Boolean;
      pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
      HBITMAP;stdcall;external
       'PDF417ENCODEDLL.DLL';

function    PDF417EncodeHybrid2Bitmap(pBuf: PChar;nLen : Integer; nRows : Integer;
       nColumns:Integer; nECLevel:Integer; nModuleWidth : Integer;
       nModuleHeight:Integer;nMargin:integer; bMacroPDFEnabled: Boolean;
      pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
       HBITMAP;stdcall;external 'PDF417ENCODEDLL.DLL';

function    PDF417EncodeText2Bitmap(pBuf: PChar; nRows : Integer; nColumns:Integer;
       nECLevel:Integer; nModuleWidth : Integer; nModuleHeight:Integer;nMargin:integer;
```

```
  bMacroPDFEnabled: Boolean;
pFileID:PChar;nSegmentIndex:Integer;nSegmentCount;clFore : TColor; clBack : TColor ):
HBITMAP;stdcall;external
  'PDF417ENCODEDLL.DLL';
```

2.2.4.2. Example

Example1

```
var
ctx : TPDF417CONTEXT;
s : string;
i : Integer;
pCtx : LPPDF417CONTEXT;
begin
    pCtx := @ctx;
    InitWorkSpace(pCtx);
    ctx.nRows := 10;
    ctx.nColumns := 12;
    ctx.nModuleWidth := 7;
    ctx.nModuleHeight := 2;
    ctx.nMargin := 20;
    ctx.nErrorCorrectLevel := 1;
    ctx.clForeGround := RGB(255,0,0);
    ctx.clBackGround := RGB(255,255,255);
    Strcopy(ctx.cData,PChar(‘edMemo.Text’));
    ctx.nSize := 11;
    PDF417Encode2File(pCtx,PChar(‘c:\test.bmp’));
    FreeWorkSpace();
end;
```

Example2

```
 if PDF417EncodeHybrid2File(PChar(‘1234567890’),10 , PChar(‘c:\test.bmp’),
      10, 12, 3,2, 7,20, RGB(255,0,0), RGB(255,255,255)) then
begin
    ShowMessage(‘Encode success’);
end;
```

Example3

```
 hBarcode : HBITMAP;
 hBarcode:= PDF417EncodeText2Bitmap(PChar(‘1234567890’),10 ,
                      10, 12, 3,2, 7,20,FALSE,nil,0,0,
   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 PDF417EncodeText2File Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal ImgFile As String, ByVal nRows As Long,
              ByVal nColumns As Long, ByVal nECLevel As Long, ByVal
              nModuleWidth As Long, ByVal nModuleHeight As Long, ByVal nMargin
              As Long, ByVal    bMacroPDFEnabled as bool, ByVal pFileID as long,
ByVal nSegmentIndex as long,ByVal nSegmentCount as long, ByVal clFore As Long,
ByVal clBack As Long) As Boolean

Private Declare Function PDF417EncodeDigit2File Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal ImgFile As String, ByVal nRows As Long,
              ByVal nColumns As Long, ByVal nECLevel As Long, ByVal
              nModuleWidth As Long, ByVal nModuleHeight As Long, ByVal nMargin
              As Long, ByVal    bMacroPDFEnabled as bool, ByVal pFileID as long,
ByVal nSegmentIndex as long,ByVal nSegmentCount as long,ByVal clFore As Long,
ByVal clBack As Long) As Boolean

Private Declare Function PDF417EncodeBinary2File Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal nLen As Long, ByVal ImgFile As String,
              ByVal nRows As Long, ByVal nColumns As Long, ByVal nECLevel As
              Long, ByVal nModuleWidth As Long, ByVal nModuleHeight As Long,
              ByVal nMargin As Long, ByVal    bMacroPDFEnabled as bool, ByVal
pFileID as long, ByVal nSegmentIndex as long,ByVal nSegmentCount as long,ByVal
clFore As Long, ByVal clBack As Long)
              As Boolean

Private Declare Function PDF417EncodeHybrid2File Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal nLen As Long, ByVal ImgFile As String,
              ByVal nRows As Long, ByVal nColumns As Long, ByVal nECLevel As
              Long, ByVal nModuleWidth As Long, ByVal nModuleHeight As Long,
              ByVal nMargin As Long, ByVal    bMacroPDFEnabled as bool, ByVal
          pFileID as long, ByVal nSegmentIndex as long,ByVal nSegmentCount as
          long,ByVal clFore As Long, ByVal clBack As Long)
              As Boolean
```

```
Private Declare Function PDF417EncodeHybrid2Bitmap Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal nLen As Long, ByVal nRows As Long,
             ByVal nColumns As Long, ByVal nECLevel As Long, ByVal
             nModuleWidth As Long, ByVal nModuleHeight As Long, ByVal nMargin
             As Long, ByVal    bMacroPDFEnabled as bool, ByVal pFileID as long,
          ByVal nSegmentIndex as long,ByVal nSegmentCount as long,ByVal clFore
          As Long, ByVal clBack As Long) As Long

Private Declare Function PDF417EncodeBinary2Bitmap Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal nLen As Long, ByVal nRows As Long,
             ByVal nColumns As Long, ByVal nECLevel As Long, ByVal
             nModuleWidth As Long, ByVal nModuleHeight As Long, ByVal nMargin
             As Long, ByVal    bMacroPDFEnabled as bool, ByVal pFileID as long,
          ByVal nSegmentIndex as long,ByVal nSegmentCount as long,ByVal clFore
          As Long, ByVal clBack As Long) As Long

Private Declare Function PDF417EncodeDigit2Bitmap Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal nRows As Long, ByVal nColumns As Long,
             ByVal nECLevel As Long, ByVal nModuleWidth As Long, ByVal
             nModuleHeight As Long, ByVal nMargin As Long, ByVal
bMacroPDFEnabled as bool, ByVal pFileID as long, ByVal nSegmentIndex as
long,ByVal nSegmentCount as long,ByVal clFore As Long,
             ByVal clBack As Long) As Long

Private Declare Function PDF417EncodeText2Bitmap Lib "PDF417Encodedll.dll"
            (ByVal pBuf As String, ByVal nRows As Long, ByVal nColumns As Long,
             ByVal nECLevel As Long, ByVal nModuleWidth As Long, ByVal
             nModuleHeight As Long, ByVal nMargin As Long, ByVal
bMacroPDFEnabled as bool, ByVal pFileID as long, ByVal nSegmentIndex as
long,ByVal nSegmentCount as long,ByVal clFore As Long,
             ByVal clBack As Long) As Long
```

2.2.5.2. Example

Example1

```
…..
   If PDF417EncodeText2File("sadfdsa", "c:\tt.bmp", 10, 12, 1, 2, 7, 10,FALSE,0,0,0,
                       65536, &HFFFFFFFF) Then
         MsgBox("SUCCESS")
   End if
…….
```

Example2

```
……
   Dim hBarcode as long
```

```
hBarcode = PDF417EncodeText2Bitmap("sadfdsa", 10, 12, 1, 2, 7, 10,
                        FALSE,0,0,0,65536, &HFFFFFFFF) Then
……..
```

2.3. ActiveX

2.3.1. Properties

2.3.1.1. Rows

The property set the Rows of PDF417 barcode

short Rows

2.3.1.2. Columns

The property set the Columns of PDF417 barcode

short Columns

2.3.1.3. ModuleWidth

The property set the module width of PDF417 barcode

short ModuleWidth

2.3.1.4. ModuleHeight

The property set the module height of PDF417 barcode

short ModuleHeight

2.3.1.5. CorrectLevel

The property set the error correction level of PDF417 barcode

short CorrectLevel

2.3.1.6. Margin

The property set the margin of PDF417 barcode

short Margin

2.3.1.7. MacroPDFEnabled

The property set the switch for macro PDF417 barcode

bool MacroPDFEnabled

2.3.1.8. FileID

The property set the file id for macro PDF417 barcode

BSTR FileID

2.3.1.8. SegmentIndex

The property set the Segment Index for macro PDF417 barcode

int SegmentIndex

2.3.1.9. SegmentCount

The property set the Total segments for macro PDF417 barcode

int SegmentCount

2.3.1.10. ForeGroundColor

The property set the Foreground color of PDF417 barcode

OLE_COLOR ForeGroundColor

2.3.1.11. BackGroundColor

The property set the Background color of PDF417 barcode

OLE_COLOR BackGroundColor

2.3.1.12. 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.

2.3.2.2. EncodeSmallFile

The method EncodeSmallFile encode the data stored in file and save the barcode image to file.

boolean EncodeSmallFile(BSTR lpDataFile, BSTR lpImageFile);

Parameters

lpDataFile

[in] specify the file name stores data to be encoded

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 PDF417EncodeOcx.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 PDF417EncodeOcx.Ocx

```
#include “PDF417EncodeOcx.h”

    CPDF417EncodeOcx    objPDF417;
    objPDF417.TextData = "http://www.aipsys.com";
    objPDF417.CorrectLevel = 1;
    objPDF417.Rows = 1;
    objPDF417.Columns = 1;
    objPDF417.nModuleWidth = 2;
    objPDF417.nModuleHeight = 7;
    objPDF417.ForeGroundColor = RGB(255,0,0);
    objPDF417.Margin = 10;
    objPDF417.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 PDF417EncodeOcx ActiveX module when dialog shows,
Install it. You can find the component in the Active Page

```
Uses
    …    PDF417EncodeOcx_TLB;
objPDF417 :    TPDF417EncodeOcx;
begin
    objPDF417.TextData := ‘http://www.aipsys.com’
    objPDF417.CorrectLevel    := 1
    objPDF417.Rows    := 1
    objPDF417.Columns    := 1
    objPDF417.nModuleWidth    := 2
    objPDF417.nModuleHeight    := 7
    objPDF417.ForeGroundColor    := &HFF00FF
    objPDF417.Margin    := 10
```

```
        objPDF417.Encode2ImageFile('c:\pdf.gif');
    end;
```

2.3.6. Example for Microsoft visual Basic

```
    Private Sub Command1_Click()
        PDF417EncodeOCX1.TextData = "http://www.aipsys.com"
        PDF417EncodeOCX1.CorrectLevel = 1
        PDF417EncodeOCX1.Rows = 1
        PDF417EncodeOCX1.Columns = 1
        PDF417EncodeOCX1.nModuleWidth = 2
        PDF417EncodeOCX1.nModuleHeight = 7
        PDF417EncodeOCX1.ForeGroundColor = &HFF00FF
        PDF417EncodeOCX1.Margin = 10
        PDF417EncodeOCX1.Encode2ImageFile("c:\pdf.gif");
    End Sub
```

2.4. ASP Control for server side

2.4.1. Properties

2.4.1.1. nRows

The property set the Rows of PDF417 barcode

short nRows

2.4.12 . nColumns

The property set the Columns of PDF417 barcode

short nColumns

2.4.13 . nModuleWidth

The property set the module width of PDF417 barcode

short nModuleWidth

2.4.14 . nModuleHeight

The property set the module height of PDF417 barcode

short nModuleHeight

2.4.15 . nCorrectLevel

The property set the error correction level of PDF417 barcode

short nCorrectLevel

2.4.16 . nMargin

The property set the margin of PDF417 barcode

short nMargin

2.4.17 . MacroPDFEnabled

The property set the switch for macro PDF417 barcode

bool MacroPDFEnabled

2.4.18 . FileID

The property set the file id for macro PDF417 barcode

BSTR FileID

2.4.18 . SegmentIndex

The property set the Segment Index for macro PDF417 barcode

int SegmentIndex

2.4.19 . SegmentCount

The property set the Total segments for macro PDF417 barcode

int SegmentCount

2.4.110 . clForeGround

The property set the Foreground color of PDF417 barcode

OLE_COLOR clForeGround

2.4.111 . clBackGround

The property set the Background color of PDF417 barcode

OLE_COLOR clBackGround

2.4.112 . 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 PDF417EncodeASP.DLL

2.4.4. Example for ASP

```
<%
    set obj = Server.CreateObject("PDF417EncodeCOM.EncodeService")
    obj.InitWorkspace()
    obj.nColumns = 10           '  Column
    obj.nRows = 12            '  Rows
    obj.nMargin = 5            '  Margin
    obj.nModuleWidth = 2           '  Margin
    obj.nModuleHeight = 7           '  Margin
    obj.nErrorCorrectLevel= 0   '  Correction Level
    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>")

%>
```

3. 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

4. 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 share*it! 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

5. 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

6. Product Information Link

. QRCode encoder SDK

. PDF417 encoder SDK

. DataMatrix encoder SDK

. Aztec encoder SDK

. Linear 1D barcode encoder SDK