Aztec Encode&Decode SDK v2.5

USER MANUAL

AIPSYS Software Laboratory

http://www.aipsys.com

Last Updated 21st March 2013

1. Introduction

1.1. Aztec Barcode introduction

About Aztec

Aztec is a high density 2D matrix bar code, created in Welch Allyn Inc., Andy Longacre in year 1995, public domain. Aztec code is a 2D matrix barcode symbology designed to combine the best characteristics of several 1st generation symbologies, with special attention paid to ease of printing, ease of finding in any orientation, allowance for field distortion, high data security with user-selected redundancy, and efficient storage over the range from small to large data messages. Aztec symbols are always a square array of square cells with a square "bullseye" at the center. Figure 1 shows an Aztec code symbol encoding 109 characters of text with 50% error correction.

Reed-Solomon error control encoding provides a user-selected level of data security and robustness.

Features

- Easy printing
- Easy decoding - bulls eye in center
- minimum size is 13 chars - equal to EAN13 to size
- No need quiet zone
- 32 code sizes
- Reed Solomon error coding
- can encode 3750 character from all 256 ASCII chars
- center of barcode is "bulls eye"
- smallest element is named "element, square dot
- error correction symbols size are selectable

The overall size of an Aztec symbol is depended on module size, the total amount of encoded data and error capacity is choosen by the user.

The smallest Aztec symbol is 15 module square and ancode 14 digits with 40 % error correction.

The largest symbol is 151 modules square and can encode 3000 chracters or 3750 digits with 25% error correction.

Encoding specification

- **Encodable character set:**
  - All 8-bit values
    - 0-127 ASCII
    - 128-255 ISO 8859-1, Latin Alphabet No.1
    - Two non-data characters can be encoded, FNC1 for compatibility and ECI escape sequence for standardized encoding....
- **Representations of data :** dark module is binary one and light module is binary zero....

- **Symbol size:** smallest 15x15 modules, largest 151x151 modules
- **Data capacity:**
  - Smallest Aztec code encodes 13 numeric or 12 alphanumeric or bytes of data
  - Largest encodes 3832 numerics, 3067 alphabetic or 1914 bytes.
- **Error correction:**
  - User-selectable, from 5 % to 95% of data
  - Recomended level is 23% of symbol capacity plus codewords
- **Codetype :** Matrix
- **Orientation independent :** Yes

Data codeword sizes and prime modulus polynomials.				
Layers	**Codeword**	**Galois field**	**Prime Modulus Polynomial**	**decimal equib.**
1-2	6 bits	64	X^6+X+1	67
3-8	8 bits	256	$X^8+X^5+X^3+X^2+1$	301
9-22	10 bits	1024	$X^{10}+X^3+1$	1033
23-32	12 bits	4096	$X^{12}+X^6+X^5+X^3+1$	4021

Aztec code high-level encoding										
	Upper		**Lower**		**Mixed**		**Punct**		**Digit**	
Value	**Char**	**ascii**	**Char**	**ascii**	**Char**	**ascii**	**Char**	**ascii**	**Char**	**ascii**
0	PS		PS		PS		FLG	***	PS	
1	SP	32	SP	32	SP	32	CR	13	SP	32
2	A	65	a	97	SOH	1	CR LF	13,10	0	48
3	B	66	b	98	STX	2	. SP	46,32	1	49
4	C	67	c	99	ETX	3	, SP	44,32	2	50
5	D	68	d	100	EOT	4	: SP	58,32	3	51
6	E	69	e	101	ENQ	5	!	33	4	52
7	F	70	f	102	ACK	6	"	34	5	53
8	G	71	g	103	BEL	7	#	35	6	54
9	H	72	h	104	BS	8	$	36	7	55
10	I	73	i	105	HT	9	%	37	8	56
11	J	74	j	106	LF	10	&	38	9	57
12	K	75	k	107	VT	11	'	39	,	44
13	L	76	l	108	FF	12	(	40	.	46
14	M	77	m	109	CR	13	)	41	UL	
15	N	78	n	110	ESC	27	*	42	US	
16	O	79	o	111	FS	28	+	43		

17	P	80	p	112	GS	29	,	44		
18	Q	81	q	113	RS	30	-	45		
19	R	82	r	114	US	31	.	46		
20	S	83	s	115	@	64	/	47		
21	T	84	t	116	\	92	:	58		
22	U	85	u	117	^	94	;	99		
23	V	86	v	118		95	<	60		
24	W	87	w	119	`	96	=	61		
25	X	88	x	120	|	124	>	62		
26	Y	89	y	121	~	126	?	63		
27	Z	90	z	122	DEL	127	[	91		
28	LL		US		LL		]	93		
29	ML		ML		UL		{	123		
30	DL		DL		PL		}	125		
31	BS		BS		BS		UL			

SMALL AZTEC CODE.

Small Aztec code is smaller version of Aztec code for coding smaller messages to 95 chars.Decoding rights is some as in Aztec Code

- Only 4 sizes of barcode
- Easy printing
- Easy decoding
- Maximum 95 chars or 120 digits

1.2 Aztec parameters introduction

nEncodeMode: There are two modes for encoding:

ENC_NORMAL: can encode any character but it is not very efficient encoding binary values (values above 128).

ENC_BINARY: use this mode only if your data contains many bytes/characters above 128

nCorrectionLevel: percentage of errors which can be recovered: 5 to 95

nConfigType: use this property to define which formats can be used: ANY (any), COMPACT (only compact formats) or FULL (only full range formats).

ANY:

COMPACT:

FULL RANGE:.

nRune: set a value between 0 and 255 to create a Aztec code rune (default is -1, disabled)

nConfiguration: preferred format. Another format will be automatically selected if AutoConfigurate=true and the amount of data and the selected error correction level does not fit in the preferred format:

CONFIGURATION_AUTO = -1;
CONFIGURATION_15X15_COMPACT = 0;
CONFIGURATION_19X19 = 1;
CONFIGURATION_19X19_COMPACT = 2;
CONFIGURATION_23X23 = 3;
CONFIGURATION_23X23_COMPACT = 4;
CONFIGURATION_27X27 = 5;
CONFIGURATION_27X27_COMPACT = 6;
CONFIGURATION_31X31 = 7;
CONFIGURATION_37X37 = 8;
CONFIGURATION_41X41 = 9;
CONFIGURATION_45X45 = 10;
CONFIGURATION_49X49 = 11;
CONFIGURATION_53X53 = 12;
CONFIGURATION_57X57 = 13;
CONFIGURATION_61X61 = 14;
CONFIGURATION_67X67 = 15;
CONFIGURATION_71X71 = 16;
CONFIGURATION_75X75 = 17;
CONFIGURATION_79X79 = 18;
CONFIGURATION_83X83 = 19;
CONFIGURATION_87X87 = 20;
CONFIGURATION_91X91 = 21;
CONFIGURATION_95X95 = 22;
CONFIGURATION_101X101 = 23;
CONFIGURATION_105X105 = 24;
CONFIGURATION_109X109 = 25;
CONFIGURATION_113X113 = 26;
CONFIGURATION_117X117 = 27;
CONFIGURATION_121X121 = 28;
CONFIGURATION_125X125 = 29;
CONFIGURATION_131X131 = 30;
CONFIGURATION_135X135 = 31;
CONFIGURATION_139X139 = 32;
CONFIGURATION_143X143 = 33;
CONFIGURATION_147X147 = 34;
CONFIGURATION_151X151 = 35;

bStructuredAppend : allows you to encode large messages by means of a sequence of symbols

bProcessTilde: if true (default) the tilde character (~) will be processed like this:

- ~~: will be replaced with ~
- ~dxxx: will be replaced by the character whose ascii code is xxx. For example ~d065 will be replaced with A.
- ~F: will be replaced with the FNC1 flag (allowed as first codeword only).
- ~Exxxxxx: will be replaced with the Extended Interpretation Channel flag xxxxxx. For example to activate Extended Interpretation Channel 1, use ~E000001

nStructuredAppendCounter(int): When the symbol is part of an Structured Append set, this is the total number of symbols in the set. Valid values : 2 to 26. If this is not set, or set to 1, the symbol is assumed to be standalone (no structured append used)..

nStructuredAppendIndex(int): When the symbol is part of an Structured Append set, this is the number of this symbol in the set. Valid values : 2 to 26.

pFileID: File Id for Structured Append. Optional

1.3 Data capacity for Aztec versions

The size and data bits capacity of Aztec code symbols								
Layers	**Size**	**Capacity**	**Layers**	**Size**	**Capacity**	**Layers**	**Size**	**Capacity**
1	19x19	21x6	12	67x67	364x10	23	113x113	920x12
2	23x23	48x6	13	71x71	416x10	24	117x117	992x12
3	27x27	60x8	14	75x75	470x10	25	121x121	1066x12
4	31x31	88x8	15	79x79	528x10	26	125x125	1144x12
5	37x37	120x8	16	83x83	588x10	27	131x131	1224x12
6	41x41	156x8	17	87x87	652x10	28	135x135	1306x12
7	45x45	196x8	18	91x91	720x10	29	139x139	1392x12
8	49x49	240x8	19	95x95	790x10	30	143x143	1480x12
9	53x53	230x10	20	101x101	864x10	31	147x147	1570x12
10	57x57	272x10	21	105x105	940x10	32	15x151	1664x12
11	61x61	316x10	22	109x109	1020x10			

The Size and Bit Capacity of Small Aztec Symbols					
Layers	**Size**	**Capacity**	**Layers**	**Size**	**Capacity**
1	15x15	17x6	3	23x23	51x6
2	19x19	40x6	4	27x27	76x8

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.

1.5. 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[Win32/Win64/WindowsMobile/CE]

2.1.1 Constants

Encoding scheme

ENC_NORMAL	0;
ENC_BINARY	1;

ConfigType

CONFIGURATION_ANY	0;
CONFIGURATION_COMPACT	1;
CONFIGURATION_FULL	2;

Correction Level

0~35%

Configuration

#define CONFIGURATION_AUTO	-1
#define CONFIGURATION_15X15_COMPACT	0
#define CONFIGURATION_19X19 =	1
#define CONFIGURATION_19X19_COMPACT	2
#define CONFIGURATION_23X23	3
#define CONFIGURATION_23X23_COMPACT	4
#define CONFIGURATION_27X27	5
#define CONFIGURATION_27X27_COMPACT	6
#define CONFIGURATION_31X31	7
#define CONFIGURATION_37X37	8
#define CONFIGURATION_41X41	9
#define CONFIGURATION_45X45	10
#define CONFIGURATION_49X49	11
#define CONFIGURATION_53X53	12


```
#define CONFIGURATION_57X57          13
#define CONFIGURATION_61X61          14
#define CONFIGURATION_67X67          15
#define CONFIGURATION_71X71          16
#define CONFIGURATION_75X75          17
#define CONFIGURATION_79X79          18
#define CONFIGURATION_83X83          19
#define CONFIGURATION_87X87          20
#define CONFIGURATION_91X91          21
#define CONFIGURATION_95X95          22
#define CONFIGURATION_101X101        23
#define CONFIGURATION_105X105        24
#define CONFIGURATION_109X109        25
#define CONFIGURATION_113X113        26
#define CONFIGURATION_117X117        27
#define CONFIGURATION_121X121        28
#define CONFIGURATION_125X125        29
#define CONFIGURATION_131X131        30
#define CONFIGURATION_135X135        31
#define CONFIGURATION_139X139        32
#define CONFIGURATION_143X143        33
#define CONFIGURATION_147X147        34
#define CONFIGURATION_151X151        35
```

2.1.2 Data structure

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

```
typedef struct tagAZTECCONTEXT
{
    int       nEncodeMode;
    int       nCorrectionLevel;
    int       nConfigType;
    int       nConfiguration;
    int       nRune;
    int       nPixelSize;
    int       nMargin;
    int       nStructuredAppendCounter;
    int       nStructuredAppendIndex;
    int       nDataSize;
    BOOL      bProcessTilde;
    BOOL      bStructuredAppend;
    COLORREF clForeColor;
```



```
    COLORREF clBackColor;
    char *pFileID;
    char cEncodedData[3750];
}_AZTECCONTEXT;
```

2.1.3. Function or procedure

2.1.3.1. _InitAztecContext

The _InitAztecContext function initilize the environment of Aztec encoder with default value.

void __stdcall _*InitAztecContext* (_AZTECCONTEXT * pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type _AZTECCONTEXT

Return values

None

2.1.3.2. _AztecEncode2File

The _AztecEncode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall _AztecEncode2File (_AZTECCONTEXT * pAztecCtx,
LPCTSTR lpImageFile);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type _ AZTECCONTEXT

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. _AztecEncode2Bitmap

The **_AztecEncode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the Aztec barcode image

```
HBITMAP __stdcall _AztecEncode2Bitmap (
                                _AZTECCONTEXT * pAztecCtx);
```

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type _AZTECCONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of Aztec barcode, otherwise , return NULL.

2.1.3.4. _FreeAztecContext

The _FreeAztecContext function free environment of the Aztec encoder

```
BOOL __stdcall _FreeAztecContext ();
```

Return values

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

2.1.4. Example for Microsoft visual C++

Example1

```
#include "AztecEncodeLib.h"
....
    _tagAZTECCONTEXT tAztecCtx;
    _InitAztecContext(&tAztecCtx);

    tAztecCtx.nConfigType = CONFIGURATION_ANY;
    tAztecCtx.nCorrectionLevel = 23;
    tAztecCtx.nConfiguration = CONFIGURATION_AUTO;
    tAztecCtx.nEncodeMode = m_EncodeMode.GetCurSel();
    tAztecCtx.nStructuredAppendIndex = m_nAppendIndex;
    tAztecCtx.nStructuredAppendCounter = m_nAppendCounter;
    tAztecCtx.bStructuredAppend = false;
```



```
    tAztecCtx.bProcessTilde = false;
    tAztecCtx.nMargin = 10;
    tAztecCtx.nPixelSize = 4;
    tAztecCtx.clBackColor = 0xFFFFFFFF;
    tAztecCtx.clForeColor = 0

memcpy(tAztecCtx.cEncodedData,http://www.aipsys.com”,23);
    _AztecEncode2File(&tQrCtx, "c:\\Aztec.bmp");
    _FreeAztecContext();
. . . .
```

LIBRARY for linking

AztecEncodeLIB.LIB

2.2. Dynamic link library[Win32/Win64/WindowsMobile/CE]

2.2.1 Constants

Encoding scheme

```
ENC_NORMAL      0;
ENC_BINARY      1;
```

ConfigType

```
CONFIGURATION_ANY           0;
CONFIGURATION_COMPACT       1;
CONFIGURATION_FULL          2;
```

Correction Level

0~35%

Configuration

```
#define CONFIGURATION_AUTO              -1
#define CONFIGURATION_15X15_COMPACT     0
#define CONFIGURATION_19X19 =           1
#define CONFIGURATION_19X19_COMPACT     2
#define CONFIGURATION_23X23             3
#define CONFIGURATION_23X23_COMPACT     4
#define CONFIGURATION_27X27             5
#define CONFIGURATION_27X27_COMPACT     6
#define CONFIGURATION_31X31             7
```



```
#define CONFIGURATION_37X37          8
#define CONFIGURATION_41X41          9
#define CONFIGURATION_45X45          10
#define CONFIGURATION_49X49          11
#define CONFIGURATION_53X53          12
#define CONFIGURATION_57X57          13
#define CONFIGURATION_61X61          14
#define CONFIGURATION_67X67          15
#define CONFIGURATION_71X71          16
#define CONFIGURATION_75X75          17
#define CONFIGURATION_79X79          18
#define CONFIGURATION_83X83          19
#define CONFIGURATION_87X87          20
#define CONFIGURATION_91X91          21
#define CONFIGURATION_95X95          22
#define CONFIGURATION_101X101        23
#define CONFIGURATION_105X105        24
#define CONFIGURATION_109X109        25
#define CONFIGURATION_113X113        26
#define CONFIGURATION_117X117        27
#define CONFIGURATION_121X121        28
#define CONFIGURATION_125X125        29
#define CONFIGURATION_131X131        30
#define CONFIGURATION_135X135        31
#define CONFIGURATION_139X139        32
#define CONFIGURATION_143X143        33
#define CONFIGURATION_147X147        34
#define CONFIGURATION_151X151        35
```

2.2.2. Data structure

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

```
typedef struct tagAZTECCONTEXT
{
    int    nEncodeMode;
     int    nCorrectionLevel;
    int     nConfigType;
    int     nConfiguration;
    int    nRune;
    int    nPixelSize;
```



```
    int    nMargin;
    int    nStructuredAppendCounter;
    int    nStructuredAppendIndex;
    int    nDataSize;
    BOOL bProcessTilde;
    BOOL bStructuredAppend;
    COLORREF clForeColor;
    COLORREF clBackColor;
    char *pFileID;
    char cEncodedData[3750];
}AZTECCONTEXT;
```

2.2.2. Function or procedure

2.2.2.1. InitWorkSpace

The _InitWorkSpace function initilize the environment of Aztec encoder with default value.

void __stdcall _InitWorkSpace(AZTECCONTEXT *pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT

Return values

None

2.2.2.2. AztecEncode2File

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

BOOL __stdcall AztecEncode2File (AZTECCONTEXT *pAztecCtx,
LPCTSTR lpImageFile);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT

lpImageFile

[in] define the image file output, currently bitmap image supported

Return values

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

2.2.2.3. AztecEncode2Bitmap

The **AztecEncode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the Aztec barcode image

HBITMAP __stdcall AztecEncode2Bitmap (
AZTECCONTEXT *pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT

Return values

If the function succeeds, the return value is BITMAP handle of AZTEC barcode, otherwise , return NULL.

2.2.2.4. FreeWorkSpace

The FreeWorkSpace function free environment of the Aztec encoder

BOOL __stdcall FreeWorkSpace();

Return values

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

2.2.3. Example for Microsoft visual C++

Example1

```
#include "AztecEncodeDLL.h"
.....
    tagAZTECCONTEXT tAztecCtx;
```



```
        InitWorkSpace(&tAztecCtx);

        tAztecCtx.nConfigType = CONFIGURATION_ANY;
        tAztecCtx.nCorrectionLevel = 23;
        tAztecCtx.nConfiguration = CONFIGURATION_AUTO;
        tAztecCtx.nEncodeMode = m_EncodeMode.GetCurSel();
        tAztecCtx.nStructuredAppendIndex = m_nAppendIndex;
        tAztecCtx.nStructuredAppendCounter = m_nAppendCounter;
        tAztecCtx.bStructuredAppend = false;
        tAztecCtx.bProcessTilde = false;
        tAztecCtx.nMargin = 10;
        tAztecCtx.nPixelSize = 4;
        tAztecCtx.clBackColor = 0xFFFFFFFF;
        tAztecCtx.clForeColor = 0

        memcpy(tAztecCtx.cEncodedData,http://www.aipsys.com”,23);
        AztecEncode2File(&tQrCtx, "c:\\Aztec.bmp");
        FreeWorkSpace();
    ....
```

LIBRARY for linking
AztecEncodeDLL.lib
RUNTIME LIBRARY
AztecEncodeDLL.DLL

2.2.4. Example for Borland Delphi

2.2.4.1. Redeclaration of the data type and function

```
LPAZTECCONTEXT = ^TAztecContext;

TAztecContext = record
                    encodeMode : Integer;
                    correctionLevel : Integer;
                    configType : Integer;
                    rune : Integer;
                    pixelSize : Integer;
                    margin : Integer;
                    structuredAppendCounter : Integer;
                    structuredAppendIndex : Integer;
                    dataSize : Integer;
```



```
            processTilde : boolean;
            structuredAppend : boolean;
            foreColor : TColor;
            backColor : TColor;
            fileID : PChar;
            encodedData : array[1..3750] of char;

    end;

procedure InitWorkSpace(pAztecCtx : LPAZTECCONTEXT ); stdcall; external
'AZTECENCODEDLL.DLL';

function    AztecEncode2File(pAztecCtx :
        LPAZTECCONTEXT;lpImageFile :PChar) : boolean; stdcall;external
        'AZTECENCODEDLL.DLL';

function    AztecEncode2Bitmap(pAztecCtx : LPAZTECCONTEXT) :
        HBITMAP;stdcall;external 'AZTECENCODEDLL.DLL';

function    FreeWorkSpace : boolean;stdcall external
        'AZTECENCODEDLL.DLL';
```

2.2.4.2. Example

Example1

```
var
  ctx : TAZTECCONTEXT;
  s : string;
  i : Integer;
  bmp : HBITMAP;
  img : TBitmap;
  pCtx : LPAZTECCONTEXT;
begin
    pCtx := @tAztecCtx;
    InitWorkSpace(pCtx);

    tAztecCtx.configType := ComboBox3.ItemIndex;
    tAztecCtx.correctionLevel := ComboBox2.ItemIndex+1;
    tAztecCtx.encodeMode := ComboBox1.ItemIndex;

    tAztecCtx.structuredAppendIndex := StrToInt(Edit2.Text);
    tAztecCtx.structuredAppendCounter := StrToInt(Edit4.Text);
```



```
    tAztecCtx.structuredAppend := CheckBox2.Checked;
    tAztecCtx.processTilde := CheckBox2.Checked;
    tAztecCtx.dataSize := length(Edit6.Text);
    tAztecCtx.margin := StrToInt(Edit5.Text);
    tAztecCtx.pixelSize := StrToInt(Edit3.Text);
    if (Edit6.Text <> '') then

        StrMove(@tAztecCtx.encodedData,PChar(Edit6.Text),length(Edit6.Text
        ));
    AztecEncode2File(pCtx,pChar('c:\1.bmp'));

    FreeWorkSpace();
end;
```

2.3. ActiveX[Win32/Win64]

2.3.1. Properties

2.3.1.1. PreferredConfig

The property set the configuration of Aztec 0~35, autoconfigure the parameter when value is –1;

short PreferredConfig

2.3.1.2. CorrectionLevel

The property set the error correction level 0~35%

short **CorrectionLevel**

2.3.1.3. EncodeMode

The property set the encoding scheme of Aztec barcode

```
ENC_NORMAL = 0;
ENC_BINARY  = 1;
```

short **EncodeMode**

2.3.1.4. StructuredAppendCounter

The property set the StructuredAppendCounter of Aztec barcode

short StructuredAppendCounter

2.3.1.5. ConfigType

The property set the config type of Aztec barcode

short ConfigType

2.3.1.6. StructuredAppend

The property set if Aztec is structured append

bool StructuredAppend

2.3.1.7. ProcessTilde

The property set if Aztec is process tilde

bool ProcessTilde

2.3.1.8. Rune

The property set rune of Aztec

bool Rune

2.3.1.9. Margin

The property set barcode Margin

bool Margin

2.3.1.10. PixelSize

The property set module Size of barcode image

bool PixelSize

2.3.1.11. ForeColor

The property set the Foreground color of Aztec barcode

OLE_COLOR ForeColor

2.3.1.12. BackColor

The property set the Background color of Aztec barcode

OLE_COLOR BackGroundColor

2.3.1.13. EncodedData

The property set the data to be encoded

BSTR EncodedData

2.3.1.14. FileID

The property set the file ID of Aztec barcode

BSTR FileID

2.3.1.15. StructuredAppendIndex

The property set the StructuredAppendIndexof Aztec barcode

short StructuredAppendIndex

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 AztecEncodeOcx.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 AztecEncodeOcx.Ocx

```
#include “AztecEncodeOcx.h”

    CAztecEncodeOcx    objAztec;
    objAztec.EncodedData = "http://www.aipsys.com";
    objAztec. Configuration = -1;
    objAztec.EncodMode = 0;
    objAztec.EncodType = 0;
    objAztec.CorrectinLevel = 23;
     objAztec.PixelSize = 4;
    objAztec.ForeGroundColor = RGB(255,0,0);
    objAztec.Margin = 10;
    objAztec.Encode2ImageFile("c:\\aztec.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
    …   AztecEncodeOcx_TLB;
objAztec:   TAztecEncodeOcx;
begin
    objAztec.EncodedData := 'http://www.aipsys.com';
    objAztec.PixelSize   := 4;
    objAztec.EncodeMode   := 0;
    objAztec. Configuration := -1;
    objAztec. ConfigType := 0;
    objAztec. CorrectionLevel := 23;
    objAztec.ForeColor   := &HFF00FF
    objAztec.Margin   := 10
    objAztec.Encode2ImageFile('c:\qr.gif');
end;
```

2.4. ASP Control for server side[Win32/Win64]

2.4.1. Properties

2.4.1.1. PreferredConfig

The property set the PreferredConfig of Aztec 0~35, auto configurate when value is –1.

short PreferredConfig

2.4.1.2. PixelSize

The property set the module width of Aztec barcode

short PixelSize

2.4.1.5. EncodeMode

The property set the encoding mode of Aztec barcode

short EncodeMode

2.4.1.6. Margin

The property set the margin of Aztec barcode

short Margin

2.4.1.7. CorrectionLevel

The property set the error correction level of Aztec barcode

short CorrectionLevel

2.4.1.8. ForeColor

The property set the Foreground color of Aztec barcode

OLE_COLOR ForeColor

2.4.1.9. BackColor

The property set the Background color of Aztec barcode

OLE_COLOR BackColor

2.4.1.9. EncodedData

The property set the data to be encoded

BSTR EncodedData

2.4.1.10. ConfigType

The property set the config type f aztec barcode


```
int  ConfigType
```

2.4.1.11. StructuredAppend

The property set if Aztec is structured append

bool StructuredAppend

2.4.1.12. ProcessTilde

The property set if Aztec is process tilde

bool ProcessTilde

2.4.1.13. FileID

The property set file ID of Aztec

BSTR FileID

2.4.1.14. StructuredAppendCounter

The property set the StructuredAppendCounter of Aztec barcode

short StructuredAppendCounter

2.4.1.14. StructuredAppendIndex

The property set the StructuredAppendIndexof Aztec barcode

short StructuredAppendIndex

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.

2.4.2.3. 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 AztecEncodeASP.DLL

2.4.4. Example for ASP

```
<%

    set obj = Server.CreateObject("AztecEncodeCOM.EncodeService")
    obj.InitWorkspace()
```



```
        obj.encodeMode= 0                   '    Encodeing mode
        obj.CorrectionLevel = 23
        obj..configType = 1
        obj.Rune = -1
        obj.PixelSize = 4                  ' Module size
        obj.PreferredConfig = -1
        obj.Margin = 5                      '    Margin
       obj.StructuredAppendCounter = 0;
       obj.StructuredAppendIndex = 0;
       obj.ProcessTilde = true;
       obj.StructuredAppend = false;
        obj.EncodedData= "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

Encoding scheme

ENC_NORMAL	0;
ENC_BINARY	1;

ConfigType

CONFIGURATION_ANY	0;
CONFIGURATION_COMPACT	1;
CONFIGURATION_FULL	2;

Correction Level

0~35%

Configuration

#define CONFIGURATION_ANY	0
#define CONFIGURATION_COMPACT	1
#define CONFIGURATION_FULL	2
#define CONFIGURATION_AUTO	-1
#define CONFIGURATION_15X15_COMPACT	0


```
#define CONFIGURATION_19X19 =                1
#define CONFIGURATION_19X19_COMPACT          2
#define CONFIGURATION_23X23                  3
#define CONFIGURATION_23X23_COMPACT          4
#define CONFIGURATION_27X27                  5
#define CONFIGURATION_27X27_COMPACT          6
#define CONFIGURATION_31X31                  7
#define CONFIGURATION_37X37                  8
#define CONFIGURATION_41X41                  9
#define CONFIGURATION_45X45                  10
#define CONFIGURATION_49X49                  11
#define CONFIGURATION_53X53                  12
#define CONFIGURATION_57X57                  13
#define CONFIGURATION_61X61                  14
#define CONFIGURATION_67X67                  15
#define CONFIGURATION_71X71                  16
#define CONFIGURATION_75X75                  17
#define CONFIGURATION_79X79                  18
#define CONFIGURATION_83X83                  19
#define CONFIGURATION_87X87                  20
#define CONFIGURATION_91X91                  21
#define CONFIGURATION_95X95                  22
#define CONFIGURATION_101X101                23
#define CONFIGURATION_105X105                24
#define CONFIGURATION_109X109                25
#define CONFIGURATION_113X113                26
#define CONFIGURATION_117X117                27
#define CONFIGURATION_121X121                28
#define CONFIGURATION_125X125                29
#define CONFIGURATION_131X131                30
#define CONFIGURATION_135X135                31
#define CONFIGURATION_139X139                32
#define CONFIGURATION_143X143                33
#define CONFIGURATION_147X147                34
#define CONFIGURATION_151X151                35
```

2.5.2. Data structure

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


```
typedef struct tagAZTECCONTEXT
{
	int   nEncodeMode;
	int   nCorrectionLevel;
	int     nConfigType;
	int     nConfiguration;
	int   nRune;
	int   nPixelSize;
	int   nMargin;
	int   nStructuredAppendCounter;
	int   nStructuredAppendIndex;
	int   nDataSize;
	BOOL bProcessTilde;
	BOOL bStructuredAppend;
	COLORREF clForeColor;
	COLORREF clBackColor;
	char *pFileID;
	char cEncodedData[3750];
}AZTECCONTEXT;
```

2.5.3. Function or procedure

2.5.3.1. InitAztecContext

The _ **InitAztecContext** function initilize the environment of Aztec encoder with default value.

void _InitAztecContext (_AZTECCONTEXT *pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoder, refer structure type AZTECCONTEXT

Return values

None

2.5.3.2. AztecEncode2Bitmap

The AztecEncode2Bitmap 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* _AztecEncode2Bitmap(_AZTECCONTEXT *pAztecCtx,int *pWidth,int *pHeight);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT

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 AZTEC barcode , pWidth \pHeight ;otherwise , return NULL.

2.5.3.3. AztecRegister

The _AztecRegister function help user register the SDK before running. AIPSYS will send user the register information including mail-box string and regcode.

bool _AztecRegister(const char *pMailBox,const 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(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”**)**

```
#include "AztecEncodeLib.h"
.....

/* 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;
    }


    _AZTECCONTEXT tagAztec;
        int width,height;
        unsigned char *pMap;


        _InitAztecContext(&tagAztec);
        tagAztec.nEncodeMode= ENC_NORMAL;
        tagAztec.nConfigType = CONFIGURATION_FULL;
        tagAztec.nConfiguration = -1;
        tagAztec.nCorrectionLevel = 1;
        memcpy(tagAztec.cEncodedData,"http://www.aipsys.com",23);
        tagAztec.nDataSize = 23;
        tagAztec.nMargin = 10;
```



```
tagAztec.nPixelSize = 4;

pMap = _AztecEncode2Bitmap(&tagAztec,&width,&height);

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);

....
```

LIBRARY for linking

libAztecEncodeLibIOS.a

2.6. Library for Linux

2.6.1 Constants

Encoding scheme

ENC_NORMAL	0;
ENC_BINARY	1;

ConfigType

CONFIGURATION_ANY	0;
CONFIGURATION_COMPACT	1;
CONFIGURATION_FULL	2;

Correction Level

0~35%

Configuration

#define CONFIGURATION_ANY	0
#define CONFIGURATION_COMPACT	1
#define CONFIGURATION_FULL	2
#define CONFIGURATION_AUTO	-1
#define CONFIGURATION_15X15_COMPACT	0
#define CONFIGURATION_19X19 =	1
#define CONFIGURATION_19X19_COMPACT	2
#define CONFIGURATION_23X23	3

```
#define CONFIGURATION_23X23_COMPACT     4
#define CONFIGURATION_27X27             5
#define CONFIGURATION_27X27_COMPACT     6
#define CONFIGURATION_31X31             7
#define CONFIGURATION_37X37             8
#define CONFIGURATION_41X41             9
#define CONFIGURATION_45X45             10
#define CONFIGURATION_49X49             11
#define CONFIGURATION_53X53             12
#define CONFIGURATION_57X57             13
#define CONFIGURATION_61X61             14
#define CONFIGURATION_67X67             15
#define CONFIGURATION_71X71             16
#define CONFIGURATION_75X75             17
#define CONFIGURATION_79X79             18
#define CONFIGURATION_83X83             19
#define CONFIGURATION_87X87             20
#define CONFIGURATION_91X91             21
#define CONFIGURATION_95X95             22
#define CONFIGURATION_101X101           23
#define CONFIGURATION_105X105           24
#define CONFIGURATION_109X109           25
#define CONFIGURATION_113X113           26
#define CONFIGURATION_117X117           27
#define CONFIGURATION_121X121           28
#define CONFIGURATION_125X125           29
#define CONFIGURATION_131X131           30
#define CONFIGURATION_135X135           31
#define CONFIGURATION_139X139           32
#define CONFIGURATION_143X143           33
#define CONFIGURATION_147X147           34
#define CONFIGURATION_151X151           35
```

2.6.2. Data structure

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

```
typedef struct tagAZTECCONTEXT
{
    int   nEncodeMode;
    int   nCorrectionLevel;
```



```
    int     nConfigType;
    int     nConfiguration;
    int   nRune;
    int   nPixelSize;
    int   nMargin;
    int   nStructuredAppendCounter;
    int   nStructuredAppendIndex;
    int   nDataSize;
    BOOL bProcessTilde;
    BOOL bStructuredAppend;
    COLORREF clForeColor;
    COLORREF clBackColor;
    char *pFileID;
    char cEncodedData[3750];
}AZTECCONTEXT;
```

2.6.3. Function or procedure

2.6.3.1. InitAztecContext

The _ **InitAztecContext** function initilize the environment of Aztec encoder with default value.

void _InitAztecContext (_AZTECCONTEXT *pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoder, refer structure type AZTECCONTEXT

Return values

None

2.6.3.2. AztecEncode2Bitmap

The AztecEncode2Bitmap 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* _AztecEncode2Bitmap(_AZTECCONTEXT *pAztecCtx,int *pWidth,int *pHeight);

Parameters

pAztecCtx
[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT
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 AZTEC barcode , pWidth \pHeight ;otherwise , return NULL.

2.6.3.3. AztecRegister

The _AztecRegister function help user register the SDK before running. AIPSYS will send user the register information including mail-box string and regcode.

bool _AztecRegister(const char *pMailBox,const 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++

2.6.4.1Example1

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 "AztecEncodeLib.h"
.....
 _AZTECCONTEXT tagAztec;
        int width,height;
        unsigned char *pMap;


        _InitAztecContext(&tagAztec);
        tagAztec.nEncodeMode= ENC_NORMAL;
        tagAztec.nConfigType = CONFIGURATION_FULL;
        tagAztec.nConfiguration = -1;
        tagAztec.nCorrectionLevel = 1;
        memcpy(tagAztec.cEncodedData,"http://www.aipsys.com",23);
        tagAztec.nDataSize = 23;
        tagAztec.nMargin = 10;
        tagAztec.nPixelSize = 4;
        pMap = _AztecEncode2Bitmap(&tagAztec,&width,&height);
        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);
        ....
```


LIBRARY for linking
libAztecEncodeLibLinux.a

2.7. Library for Linux ARM

2.7.1 Constants

Encoding scheme

ENC_NORMAL	0;
ENC_BINARY	1;

ConfigType

CONFIGURATION_ANY	0;
CONFIGURATION_COMPACT	1;
CONFIGURATION_FULL	2;

Correction Level

0~35%

Configuration

#define CONFIGURATION_ANY	0
#define CONFIGURATION_COMPACT	1
#define CONFIGURATION_FULL	2
#define CONFIGURATION_AUTO	-1
#define CONFIGURATION_15X15_COMPACT	0
#define CONFIGURATION_19X19 =	1
#define CONFIGURATION_19X19_COMPACT	2
#define CONFIGURATION_23X23	3
#define CONFIGURATION_23X23_COMPACT	4
#define CONFIGURATION_27X27	5
#define CONFIGURATION_27X27_COMPACT	6
#define CONFIGURATION_31X31	7
#define CONFIGURATION_37X37	8
#define CONFIGURATION_41X41	9
#define CONFIGURATION_45X45	10
#define CONFIGURATION_49X49	11
#define CONFIGURATION_53X53	12
#define CONFIGURATION_57X57	13
#define CONFIGURATION_61X61	14
#define CONFIGURATION_67X67	15
#define CONFIGURATION_71X71	16

```
#define CONFIGURATION_75X75          17
#define CONFIGURATION_79X79          18
#define CONFIGURATION_83X83          19
#define CONFIGURATION_87X87          20
#define CONFIGURATION_91X91          21
#define CONFIGURATION_95X95          22
#define CONFIGURATION_101X101        23
#define CONFIGURATION_105X105        24
#define CONFIGURATION_109X109        25
#define CONFIGURATION_113X113        26
#define CONFIGURATION_117X117        27
#define CONFIGURATION_121X121        28
#define CONFIGURATION_125X125        29
#define CONFIGURATION_131X131        30
#define CONFIGURATION_135X135        31
#define CONFIGURATION_139X139        32
#define CONFIGURATION_143X143        33
#define CONFIGURATION_147X147        34
#define CONFIGURATION_151X151        35
```

2.7.2. Data structure

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

```
typedef struct tagAZTECCONTEXT
{
    int   nEncodeMode;
    int   nCorrectionLevel;
    int     nConfigType;
    int     nConfiguration;
    int   nRune;
    int   nPixelSize;
    int   nMargin;
    int   nStructuredAppendCounter;
    int   nStructuredAppendIndex;
    int   nDataSize;
    BOOL bProcessTilde;
    BOOL bStructuredAppend;
    COLORREF clForeColor;
    COLORREF clBackColor;
    char *pFileID;
```



```
    char cEncodedData[3750];
}AZTECCONTEXT;
```

2.7.3. Function or procedure

2.7.3.1. InitAztecContext

The _ **InitAztecContext** function initilize the environment of Aztec encoder with default value.

void _InitAztecContext (_AZTECCONTEXT *pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoder, refer structure type AZTECCONTEXT

Return values

None

2.7.3.2. AztecEncode2Bitmap

The AztecEncode2Bitmap 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* _AztecEncode2Bitmap(_AZTECCONTEXT *pAztecCtx,int *pWidth,int *pHeight);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT

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 AZTEC barcode , pWidth \pHeight ;otherwise , return NULL.

2.7.3.3. AztecRegister

The _AztecRegister function help user register the SDK before running. AIPSYS will send user the register information including mail-box string and regcode.

bool _AztecRegister(const char *pMailBox,const 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 "AztecEncodeLib.h"
        .....
        _AZTECCONTEXT tagAztec;
```

```
int width,height;

unsigned char *pMap;



_InitAztecContext(&tagAztec);

tagAztec.nEncodeMode= ENC_NORMAL;

tagAztec.nConfigType = CONFIGURATION_FULL;

tagAztec.nConfiguration = -1;

tagAztec.nCorrectionLevel = 1;

memcpy(tagAztec.cEncodedData,"http://www.aipsys.com",23);

tagAztec.nDataSize = 23;

tagAztec.nMargin = 10;

tagAztec.nPixelSize = 4;

pMap = _AztecEncode2Bitmap(&tagAztec,&width,&height);

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);

....
```

LIBRARY for linking
libAztecEncodeLibLinuxARM.a

2.8. Library for MAC

2.8.1 Constants

Encoding scheme

ENC_NORMAL 0;
ENC_BINARY 1;

ConfigType

CONFIGURATION_ANY 0;
CONFIGURATION_COMPACT 1;
CONFIGURATION_FULL 2;

Correction Level

0~35%

Configuration

#define CONFIGURATION_ANY 0
#define CONFIGURATION_COMPACT 1
#define CONFIGURATION_FULL 2

#define CONFIGURATION_AUTO -1
#define CONFIGURATION_15X15_COMPACT 0
#define CONFIGURATION_19X19 = 1
#define CONFIGURATION_19X19_COMPACT 2
#define CONFIGURATION_23X23 3
#define CONFIGURATION_23X23_COMPACT 4
#define CONFIGURATION_27X27 5
#define CONFIGURATION_27X27_COMPACT 6
#define CONFIGURATION_31X31 7
#define CONFIGURATION_37X37 8
#define CONFIGURATION_41X41 9
#define CONFIGURATION_45X45 10
#define CONFIGURATION_49X49 11
#define CONFIGURATION_53X53 12
#define CONFIGURATION_57X57 13
#define CONFIGURATION_61X61 14
#define CONFIGURATION_67X67 15
#define CONFIGURATION_71X71 16
#define CONFIGURATION_75X75 17
#define CONFIGURATION_79X79 18
#define CONFIGURATION_83X83 19
#define CONFIGURATION_87X87 20
#define CONFIGURATION_91X91 21
#define CONFIGURATION_95X95 22
#define CONFIGURATION_101X101 23
#define CONFIGURATION_105X105 24
#define CONFIGURATION_109X109 25
#define CONFIGURATION_113X113 26


```
#define CONFIGURATION_117X117        27
#define CONFIGURATION_121X121        28
#define CONFIGURATION_125X125        29
#define CONFIGURATION_131X131        30
#define CONFIGURATION_135X135        31
#define CONFIGURATION_139X139        32
#define CONFIGURATION_143X143        33
#define CONFIGURATION_147X147        34
#define CONFIGURATION_151X151        35
```

2.8.2. Data structure

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

```
typedef struct tagAZTECCONTEXT
{
    int   nEncodeMode;
    int   nCorrectionLevel;
    int     nConfigType;
    int     nConfiguration;
    int   nRune;
    int   nPixelSize;
    int   nMargin;
    int   nStructuredAppendCounter;
    int   nStructuredAppendIndex;
    int   nDataSize;
    BOOL bProcessTilde;
    BOOL bStructuredAppend;
    COLORREF clForeColor;
    COLORREF clBackColor;
    char *pFileID;
    char cEncodedData[3750];
}AZTECCONTEXT;
```


2.8.3. Function or procedure

2.8.3.1. InitAztecContext

The _ **InitAztecContext** function initilize the environment of Aztec encoder with default value.

void _InitAztecContext (_AZTECCONTEXT *pAztecCtx);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoder, refer structure type AZTECCONTEXT

Return values

None

2.8.3.2. AztecEncode2Bitmap

The AztecEncode2Bitmap 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* _AztecEncode2Bitmap(_AZTECCONTEXT *pAztecCtx,int *pWidth,int *pHeight);

Parameters

pAztecCtx

[in] define the Aztec attributes for encoding, refer structure type AZTECCONTEXT

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 AZTEC barcode , pWidth \pHeight ;otherwise , return NULL.

2.8.3.3. AztecRegister

The _AztecRegister function help user register the SDK before running. AIPSYS will send user the register information including mail-box string and regcode.

bool _AztecRegister(const char *pMailBox,const 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 Objective-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 "AztecEncodeLib.h"
        .....
        _AZTECCONTEXT tagAztec;
        int width,height;
        unsigned char *pMap;
```



```
_InitAztecContext(&tagAztec);
tagAztec.nEncodeMode= ENC_NORMAL;
tagAztec.nConfigType = CONFIGURATION_FULL;
tagAztec.nConfiguration = -1;
tagAztec.nCorrectionLevel = 1;
memcpy(tagAztec.cEncodedData,"http://www.aipsys.com",23);
tagAztec.nDataSize = 23;
tagAztec.nMargin = 10;
tagAztec.nPixelSize = 4;
pMap = _AztecEncode2Bitmap(&tagAztec,&width,&height);
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);
....
```

LIBRARY for linking

libAztecEncodeLibMAC.a

3. Decoder SDK

3.1. Static link library

3.1.1 Data structure

The following data structure define the properties of the result of Aztec barcode decoder.

```
typedef struct tagResult
{
    BYTE *pData;                //result content
    int nSize;                  //content length
    POINT ptsBarcode[4];        //barcode position
}Result;
```

3.1.2. Function or procedure

3.1.2.1. _ LoadImageEx

The method load the image file and return the BITMAP handler, it use GDI+ library.

HBITMAP __stdcall _LoadImageEx(char *lpImageFile);

Parameters

lpImageFile : image file including path.

Return values

The Bitmap handler.

3.1.2.2. _ AztecReaderRegister

The function help user register the SDK before running. AIPSYS will send user the register information including mail-box string and regcode.

BOOL __stdcall _AztecReaderRegister(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 .

3.1.2.3. _ AztecDecodeGrayImage

The function will detect and decode Aztec barcode from data buffer of gray image.

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

Parameters

pGray: Data buffer of gray image
width: image width
height: image height

Return values

Return Result including content, position if decoding successfully, otherwise return NULL.

3.1.2.4. _ AztecDecode

The function will detect and decode Aztec barcode from bitmap handler

Result * __stdcall _AztecDecode(HBITMAP hImage);

Parameters

hImage: bitmap handler of image

Return values

Return Result including content, position if decoding successfully, otherwise return NULL.

3.1.2.5. _ AztecDecodeImageFile

The function will detect and decode Aztec barcode from image file

Result * __stdcall _AztecDecodeImageFile(char *pImageFile);

Parameters

pImageFile: image file including path.

Return values

Return Result including content, position if decoding successfully, otherwise return NULL.

3.1.2.6. _ AztecFree

The function release the memory of result.

void __stdcall _AztecFree(Result *r);

Parameters

r: Result pointer.

Return values

N/A.

3.2. Dynamic link library

3.2.1 Data structure

The following data structure define the properties of the result of Aztec barcode decoder.

```
typedef struct tagResult
{
    BYTE *pData;                //result content
    int nSize;                  //content length
    POINT ptsBarcode[4];        //barcode position
}Result;
```

3.2.2. Function or procedure

3.2.2.1. LoadImageEx

The method load the image file and return the BITMAP handler, it use GDI+ library.

HBITMAP __stdcall LoadImageEx(char *lpImageFile);

Parameters

lpImageFile : image file including path.

Return values

The Bitmap handler.

3.2.2.2. AztecReaderRegister

The function help user register the SDK before running. AIPSYS will send user the register information including mail-box string and regcode.

BOOL __stdcall AztecReaderRegister(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 .

3.2.2.3. AztecDecodeGrayImage

The function will detect and decode Aztec barcode from data buffer of gray image.

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

Parameters

pGray: Data buffer of gray image

width: image width

height: image height

Return values

Return Result including content, position if decoding successfully, otherwise return NULL.

3.2.2.4. AztecDecode

The function will detect and decode Aztec barcode from bitmap handler

Result * __stdcall AztecDecode(HBITMAP hImage);

Parameters

hImage: bitmap handler of image

Return values

Return Result including content, position if decoding successfully, otherwise return NULL.

3.2.2.5. AztecDecodeImageFile

The function will detect and decode Aztec barcode from image file

Result * __stdcall AztecDecodeImageFile(char *pImageFile);

Parameters

pImageFile: image file including path.

Return values

Return Result including content, position if decoding successfully, otherwise return NULL.

3.2.2.6. AztecFree

The function release the memory of result.

void __stdcall AztecFree(Result *r);

Parameters

r: Result pointer.

Return values

N/A.

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

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	2.0
Dynamic Library	300222332	300222333	300222334	300222335	300222336	2.0
ActiveX	300222367	300222368	300222369	300222370	300222371	2.0
ASP Component		300222388	300222389	300222390	300222391	2.0
QRCode Encode SDK						
Static Library		300222413	300222284	300222285	300222286	2.0
Dynamic Library	300222309	300222312	300222314	300222317	300222320	2.0
ActiveX	300222343	300222344	300222347	300222350	300222355	2.0
ASP Component		300222376	300222377	300222377	300222379	2.0
DataMatrix Encode SDK						
Static Library		300222291	300222292	300222293	300222294	2.0
Dynamic Library	300222321	300222322	300222324	300222325	300222326	2.0
ActiveX	300222357	300222358	300222359	300222360	300222361	2.0
ASP Component		300222380	300222381	300222382	300222383	2.0
PDF417 Encode SDK						
Static Library		300222280	300222281	300222282	300222283	2.0
Dynamic Library	300222299	300222300	300222301	300222303	300222305	2.0
ActiveX	300222337	300222338	300222339	300222340	300222341	2.0
ASP Component		300222372	300222373	300222374	300222375	2.0
Aztec Encode SDK						
Static Library		300222288	300222414	300222289	300222290	2.0
Dynamic Library	300222323	300222327	300222329	300222330	300222331	2.0
ActiveX	300222362	300222363	300222364	300222365	300222366	2.0
ASP Component		300222384	300222385	300222386	300222387	2.0

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

6. Product Information Link

. Aztec encoder SDK

. PDF417 encoder SDK

. DataMatrix encoder SDK

. Aztec encoder SDK

. Linear 1D barcode encoder SDK

