QRCode Encode/Decode SDK v3.0

USER MANUAL

AIPSYS Software Laboratory

http://www.aipsys.com

Last Updated 21st March 2013

1. Introduction

1.1. QRCode Barcode introduction

About QRCode

QR Code is a kind of 2-D (two-dimensional) symbology developed by Denso Wave (a division of Denso Corporation at the time) and released in 1994 with the primary aim of being • ga symbol that is easily interpreted by scanner equipment.

QR Code (2D Code) contains information in both the vertical and horizontal directions, whereas a bar code contains data in one direction only. QR Code holds a considerably greater volume of information than a bar code.

QR Code provides the following features compared with conventional bar codes.

High Capacity Encoding of Data

While conventional bar codes are capable of storing a maximum of approximately 20 digits, QR Code is capable of handling several dozen to several hundred times more information. QR Code is capable of handling all types of data, such as numeric and alphabetic characters, Kanji, Kana, Hiragana, symbols, binary, and control codes. Up to 7,089 characters can be encoded in one symbol.

QR Code Data capacity	
Numeric only	Max. 7,089 characters
Alphanumeric	Max. 4,296 characters
Binary (8 bits)	Max. 2,953 bytes
Kanji, full-width Kana	Max. 1,817 characters

A QR Code symbol of this size can encode 300 alphanumeric characters

Small Printout Size

Since QR Code carries information both horizontally and vertically, QR Code is capable of encoding the same amount of data in approximately one-tenth the space of a traditional bar code. (For a smaller printout size, Micro QR Code is available

Kanji and Kana Capability.

As a symbology developed in Japan, QR Code is capable of encoding JIS Level 1 and Level 2 kanji character set.

In case of Japanese, one full-width Kana or Kanji character is efficiently encoded in 13 bits, allowing QR Code to hold more than 20% data than other 2D symbologies.

Dirt and Damage Resistant

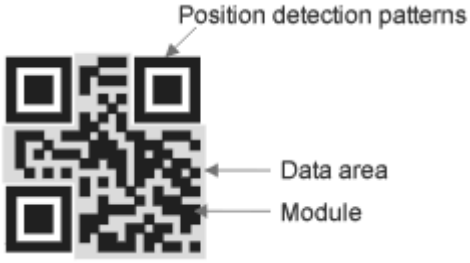
QR Code has error correction capability. Data can be restored even if the symbol is partially dirty or damaged. A maximum 30% of codewords*[1] can be restored*[2].

*1: A codeword is a unit that constructs the data area. In the case of QR Code, one codeword is equal to 8 bits.

*2: Data restoration may not be fully performed depending on the amount of dirt or damage..

Readable from any direction in 360 °

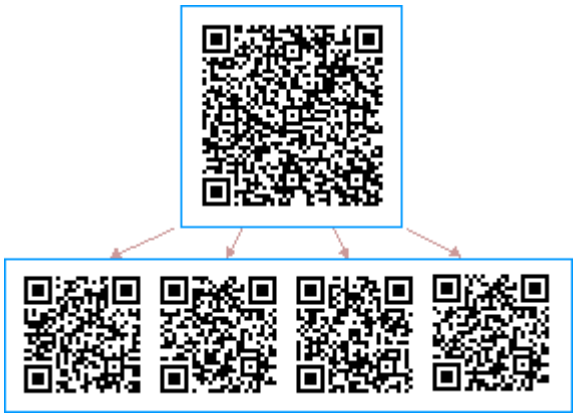
QR Code is capable of 360 degree (omni-directional), high speed reading. QR Code accomplishes this task through position detection patterns located at the three corners of the symbol. These position detection patterns guarantee stable high-speed reading, circumventing the negative effects of background interference.



Structured Append Feature

QR Code can be divided into multiple data areas. Conversely, information stored in multiple QR Code symbols can be reconstructed as single data symbols.

One data symbol can be divided into up to 16 symbols, allowing printing in a narrow area.

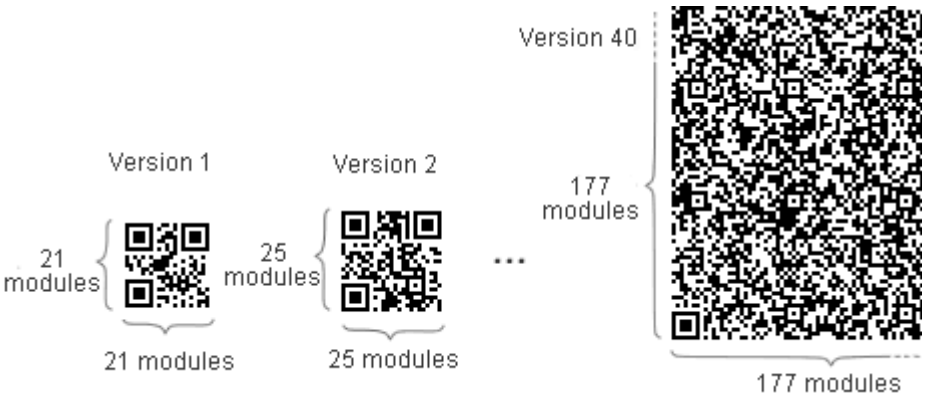


Symbol Version

The symbol versions of QR Code range from Version 1 to Version 40. Each version has a different module configuration or number of modules. (The module refers to the black and white dots that make up QR Code.) "Module configuration" refers to the number of modules contained in a symbol, commencing with Version 1 (21 × 21 modules) up to Version 40 (177 × 177 modules). Each higher version number comprises 4 additional modules per side.

Each QR Code symbol version has the maximum data capacity according to the amount of data, character type and error correction level. Check the maximum data capacity for each version.*Version and maximum data capacity table

In other words, as the amount of data increases, more modules are required to comprise QR Code, resulting in larger QR Code symbols.



Error Correction

QR Code has error correction capability to restore data if the code is dirty or damaged. Four error correction levels are available for users to choose according to the operating environment. Raising this level improves error correction capability but also increases the amount of data QR Code size.

To select error correction level, various factors such as the operating environment and QR Code size need to be considered.

Level Q or H may be selected for factory environment where QR Code get dirty, whereas Level L may be selected for clean environment with the large amount of data. Typically, Level M (15%) is most frequently selected.

QR Code Error Correction Capability*	
Level L	Approx.7%
Level M	Approx. 15%
Level Q	Approx. 25%
Level H	Approx. 30%

*Data restoration rate for total codewords (codeword is a unit that constructs the data area. One codeword of QR Code is equal to 8 bits.)

Error Correction Feature

The QR Code error correction feature is implemented by adding a Reed-Solomon Code*to the original data.

The error correction capability depends on the amount of data to be corrected. For example, if there are 100 codewords of QR Code to be encoded,50 of which need to be corrected, 100 codewords of Reed-Solomon Code are required, as Reed-Solomon Code requires twice the amount of codewords to be corrected.

In this case, the total codewords are 200, 50 of which can be corrected. Thus, the error correction rate for the total codewords is 25%. This corresponds to QR Code error correction Level Q.

In the example above, the error correction rate for QR Code codewords can be considered as 50%. However, it is not always the case that codewords of not Reed-Solomon Code but only QR Code are susceptible to dirt and damage.QR Code therefore represents its error correction rate as a ratio of the total codewords.

(*) Reed-Solomon Code is a mathematical error correction method used for music CDs etc. The technology was originally developed as a measure against communication noise for artificial satellites and planetary probes. It is capable of making a correction at the byte level, and is suitable for concentrated burst errors

1.2 QRCode parameters introduction

Version : There are 40 sizes of QR Code symbols (called Version 1, Version 2 till Version 40). Version 1 measures 21 modules * 21 modules, Version 2 measures 25 modules * 25 modules and so on. Version 40 measures 177 modules * 177 modules.

correctionLevel: Supports 4 error correction levels

0: L (7% of the symbol codewords).
1: M (15% of the symbol codewords).
2: Q (25% of the symbol codewords).
3: H (30% of the symbol codewords)

Encoding:

0: Alphanumeric characters , digits 0 - 9; upper case letters A -Z and nine other characters: space, $ % * + - . / :
1: Byte data (bytes 0-255)
2: Numeric data (digits 0-9).
3: Kanji characters (hexadecimal values 8140 -9FFC and E040 - EBBF)
4: Automatic adapt char type

Eci: Extended Channel Interpretation (optional): enables data using character sets other than the default set (e.g. Arabic, Cyrillic, Greek)

fnc1Mode: FNC1 mode is used for messages containing data formatted either in accordance with the UCC/EAN Application Identifiers standard or in accordance with a specific industry standard previously agreed with AIM International.

structuredAppendIndex:

structuredAppend : This allows files of data to be represented logically in up to 16 QR Code symbols.

processTilde: use the tilde character "~" to recognize special characters when "Apply Tilde" or "Process Tilde" is enabled. The following tilde options are available:

~dNNN: Represents the ASCII character encoded by the 3 digits NNN. For example, ~d009 represents a tab, ~d013 represents a return and ~d065 represents the character 'A'.

~1: Represents the character FNC1. When FNC1 appears in the first position (or in the fifth position of the first symbol of a Structured Append), it indicates that the data conforms to the UCC/EAN Application Identifier standard format.

When enabled it works as follows:

- ~~: will be replaced with ~
- ~dxxx: will be replaced by the character whose ascii code is xxx. For example ~d065 will be replaced with A

StructuredAppendCounter(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 16. If this is not set, or set to 1, the symbol is assumed to be standalone (no structured append used).

StructuredAppendIndex(int): When the symbol is part of an Structured Append set, this is the number of this symbol in the set. Valid values : 1 to 16.

1.3 Data capacity for QRCode versions

Version	Error Correction Level	Numeric	Alphanumeric	Byte	Kanji
1	L	41	25	17	10
	M	34	20	14	8
	Q	27	16	11	7
	H	17	10	7	4
2	L	77	47	32	20
	M	63	38	26	16
	Q	48	29	20	12
	H	34	20	14	8

3	L M Q H	127 101 77 58	77 61 47 35	53 42 32 24	32 26 20 15
4	L M Q H	187 149 111 82	114 90 67 50	78 62 46 34	48 38 28 21
5	L M Q H	255 202 144 106	154 122 87 64	106 84 60 44	65 52 37 27
6	L M Q H	322 255 178 139	195 154 108 84	134 106 74 58	82 65 45 36
7	L M Q H	370 293 207 154	224 178 125 93	154 122 86 64	95 75 53 39
8	L M Q H	461 365 259 202	279 221 157 122	192 152 108 84	118 93 66 52
9	L M Q H	552 432 312 235	335 262 189 143	230 180 130 98	141 111 80 60
10	L M Q H	652 513 364 288	395 311 221 174	271 213 151 119	167 131 93 74
11	L M Q H	772 604 427 331	468 366 259 200	321 251 177 137	198 155 109 85
12	L M Q	883 691 489	535 419 296	367 287 203	226 177 125

	H	374	227	155	96
13	L	1022	619	425	262
	M	796	483	331	204
	Q	580	352	241	149
	H	427	259	177	109
14	L	1101	667	458	282
	M	871	528	362	223
	Q	621	376	258	159
	H	468	283	194	120
15	L	1250	758	520	320
	M	991	600	412	254
	Q	703	426	292	180
	H	530	321	220	136
16	L	1408	854	586	361
	M	1082	656	450	277
	Q	775	470	322	198
	H	602	365	250	154
17	L	1548	938	644	397
	M	1212	734	504	310
	Q	876	531	364	224
	H	674	408	280	173
18	L	1725	1046	718	442
	M	1346	816	560	345
	Q	948	574	394	243
	H	746	452	310	191
19	L	1903	1153	792	488
	M	1500	909	624	384
	Q	1063	644	442	272
	H	813	493	338	208
20	L	2061	1249	858	528
	M	1600	970	666	410
	Q	1159	702	482	297
	H	919	557	382	235
21	L	2232	1352	929	572
	M	1708	1035	711	438
	Q	1224	742	509	314
	H	969	587	403	248
22	L	2409	1460	1003	618
	M	1872	1134	779	480

	Q	1358	823	565	348
	H	1056	640	439	270
23	L	2620	1588	1091	672
	M	2059	1248	857	528
	Q	1468	890	611	376
	H	1108	672	461	284
24	L	2812	1704	1171	721
	M	2188	1326	911	561
	Q	1588	963	661	407
	H	1228	744	511	315
25	L	3057	1853	1273	784
	M	2395	1451	997	614
	Q	1718	1041	715	440
	H	1286	779	535	330
26	L	3283	1990	1367	842
	M	2544	1542	1059	652
	Q	1804	1094	751	462
	H	1425	864	593	365
27	L	3517	2132	1465	902
	M	2701	1637	1125	692
	Q	1933	1172	805	496
	H	1501	910	625	385
28	L	3669	2223	1528	940
	M	2857	1732	1190	732
	Q	2085	1263	868	534
	H	1581	958	658	405
29	L	3909	2369	1628	1002
	M	3035	1839	1264	778
	Q	2181	1322	908	559
	H	1677	1016	698	430
30	L	4158	2520	1732	1066
	M	3289	1994	1370	843
	Q	2358	1429	982	604
	H	1782	1080	742	457
31	L	4417	2677	1840	1132
	M	3486	2113	1452	894
	Q	2473	1499	1030	634
	H	1897	1150	790	486
32	L	4686	2840	1952	1201

	M	3693	2238	1538	947
	Q	2670	1618	1112	684
	H	2022	1226	842	518
33	L	4965	3009	2068	1273
	M	3909	2369	1628	1002
	Q	2805	1700	1168	719
	H	2157	1307	898	553
34	L	5253	3183	2188	1347
	M	4134	2506	1722	1060
	Q	2949	1787	1228	756
	H	2301	1394	958	590
35	L	5529	3351	2303	1417
	M	4343	2632	1809	1113
	Q	3081	1867	1283	790
	H	2361	1431	983	605
36	L	5836	3537	2431	1496
	M	4588	2780	1911	1176
	Q	3244	1966	1351	832
	H	2524	1530	1051	647
37	L	6153	3729	2563	1577
	M	4775	2894	1989	1224
	Q	3417	2071	1423	876
	H	2625	1591	1093	673
38	L	6479	3927	2699	1661
	M	5039	3054	2099	1292
	Q	3599	2181	1499	923
	H	2735	1658	1139	701
39	L	6743	4087	2809	1729
	M	5313	3220	2213	1362
	Q	3791	2298	1579	972
	H	2927	1774	1219	750
40	L	7089	4296	2953	1817
	M	5596	3391	2331	1435
	Q	3993	2420	1663	1024
	H	3057	1852	1273	784

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

2.1.1 Constants

Encoding scheme

ENC_ALPHA 0;
ENC_BYTE 1;
ENC_NUMERIC 2;
ENC_CHINESE 3;

FNC1 mode

FNC1_MODE_NO 0;
FNC1_MODE_FIRST 1;
FNC1_MODE_SECOND 2;

Correction Level

CORRECTION_LEVEL_L 0;
CORRECTION_LEVEL_M 1;
CORRECTION_LEVEL_Q 2;
CORRECTION_LEVEL_H 3;

Version for Micro QRCode

MICROQR_VER_M1 1;
MICROQR_VER_M2 2;
MICROQR_VER_M3 3;
MICROQR_VER_M4 4;

2.1.2 Data structure

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

```
typedef struct _tagQRCODECONTEXT
{
    int  nVersion;    //the selected version of QRCode 1~40
```

```
int  nCorrectionLevel;    //error correction level
                    CORRECTION_LEVEL_L = 0;
                    CORRECTION_LEVEL_M = 1;
                    CORRECTION_LEVEL_Q = 2;
                    CORRECTION_LEVEL_H = 3;
int  nEncodeMode;                  //encoding mode,
                    ENC_ALPHA = 0;
                    ENC_BYTE = 1;
                    ENC_NUMERIC  = 2;
                    ENC_KANJI_CHINESE = 3
int  nEci;
int  nFnc1Mode;                 //fnc mode
                    FNC1_MODE_NO  = 0;
                    FNC1_MODE_FIRST  = 1;
                    FNC1_MODE_SECOND = 2;
int  nStructuredAppendIndex;
int  nStructuredAppendCounter;
int  nApplicationIndicator;
bool bStructuredAppend;       //structure appending mode if true
bool bProcessTilde;       //processing tilde if true
bool bAutoConfigurate;       //configurate automaticlly if true
char cData[7100];          //data to be encoded
int  nSize;         //if data is binary , use this variable to tell systemthe input length;
BYTE nMargin;            //size of white space of barcode
BYTE nPixelSize;         //the the pixel size when drawing barcode
COLORREF clBackGround; //background color
COLORREF clForeGround; //foreground color
} _QRCODECONTEXT,*_LPQRCODECONTEXT;
```

2.1.3. Function or procedure

2.1.3.1. _InitQRCodeContext

The _InitQRCodeContext function initilize the environment of QRCode encoding with default value.

void __stdcall *_InitQRCodeContext (*_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _LPQRCODECONTEXT

Return values

None

2.1.3.2. _QRCodeEncode2File

The _QRCodeEncode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall _QRCodeEncode2File (_QRCODECONTEXT * pQRCodeCtx,
LPCTSTR lpImageFile);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type
_ QRCODECONTEXT
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. _QRCodeEncode2Bitmap

The **_QRCodeEncode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

HBITMAP __stdcall _QRCodeEncode2Bitmap (
_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type
_QRCODECONTEXT

Return values

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

2.1.3.4. _MicroQREncode2Bitmap

The **_MicroQREncode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

HBITMAP __stdcall _QRCodeEncode2Bitmap (
_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT

Return values

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

2.1.3.5. _MicroQREncode2File

The _MicroQREncode2File function encode the data inputed with the defined attributes and save the barcode to an image file

BOOL __stdcall _MicroQREncode2File (_QRCODECONTEXT * pQRCodeCtx,
LPCTSTR lpImageFile);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _ QRCODECONTEXT

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.

2.1.3.6. _FreeQRCodeContext

The _FreeQRCodeContext function free environment of the QRCode encoding

BOOL __stdcall _FreeQRCodeContext ();

Return values

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

2.1.4. Example for Microsoft visual C++

Example1

```
#include "QRCodeEncodeLib.h"
....
  _tagQRCODECONTEXT tQrCtx;
  _InitQRCodeContext(&tQrCtx);
  tQrCtx.nVersion = 2;
  tQrCtx.nCorrectionLevel = 0;
  tQrCtx.nEncodeMode = 0;
  tQrCtx.nEci = 0;
  tQrCtx.nFnc1Mode = 0;
  tQrCtx.bStructuredAppend = false;
  tQrCtx.bProcessTilde = true;
  tQrCtx.bAutoConfigurate = false;
  tQrCtx.nMargin = 10;
  tQrCtx.nPixelSize = 4;
  tQrCtx.clBackGround = RGB(255,255,255);
  tQrCtx.clForeGround = RGB(255,0,0);
  sprintf(tQrCtx.cData,"http://www.aipsys.com");
  _QRCodeEncode2File(&tQrCtx, "c:\\qrcode.bmp");
  _FreeQRCodeContext();
....
```

LIBRARY for linking
QRCodeEncodeLIB.LIB

2.2. Dynamic link library

2.2.1 Constants

Encoding scheme

ENC_ALPHA	0;
ENC_BYTE	1;

ENC_NUMERIC 2;
ENC_CHINESE 3;

FNC1 mode

FNC1_MODE_NO 0;
FNC1_MODE_FIRST 1;
FNC1_MODE_SECOND 2;

Correction Level

CORRECTION_LEVEL_L 0;
CORRECTION_LEVEL_M 1;
CORRECTION_LEVEL_Q 2;
CORRECTION_LEVEL_H 3;

2.2.2. Data structure

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

```
typedef struct  tagQRCODECONTEXT
{
    int  nVersion;   //the selected version of QRCode 1~40
    int  nCorrectionLevel;   //error correction level
                    CORRECTION_LEVEL_L = 0;
                    CORRECTION_LEVEL_M = 1;
                    CORRECTION_LEVEL_Q = 2;
                    CORRECTION_LEVEL_H = 3;
    int  nEncodeMode;                //encoding mode,
                    ENC_ALPHA = 0;
                    ENC_BYTE = 1;
                    ENC_NUMERIC  = 2;
                    ENC_KANJI_CHINESE = 3
    int  nEci;
    int  nFnc1Mode;                //fnc mode
                    FNC1_MODE_NO  = 0;
                    FNC1_MODE_FIRST  = 1;
                    FNC1_MODE_SECOND = 2;
    int  nStructuredAppendIndex;
    int  nStructuredAppendCounter;
    int  nApplicationIndicator;
    bool bStructuredAppend;      //structure appending mode if true
    bool bProcessTilde;      //processing tilde if true
    bool bAutoConfigurate;      //configurate automaticlly if true
```



```
char cData[7100];          //data to be encoded
int  nSize;        //if data is binary , use this variable to tell systemthe input length;
BYTE nMargin;              //size of white space of barcode
BYTE nPixelSize;           //the the pixel size when drawing barcode
COLORREF clBackGround; //background color
COLORREF clForeGround; //foreground color
}QRCODECONTEXT,* LPQRCODECONTEXT;
```

2.2.2. Function or procedure

2.2.2.1. InitWorkSpace

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

void __stdcall _InitWorkSpace(QRCODECONTEXT *pQRCodeCtx);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type QRCODECONTEXT

Return values

None

2.2.2.2. QRCodeEncode2File

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

BOOL __stdcall QRCodeEncode2File (QRCODECONTEXT *pQRCodeCtx,
LPCTSTR lpImageFile);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type QRCODECONTEXT

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. QRCodeEncode2Bitmap

The **QRCodeEncode2Bitmap** function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

HBITMAP __stdcall QRCodeEncode2Bitmap (
QRCODECONTEXT *pQRCodeCtx);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type QRCODECONTEXT

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

2.2.2.4. FreeWorkSpace

The FreeWorkSpace function free environment of the QRCode encoding

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 "QRCodeEncodeDLL.h"
.....
    tagQRCODECONTEXT tQrCtx;

    InitWorkSpace(&tQrCtx);
    tQrCtx.nVersion = 2;
    tQrCtx.correctionLevel = 0;
```



```
    tQrCtx.nEncodeMode = 1;
    tQrCtx.nEci = 0;
    tQrCtx.nFnc1Mode = 0;
// tQrCtx.nStructuredAppendIndex;
// tQrCtx.nStructuredAppendCounter;
// tQrCtx.nApplicationIndicator;
    tQrCtx.bStructuredAppend = false;
    tQrCtx.bProcessTilde = true;
    tQrCtx.bAutoConfigurate = false;
    tQrCtx.nMargin = 10;
    tQrCtx.nPixelSize = 4;
// tQrCtx.clBackGround = RGB(255,255,255);
    tQrCtx.clForeGround = RGB(255,0,0);
    sprintf(tQrCtx.cData,"http://www.aipsys.com");
    QRCodeEncode2File(&tQrCtx, "c:\\qrcode.bmp");
    FreeWorkSpace();
....

LIBRARY for linking
QRCodeEncodeDLL.lib
RUNTIME LIBRARY
QRCodeEncodeDLL.DLL
```

2.2.4. Example for Borland Delphi

2.2.4.1. Redeclaration of the data type and function

```
LPQRCODECONTEXT = ^TQRCODECONTEXT;
                  TQRCODECONTEXT = record
                  nVersion : integer;
                  nCorrectionLevel : integer;
                  nEncodeMode : integer;
                  nEci : integer;
                  nFnc1Mode :integer;
                  nStructuredAppendIndex : integer;
                  nStructuredAppendCounter : integer;
                  nApplicationIndicator : integer;
                  bStructuredAppend : boolean;
                  bProcessTilde: boolean;
                  bAutoConfigurate: boolean;
                  cData: array[1..7100] of char;
```

```
                    nSize : integer;
                    nMargin : BYTE;
                    nPixelSize : BYTE;
                    clBackGround : TColor;
                    clForeGround : TColor;
                    end;

procedure InitWorkSpace(pQRCodeCtx : LPQRCODECONTEXT ); stdcall; external
        'QRCODEENCODEDLL.DLL';

function    QRCodeEncode2File(pQRCodeCtx : LPQRCODECONTEXT;lpImageFile :PChar) :
        boolean; stdcall;external 'QRCODEENCODEDLL.DLL';

function    QRCodeEncode2Bitmap(pQRCodeCtx : LPQRCODECONTEXT) :
        HBITMAP;stdcall;external 'QRCODEENCODEDLL.DLL';

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

2.2.4.2. Example

Example1

```
var
  ctx : TQRCODECONTEXT;
  s : string;
  i : Integer;
  bmp : HBITMAP;
  img : TBitmap;
  pCtx : LPQRCODECONTEXT;
begin
    pCtx := @ctx;
    InitWorkSpace(pCtx);
    ctx.nVersion := Combobox1.ItemIndex + 1;
    ctx.nCorrectionLevel := Combobox3.ItemIndex;
    ctx.nEncodeMode := Combobox4.ItemIndex;
    ctx.nFnc1Mode := Combobox5.ItemIndex;

    ctx.nStructuredAppendIndex := StrToInt(Edit1.Text);
    ctx.nStructuredAppendCounter := StrToInt(Edit6.Text);
    ctx.bStructuredAppend := CheckBox1.checked;
    ctx.bProcessTilde := CheckBox2.checked;
    ctx.bAutoConfigurate := CheckBox3.checked;
    ctx.nMargin := StrToInt(Edit3.Text);
```



```
    ctx.nPixelSize := StrToInt(Edit4.Text);
    ctx.clBackGround := ColorBox1.Color;
    ctx.clForeGround := ColorBox2.Color;
    if (Edit5.Text <> '') then
        StrMove(@ctx.cData,PChar(Edit5.Text),Length(Edit5.Text));
    image1.Picture.CleanupInstance;
    QRCodeEncode2File(pCtx,PChar('c:\1.bmp'));
    image1.Picture.LoadFromFile('c:\1.bmp');
    FreeWorkSpace();
end;
```

Example 2

```
var
  ctx : TQRCODECONTEXT;
  s : string;
  i : Integer;
  bmp : HBITMAP;
  img : TBitmap;
  pCtx : LPQRCODECONTEXT;
begin
    pCtx := @ctx;
    InitWorkSpace(pCtx);
    ctx.nVersion := Combobox1.ItemIndex + 1;
    ctx.nCorrectionLevel := Combobox3.ItemIndex;
    ctx.nEncodeMode := Combobox4.ItemIndex;
    ctx.nFnc1Mode := Combobox5.ItemIndex;

    ctx.nStructuredAppendIndex := StrToInt(Edit1.Text);
    ctx.nStructuredAppendCounter := StrToInt(Edit6.Text);
    ctx.bStructuredAppend := CheckBox1.checked;
    ctx.bProcessTilde := CheckBox2.checked;
    ctx.nAutoConfigurate := CheckBox3.checked;
    ctx.nMargin := StrToInt(Edit3.Text);
    ctx.nPixelSize := StrToInt(Edit4.Text);
    ctx.clBackGround := ColorBox1.Color;
    ctx.clForeGround := ColorBox2.Color;
    if (Edit5.Text <> '') then
        StrMove(@ctx.cData,PChar(Edit5.Text),Length(Edit5.Text));
    image1.Picture.CleanupInstance;
    bmp := QRCodeEncode2Bitmap(pCtx);
    img := TBitmap.Create;
    img.Handle := bmp;
    image1.Picture.Assign(img);
```



```
    img.Free;
    ShowMessage('Make QRCode barcode successfully');
    FreeWorkSpace();
end;
```

2.2.5. Example for Microsoft visual Basic

2.2.6. Example for CA Visual Objects

2.2.6.1. Redeclaration of the data type and function

```
_dll FUNC FreeWorkSpace() as LOGIC PASCAL:QRcodeEncodeDLL.FreeWorkSpace

_dll FUNC InitWorkSpace(stCode) as void PASCAL:QRcodeEncodeDLL.InitWorkSpace

_dll FUNC QRCodeEncode2File(stCode as ptr,cBitmap as psz) as  LOGIC
      PASCAL:QRcodeEncodeDLL.QRCodeEncode2File



   _dll FUNC Encode2File(pData as psz,nCorrectionLevel as int,nMargin
        as int, nPixelSize as int,clBackground as DWORD, clForeGround
        as DWORD,cBitmap as psz) as   LOGIC
        PASCAL:QRcodeEncodeDLL.Encode2File

      STRUCTURE QRCODECONTEXT ALIGN 1
                    MEMBER nSelectedVersion as int
                    MEMBER nCorrectionLevel as int
                    MEMBER nEncodeMode as int
                    MEMBER nEci as int
                    MEMBER nFnc1Mode as int
                    MEMBER nStructuredAppendIndex as int
                    MEMBER nStructuredAppendCounter as int
                    MEMBER nApplicationIndicator as int
                    MEMBER bProcessTilde as byte
                    MEMBER bStructuredAppend as byte
                    MEMBER bAutoConfigurate as byte
                    MEMBER dim cData[7100+1] as byte
                    MEMBER nSize as int
                    MEMBER nMargin as byte
```



```
MEMBER nPixelSize as byte
MEMBER clBackGround as DWORD
MEMBER clForeGround as DWOR
```

2.2.6.2. Example

```
METHOD MakeQRCode1( ) CLASS TestWindow


  LOCAL stQRCode as QRCODECONTEXT
  LOCAL lSucceed:=.F. as LOGIC
  LOCAL oWarningBox as EveWarningBox
  LOCAL oApp as App
  LOCAL cText as STRING
  oApp:=GetAppObject()
  cText:="Test"
  stQRCode := MemAlloc(_sizeof(QRCODECONTEXT))
  InitWorkSpace(stQRCode)
  stQRCode.nSelectedVersion:=3
  stQRCode.nCorrectionLevel:=1
  stQRCode.nEncodeMode:=4
*  stQRCode.nEci:=0
*  stQRCode.nFnc1Mode:=0
*  stQRCode.nStructuredAppendIndex:=0
*  stQRCode.nStructuredAppendCounter:=0
*  stQRCode.nApplicationIndicator:=0
  stQRCode.nSize :=Len(cText)
*  stQRCode.bStructuredAppend:=false
*  stQRCode.bProcessTilde:=true
*  stQRCode.bAutoConfigurate:=false
  stQRCode.nMargin:=15
  stQRCode.nPixelSize:=4
  IF stQRCode.nSize < 10
            MemCopyString(@stQRCode.cData, PadR(cText,10," "), 10)
  ELSE
          MemCopyString(@stQRCode.cData, cText, stQRCode.nSize)
  ENDIF
  lSucceed:=QRCodeEncode2File(stQRCode,String2Psz("c:\temp\qrcodetest.bmp"))
  IF lSucceed = .F.
     oWarningBox:=EveWarningBox{,"No QR code created!"+CHR(13)+"Program will be
closed"}
```



```
        oWarningBox:Show()
    ELSE

        oWarningBox:=EveWarningBox{,"QR code created
(c:\temp\qrcodetest.bmp)!"+CHR(13)+"Program will be closed"}
        oWarningBox:Show()

    ENDIF
    FreeWorkSpace()
    MemFree(stQRCode)
    self:EndWindow()
    oApp:Quit()
```

2.3. ActiveX

2.3.1. Properties

2.3.1.1. Version

preferred version of QRCode 1~40

short PreferredVersion

2.3.1.3. CorrectionLevel

The property set the error correction level

```
CORRECTION_LEVEL_L = 0;
    CORRECTION_LEVEL_M = 1;
    CORRECTION_LEVEL_Q = 2;
    CORRECTION_LEVEL_H = 3;
```

short **CorrectionLevel**

2.3.1.4. EncodeMode

The property set the encoding scheme of QRCode barcode

ENC_ALPHA = 0;
ENC_BYTE = 1;

ENC_NUMERIC = 2;
ENC_KANJI_CHINESE = 3

short EncodeMode

2.3.1.5. EciMode

The property set the eci mde of QRCode barcode

short EciMode

2.3.1.6. Fnc1Mode

The property set the fnc1 mode of QRCode barcode

short Fnc1Mode

2.3.1.7. StructuredAppend

The property set if QRCode is structured append

bool StructuredAppend

2.3.1.8. ProcessTilde

The property set if QRCode is process tilde

bool ProcessTilde

2.3.1.9. AutoConfigurate

The property set if version of QRCode is auto configurated

bool AutoConfigurate

2.3.1.10. Margin

The property set barcode Margin

bool Margin

2.3.1.11. PixelSize

The property set module Size of barcode image

bool PixelSize

2.3.1.12. ForeGroundColor

The property set the Foreground color of DataMatrix barcode

OLE_COLOR ForeGroundColor

2.3.1.13. BackGroundColor

The property set the Background color of DataMatrix barcode

OLE_COLOR BackGroundColor

2.3.1.7. TextData

The property set the data to be encoded

BSTR TextData

2.3.2. Methods

2.3.2.1. Encode2ImageFile

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

boolean Encode2ImageFile(BSTR lpImageFile);

Parameters

lpImageFile

[in] specify the barcode image file to be saved

Return values

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

2.3.3. Register activeX component

Regsvr32 QRCodeEncodeOcx.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 QRCodeEncodeOcx.Ocx

```
#include “QRCodeEncodeOcx.h”

    CQRCodeEncodeOcx    objQRCode;
    objQRCode.TextData = "http://www.aipsys.com";
    objQRCode.AutoConfigurate = true;
    objQRCode.EncodMode = 1;
     objQRCode.PixelSize = 4;
    objQRCode.ForeGroundColor = RGB(255,0,0);
    objQRCode.Margin = 10;
    objQRCode.Encode2ImageFile("c:\\qr.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
    …    QRCodeEncodeOcx_TLB;
objQR :    TQRCodeEncodeOcx;
begin
    objQR.TextData := ‘http://www.aipsys.com’;
    objQR.PixelSize    := 4;
    objQR.EncodeMode    := 1;
      objQR. AutoConfigurate := true;
    objQR.ForeGroundColor    := &HFF00FF
    objQR.Margin    := 10
```

```
    objQR.Encode2ImageFile('c:\qr.gif');
end;
```

2.3.6. Example for Microsoft visual Basic

```
Private Sub Command1_Click()
        QRCodeEncodeOCX1.TextData = "http://www.aipsys.com"
        QRCodeEncodeOCX1.PixelSize = 4
        QRCodeEncodeOCX1.EncodeMode = 1
        QRCodeEncodeOCX1.ForeGroundColor = &HFF00FF
        QRCodeEncodeOCX1.Margin = 10
        QRCodeEncodeOCX1.CorrectionLevel = 1
        QRCodeEncodeOCX1.SelectedVersion = 3
        QRCodeEncodeOCX1.ProcessTilde = True
        QRCodeEncodeOCX1.Encode2ImageFile("c:\qrcode.gif")
End Sub
```

2.4. ASP Control for server side

2.4.1. Properties

2.4.1.1. nVersion

The property set the selected version of QRCode 1~40

short nVersion

2.4.1.2. nPixelSize

The property set the module width of QRCode barcode

short **nPixelSize**

2.4.1.5. nMode

The property set the encoding mode of QRCode barcode

short nMode

2.4.1.6. nMargin

The property set the margin of QRCode barcode

short nMargin

2.4.1.7. nLevel

The property set the error correction level of QRCode barcode

short nLevel

2.4.1.8. clForeGround

The property set the Foreground color of DataMatrix barcode

OLE_COLOR clForeGround

2.4.1.9. clBackGround

The property set the Background color of DataMatrix barcode

OLE_COLOR clBackGround

2.4.1.9. 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 QRCodeEncodeASP.DLL

2.4.4. Example for ASP

```
<%
    set obj = Server.CreateObject("QRCodeEncodeCOM.EncodeService")
    obj.InitWorkspace()
    obj.nVersion = 3        '  Version
    obj.nMargin = 5         '  Margin
    obj.nLevel = 1          '  Correction Level
```

```
    obj.nPixelSize   = 2      '   Module size
    obj.strText = "Http://www.aipsys.com&#65292;"
    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. Java SDK for Encoder

2.5.1. Constants

2.5.1.1. encoding mode

```
public static final int ENC_ALPHA = 0;              //Alpha mode
public static final int ENC_BYTE = 1;               //binary mode
public static final int ENC_NUMERIC = 2;            //Numeric mode(0~9)
public static final int ENC_KANJI = 3;              //KANJI mode
public static final int ENC_AUTO = 4;      //Select encoding mode automatically
```

2.5.1.2. Correction level

```
public static int CORRECTION_LEVEL_L;
public static int CORRECTION_LEVEL_M;
public static int CORRECTION_LEVEL_Q;
public static int CORRECTION_LEVEL_H;
```

2.5.1.3. FNC1 Mode

```
public static final int FNC1_MODE_NO = 0;
public static final int FNC1_MODE_FIRST = 1;
public static final int FNC1_MODE_SECOND = 2;
```

2.5.2. Methods

```
public ImageEncoder() { }      //constructor

public String getEncodedString()      // gett the string encoded

public void setEncodedString(String c)    //set the string encoded
```


public int getECI() //get ECI mode

public void setECI(int v) //set ECI mode

public int[] getEncodedBinary() //get Binary data encoded

public void setEncodedBinary(int[] c) //set Binary data encoded

public int getMargin() //get margin

public void setMargin(int d) //set margin of the barcode

public int getEncoding() //get the encoding mode

public void setEncoding(int d) //set the encoding mode see 2.5.1

public Color getBackColor() //get background color

public void setBackColor(Color c) //set background color of barcode

public Color getForeColor() //get barcode color

public void setForeColor(Color c) //set barcode color

public double getModuleSize() //get Module size of barcode

public void setModuleSize(int d) //set Module size of barcode

public void setVersion(int d) //Version= 1~40

public int getErrorCorrectionLevel() //get correction level set

public void setErrorCorrectionLevel(int d) //set correction level set see 2.5.1

public boolean getProcessTilde() //get if seting Process Tilde

public void setProcessTilde(boolean pt) //set Process Tilde mode

public int getFnc1Mode() //get FNC1 mode

public void setFnc1Mode(int mode) set FNC1 mode, see 2.5.1

public byte getApplicationIndicator() //get application indicator

public void setApplicationIndicator(byte b) //set application indicator

public boolean getAutoConfigurate() //get if the version choosen automaticly


```
public void setAutoConfigurate(boolean pt) //set if the version choosen automaticly
public void setStructuredAppend(boolean b) //get structed append mode
public boolean getStructuredAppend() //set structed append mode
public void setStructuredAppendCounter(int i) //get structured append counter
public int getStructuredAppendCounter()       //set structured append counter
public void setStructuredAppendIndex(int i) //get structured append index
public int getStructuredAppendIndex() //set structured append index
public boolean Encode2JPEGFile(String strFile)    //Encode to JPEG file
public Image encode2Image()return 0;    //encode to Image
```

2.5.3. Samples

```
import com.aipsys.barcode.qrcode.encoder.*;
//package com.aipsys.barcode.qrcode.encoder;
public class QRCodeDemo
{
    public QRCodeDemo()
    {
    }
    public static void main(String[] args) throws Exception
    {
        try
        {
            ImageEncoder iee = new ImageEncoder();
            iee.setAutoConfigurate(true);
```



```
                    iee.setEncodedString("http://www.aipsys.com");
                    iee.setEncoding(ImageEncoder.ENC_AUTO);
iee.setErrorCorrectionLevel(ImageEncoder.CORRECTION_LEVEL_H);
                    iee.setMargin(30);
                    iee.Encode2JPEGFile("c:\\qrcode1.jpg");
                    return;
                }
                catch (RuntimeException e)
                {
                            System.out.println(e.getMessage());
                }
        }

}
```

2.6. Library for IOS

2.6.1 Constants

```
#ifndef QRCODE_ENCODELIB_H
#define QRCODE_ENCODELIB_H
typedef unsigned int COLORREF;
typedef unsigned char BYTE;


#define ENC_ALPHA               0;
#define ENC_BYTE                1;
#define ENC_NUMERIC             2;
#define ENC_CHINESE             3;
#define ENC_AUTO                4;

#define FNC1_MODE_NO            0;
```

```
#define FNC1_MODE_FIRST          1;
#define FNC1_MODE_SECOND         2;

#define CORRECTION_LEVEL_L       0;
#define CORRECTION_LEVEL_M       1;
#define CORRECTION_LEVEL_Q       2;
#define CORRECTION_LEVEL_H       3;

//Version for Micro QRCode
#define MICROQR_VER_M1           1;
#define MICROQR_VER_M2           2;
#define MICROQR_VER_M3           3;
#define MICROQR_VER_M4           4;
```

2.6.2 Data structure

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

```
typedef struct _tagQRCODECONTEXT
{
    int       nVersion;
    int       nCorrectionLevel;
    int       nEncodeMode;
    int       nEci;
    int       nFnc1Mode;
    int       nStructuredAppendIndex;
    int       nStructuredAppendCounter;
    int       nApplicationIndicator;
    bool      bStructuredAppend;
    bool      bProcessTilde;
    bool      bAutoConfigurate;
    charcData[7100];
    int      nSize;
    BYTE   nMargin;
    BYTE   nPixelSize;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _QRCODECONTEXT,*_LPQRCODECONTEXT;
```


2.6.3. Function or procedure

2.6.3.1. _InitQRCodeContext

The _InitQRCodeContext function initilize the environment of QRCode encoding with default value.

void _InitQRCodeContext (_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _LPQRCODECONTEXT

Return values

None

2.6.3.2. _QRCodeEncode2Bitmap

The _QRCodeEncode2Bitmap function encode the data inputed with the defined attributes and return an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _QRCodeEncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT

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

2.6.3.3. _MicroQREncode2Bitmap

The _MicroQREncode2Bitmap function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

unsigned char * _MicroQREncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT
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 MicroQRCode barcode , pWidth \pHeight; otherwise , return NULL.

2.6.3.4. _QRCodeEncodeRegister

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

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

Parameters
pMailBox: Mail box used to generate the regcode
pRegCode: regcode generated with mail box string
Return values
Return TRUE if register the product successfully, otherwise return FALSE .

2.6.4. Example for Object C

Example1

```
#include "QRcodeEncodeLib.h"
```



```
.....
    _QRCODECONTEXT tagQRcode;
    int width,height;
    unsigned char *pMap;

    _InitQRcodeContext (&tagQRcode);
    tagQRcode.nEncodeMode= ENC_NORMAL;
    tagQRcode.nConfigType = CONFIGURATION_FULL;
    tagQRcode.nConfiguration = -1;
    tagQRcode.nCorrectionLevel = 1;
    memcpy(tagQRcode.cEncodedData,"http://www.aipsys.com",23);
    tagQRcode.nDataSize = 23;
    tagQRcode.nMargin = 10;
    tagQRcode.nPixelSize = 4;
    pMap = _QRcodeEncode2Bitmap(&tagQRcode,&width,&height);
    if(!pMap) return;

CGColorSpaceRef colorSpace = CGColorSpaceCreateDeviceRGB();
     CFDataRef rgbData = CFDataCreate(NULL,pMap,width*height*3);
     CGDataProviderRef provider = CGDataProviderCreateWithCFData(rgbData);
CGImageRef rgbImageRef =
CGImageCreate(width,height,8,24,width*3,colorSpace,kCGBitmapByteOrderDefault,provid
er,NULL,true,kCGRenderingIntentDefault);
     CFRelease(rgbData);
     CGDataProviderRelease(provider);
```



```
        CGColorSpaceRelease(colorSpace);


    /*We display the result on the image view (We need to change the orientation of the image so
    that the video is displayed correctly).

        Same thing as for the CALayer we are not in the main thread so ...*/

    UIImage *image= [UIImage imageWithCGImage:rgbImageRef scale:1.0
    orientation:UIImageOrientationRight];

        /*We relase the CGImageRef*/

        CGImageRelease(rgbImageRef);

        width = image.size.width;

        height = image.size.height;

    [imageView setImage:image];

    free(pMap);

    ....
```

LIBRARY for linking
QRCodeEncodeLibIOS.a

2.7. Library for Linux

2.7.1 Constants

```
#ifndef QRCODE_ENCODELIB_H
#define QRCODE_ENCODELIB_H
typedef unsigned int COLORREF;
typedef unsigned char BYTE;


#define ENC_ALPHA                0;
#define ENC_BYTE                 1;
#define ENC_NUMERIC              2;
#define ENC_CHINESE              3;
```



```
#define ENC_AUTO                4;

#define FNC1_MODE_NO            0;
#define FNC1_MODE_FIRST         1;
#define FNC1_MODE_SECOND        2;

#define CORRECTION_LEVEL_L      0;
#define CORRECTION_LEVEL_M      1;
#define CORRECTION_LEVEL_Q      2;
#define CORRECTION_LEVEL_H      3;

//Version for Micro QRCode
#define MICROQR_VER_M1          1;
#define MICROQR_VER_M2          2;
#define MICROQR_VER_M3          3;
#define MICROQR_VER_M4          4;
```

2.7.2 Data structure

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

```
typedef struct _tagQRCODECONTEXT
{
    int       nVersion;
    int       nCorrectionLevel;
    int       nEncodeMode;
    int       nEci;
    int       nFnc1Mode;
    int       nStructuredAppendIndex;
    int       nStructuredAppendCounter;
    int       nApplicationIndicator;
    bool      bStructuredAppend;
    bool      bProcessTilde;
    bool      bAutoConfigurate;
    charcData[7100];
    int      nSize;
    BYTE   nMargin;
    BYTE   nPixelSize;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _QRCODECONTEXT,*_LPQRCODECONTEXT;
```


2.7.3. Function or procedure

2.7.3.1. _InitQRCodeContext

The _InitQRCodeContext function initilize the environment of QRCode encoding with default value.

void _InitQRCodeContext (_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _LPQRCODECONTEXT

Return values

None

2.7.3.2. _QRCodeEncode2Bitmap

The _QRCodeEncode2Bitmap function encode the data inputed with the defined attributes and return an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _QRCodeEncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT

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

2.7.3.3. _MicroQREncode2Bitmap

The _MicroQREncode2Bitmap function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

unsigned char * _MicroQREncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT
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 MicroQRCode barcode , pWidth \pHeight; otherwise , return NULL.

2.7.3.4. _QRCodeEncodeRegister

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

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

Parameters
pMailBox: Mail box used to generate the regcode
pRegCode: regcode generated with mail box string
Return values
Return TRUE if register the product successfully, otherwise return FALSE .

2.7.4. Example for C/C++

2.7.4.1Example1

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;
}
```

2.7.4.2 Source code

```
#include "QRcodeEncodeLib.h"
.....
    _QRCODECONTEXT tagQRcode;
    int width,height;
    unsigned char *pMap;

    _InitQRcodeContext (&tagQRcode);
    tagQRcode.nEncodeMode= ENC_NORMAL;
    tagQRcode.nConfigType = CONFIGURATION_FULL;
    tagQRcode.nConfiguration = -1;
    tagQRcode.nCorrectionLevel = 1;
    memcpy(tagQRcode.cEncodedData,"http://www.aipsys.com",23);
    tagQRcode.nDataSize = 23;
    tagQRcode.nMargin = 10;
    tagQRcode.nPixelSize = 4;
    pMap = _QRcodeEncode2Bitmap(&tagQRcode,&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
QRCodeEncodeLibLinux.a

2.8. Library for Linux ARM

2.8.1 Constants

```
#ifndef QRCODE_ENCODELIB_H
#define QRCODE_ENCODELIB_H
typedef unsigned int COLORREF;
typedef unsigned char BYTE;


#define ENC_ALPHA                0;
#define ENC_BYTE                 1;
#define ENC_NUMERIC              2;
#define ENC_CHINESE              3;
#define ENC_AUTO                 4;

#define FNC1_MODE_NO             0;
#define FNC1_MODE_FIRST          1;
#define FNC1_MODE_SECOND         2;

#define CORRECTION_LEVEL_L       0;
#define CORRECTION_LEVEL_M       1;
#define CORRECTION_LEVEL_Q       2;
#define CORRECTION_LEVEL_H       3;

//Version for Micro QRCode
#define MICROQR_VER_M1           1;
#define MICROQR_VER_M2           2;
#define MICROQR_VER_M3           3;
#define MICROQR_VER_M4           4;
```

2.8.2 Data structure

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

```
typedef struct _tagQRCODECONTEXT
{
    int       nVersion;
    int       nCorrectionLevel;
    int       nEncodeMode;
    int       nEci;
    int       nFnc1Mode;
    int       nStructuredAppendIndex;
    int       nStructuredAppendCounter;
    int       nApplicationIndicator;
    bool      bStructuredAppend;
    bool      bProcessTilde;
    bool      bAutoConfigurate;
    charcData[7100];
    int      nSize;
    BYTE   nMargin;
    BYTE   nPixelSize;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _QRCODECONTEXT,*_LPQRCODECONTEXT;
```

2.8.3. Function or procedure

2.8.3.1. _InitQRCodeContext

The _InitQRCodeContext function initilize the environment of QRCode encoding with default value.

void _InitQRCodeContext (_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _LPQRCODECONTEXT

Return values

None

2.8.3.2. _QRCodeEncode2Bitmap

The _QRCodeEncode2Bitmap function encode the data inputed with the defined attributes and return an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _QRCodeEncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT
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 QRCode barcode , pWidth \pHeight; otherwise , return NULL.

2.8.3.3. _MicroQREncode2Bitmap

The _MicroQREncode2Bitmap function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

unsigned char * _MicroQREncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT
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 MicroQRCode barcode , pWidth \pHeight; otherwise , return NULL.

2.8.3.4. _QRCodeEncodeRegister

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

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

Parameters

pMailBox: Mail box used to generate the regcode

pRegCode: regcode generated with mail box string

Return values

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

2.8.4. Example for Object C/C++

2.8.4.1Example1

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;
    }
```

2.8.4.2 Source code

```
    #include "QRcodeEncodeLib.h"
    .....
        _QRCODECONTEXT tagQRcode;
        int width,height;
```



```
unsigned char *pMap;


_InitQRcodeContext (&tagQRcode);
tagQRcode.nEncodeMode= ENC_NORMAL;
tagQRcode.nConfigType = CONFIGURATION_FULL;
tagQRcode.nConfiguration = -1;
tagQRcode.nCorrectionLevel = 1;
memcpy(tagQRcode.cEncodedData,"http://www.aipsys.com",23);
tagQRcode.nDataSize = 23;
tagQRcode.nMargin = 10;
tagQRcode.nPixelSize = 4;
pMap = _QRcodeEncode2Bitmap(&tagQRcode,&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
QRCodeEncodeLibLinuxARM.a

2.9. Library for MAC

2.9.1 Constants

#ifndef QRCODE_ENCODELIB_H
#define QRCODE_ENCODELIB_H


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


#define ENC_ALPHA                0;
#define ENC_BYTE                 1;
#define ENC_NUMERIC              2;
#define ENC_CHINESE              3;
#define ENC_AUTO                 4;

#define FNC1_MODE_NO             0;
#define FNC1_MODE_FIRST          1;
#define FNC1_MODE_SECOND         2;

#define CORRECTION_LEVEL_L       0;
#define CORRECTION_LEVEL_M       1;
#define CORRECTION_LEVEL_Q       2;
#define CORRECTION_LEVEL_H       3;

//Version for Micro QRCode
#define MICROQR_VER_M1           1;
#define MICROQR_VER_M2           2;
#define MICROQR_VER_M3           3;
#define MICROQR_VER_M4           4;
```

2.9.2 Data structure

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

```
typedef struct _tagQRCODECONTEXT
{
    int     nVersion;
    int     nCorrectionLevel;
    int     nEncodeMode;
    int     nEci;
    int     nFnc1Mode;
    int     nStructuredAppendIndex;
    int     nStructuredAppendCounter;
    int     nApplicationIndicator;
    bool    bStructuredAppend;
    bool    bProcessTilde;
    bool    bAutoConfigurate;
```



```
    charcData[7100];
    int       nSize;
    BYTE    nMargin;
    BYTE    nPixelSize;
    COLORREF clBackGround;
    COLORREF clForeGround;
} _QRCODECONTEXT,*_LPQRCODECONTEXT;
```

2.9.3. Function or procedure

2.9.3.1. _InitQRCodeContext

The _InitQRCodeContext function initilize the environment of QRCode encoding with default value.

void _InitQRCodeContext (_QRCODECONTEXT * pQRCodeCtx);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _LPQRCODECONTEXT

Return values

None

2.9.3.2. _QRCodeEncode2Bitmap

The _QRCodeEncode2Bitmap function encode the data inputed with the defined attributes and return an RGB bitmap buffer, which is the pixel matrix and each pixel contains three bytes corresponding to R \ G \ B.

unsigned char * _QRCodeEncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx

[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT

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

2.9.3.3. _MicroQREncode2Bitmap

The _MicroQREncode2Bitmap function encode the data inputed with the defined attributes and return the bitmap handle of the QRCode barcode image

unsigned char * _MicroQREncode2Bitmap(_QRCODECONTEXT *pQRCodeCtx,int *pWidth,int *pHeight);

Parameters

pQRCodeCtx
[in] define the QRCode attributes for encoding, refer structure type _QRCODECONTEXT
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 MicroQRCode barcode , pWidth \pHeight; otherwise , return NULL.

2.9.3.4. _QRCodeEncodeRegister

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

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

Parameters

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

Return values

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

2.9.4. Example for Object C

2.9.4.1Example1

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;

    }
```

2.9.4.2 Source code

```
    #include "QRcodeEncodeLib.h"

    .....

        _QRCODECONTEXT tagQRcode;

        int width,height;

        unsigned char *pMap;


        _InitQRcodeContext (&tagQRcode);

        tagQRcode.nEncodeMode= ENC_NORMAL;

        tagQRcode.nConfigType = CONFIGURATION_FULL;

        tagQRcode.nConfiguration = -1;

        tagQRcode.nCorrectionLevel = 1;

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

        tagQRcode.nDataSize = 23;
```



```
        tagQRcode.nMargin = 10;

        tagQRcode.nPixelSize = 4;

        pMap = _QRcodeEncode2Bitmap(&tagQRcode,&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
QRCodeEncodeLibMAC.a

3. Decoder SDK

3.1. Static link library

Structure of the QRCode decoding result:

```
typedef struct tagResult
{
   BYTE *pData;     //result of decoding result
   int nSize;       //data length
   int xPos;        //Left top position of barcode at image
   int yPos;        // Left top position of barcode at image
} Result;
```

3.1.1. Function or procedure

3.1.1.1. _QRCodeDecodeImageFile

The _QRCodeDecodeImageFile function read QRCode figure from image and decode it to text or binary data.

Result * __stdcall _QRCodeDecodeImageFile(char *pImageFile,int *pCount);

Parameters:

pImageFile char *

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

pCount int

[out] the barcodes the function found and the number of the result returnd. The pointer need be allocate memory before calling the function.

Return values

Result *, the first result pointer. If having more barcodes, move the pointer to get the next barcode result.

NULL decode failure or no barcode found.

3.1.1.2. _QRCodeDecode

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

Result * __stdcall _QRCodeDecode (HBITMAP hImage,int *pCount);

Parameters:

hImage HBITMAP

[in] the bitmap handle containing QRCode figure,

pCount int

[out] the barcodes the function found and the number of the result returnd. The pointer need be allocate memory before calling the function.

Return values

Result *, the first result pointer. If having more barcodes, move the pointer to get the next barcode result.

NULL decode failure or no barcode found.

3.1.1.3. _QRCodeFree

The _QRCodeFree function free the result memory;

void _stdcall _QRCodeFree(Result *r,int nCount);

Parameters:

r Result *

[in] the result pointer;

nCount int

[in] number of the result;

Return values

N/A

3.1.2. Samples

3.1.2.1 Example for Microsoft visual C++

```
#include “QRCodeDecodeLib.h”
……
Result *pResult;
int   nBarcodes;
char *pFile = “c:\\qrcode.jpg”;

pResult = _QRCodeDecodeImageFile(pFile,&nBarcodes);
if(pResult)
{
     char buf[8192];
     Result *p = pResult;
     for (int I =0; I < nBarcodes; I++)
     {
          memset(buf,0,sizeof(buf));
          memcpy(buf,p->pData,p->nSize);
          MessagBox(buf);
      }
     _QRCodeFree(pResult,nBarcodes);
}
…….
LIBRARY Linked:
    QRCodeDecodelib.lib
```

3.2. Dynamic link library

Structure of the QRCode decoding result:


```
typedef struct tagResult
{
    BYTE *pData;      //result of decoding result
    int nSize;        //data length
    int xPos;         //Left top position of barcode at image
    int yPos;         // Left top position of barcode at image
} Result;
```

3.2.1. Function or procedure

3.2.1.1. QRCodeDecodeImageFile

The QRCodeDecodeImageFile function read QRCode figure from image and decode it to text or binary data.

Result * __stdcall QRCodeDecodeImageFile(char *pImageFile,int *pCount);

Parameters:

pImageFile char *

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

pCount int

[out] the barcodes the function found and the number of the result returnd. The pointer need be allocate memory before calling the function.

Return values

Result *, the first result pointer. If having more barcodes, move the pointer to get the next barcode result.

NULL decode failure or no barcode found.

3.2.1.2. QRCodeDecode

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

Result * __stdcall QRCodeDecode (HBITMAP hImage,int *pCount);

Parameters:

hImage HBITMAP

[in] the bitmap handle containing QRCode figure,

pCount int

[out] the barcodes the function found and the number of the result returnd. The pointer need be allocate memory before calling the function.

Return values

Result *, the first result pointer. If having more barcodes, move the pointer to get the next barcode result.

NULL decode failure or no barcode found.

3.2.1.3. QRCodeFree

The QRCodeFree function free the result memory;

void _stdcall QRCodeFree(Result *r,int nCount);

Parameters:

r Result *

[in] the result pointer;

nCount int

[in] number of the result;

Return values

N/A

3.2.2. Samples

3.2.2.1 Example for Microsoft visual C++

```
#include “QRCodeDecodeDll.h”
……
Result *pResult;
int nBarcodes
char *pFile = “c:\\qrcode.jpg”;

pResult = QRCodeDecodeImageFile(pFile,&nBarcodes);
if(pResult)
{
    char buf[8192];
    Result *p = pResult;
    for (int I =0; I < nBarcodes; I++)
    {
```



```
            memset(buf,0,sizof(buf));
            memcpy(buf,p->pData,p->nSize);
            MessagBox(buf);
        }
      QRCodFree(pResult,nBarcodes);
}
…….
```

LIBRARY Linked:
    QRCodeDecodeDll.lib
DYNAMIC LIBRARY ar runtime:
    QRCodeDecodeDLL.dll

3.3. Java SDK for decoder

3.3.1. Package

```
package com.aipsys.client.j2se;
```

3.3.2. Result Class

```
public class Result {

   // Constructors

   public Result(byte[] byteArray, int _int, int _int2, int _int3, int _int4, int _int5, int
_int6) { }



   // Methods

   public byte[] getValue() { return null;}    //get the decoded result

   public int getX1() { return 0;}      //get the barcode position

   public int getY1() { return 0;}    //get the barcode position

   public int getX2() { return 0;}    //get the barcode position

   public int getY2() { return 0;}    //get the barcode position
```



```
    public int getX3() { return 0;}    //get the barcode position
    public int getY3() { return 0;}    //get the barcode position
    public String toString() { return null;}
}
```

3.3.3. Scanner Class

```
import java.awt.Image;

public class Scanner {

    // Constructors
    public Scanner() { }
   // Decode qrcode barcode from image file, now support JPG,PNG,GIF format
   public Result DecodeFromImageFile(String string) throws Exception { return null;}
   //Decode qrcode barcode from    Image, u can capture the video frame to image container then decode it.
   public Result DecodeFromImage(Image image) { return null;}
}
```

3.3.4. Samples

```
package com.aipsys.client.j2se;
public final class CommandLineRunner {
   private CommandLineRunner()
```



```
    {

    }

    public static void main(String[] args) throws Exception {


        com.aipsys.client.j2se.Scanner scanner = new Scanner();

        com.aipsys.client.j2se.Result r = scanner.DecodeFromImageFile("c:\\01.jpg");

        if (r !=null)

        {

            System.err.println("Trial version will replace first char of output with *\n");

            byte []tmp = r.getValue();

            String str = new String(tmp);



            System.err.println(str);

        }

        else

           System.err.println("no barcode found\n");

    }

},
```

3.5. SDK for Windows Phone7

3.5.1 Interface

```
public class Results
{
    public override System.String ToString();
```

```
    virtual public int X
    virtual public int Y
    virtual public sbyte[] Value
}

public class Scanner
{
        public bool    RegisterQRCodeReader(String strMail, String strRegCode);
                //where sterMail is the user mail domain and strRegCOde is the register
code.
        public Results DecodeFromImage(WriteableBitmap image);
                //where image is the rgb bitmap of barcode.
        public Results DecodeFromGrayArray(byte []image,int width,int height);
                        //where image is a gray pixel array row by row.
                        //other two parameters is width and height of the image
}
```

3.5.2 Sample

```
using aipsys.barcode;

.............

                Uri uri = new Uri("/ImageTest;component/2.JPG", UriKind.Relative);
                BitmapImage bitmapImage = new BitmapImage();
                bitmapImage.CreateOptions = BitmapCreateOptions.None;
                bitmapImage.UriSource = uri;

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

.................
```

3.5.3. Library

QRCodeDecodeWP7.dll

3.6. SDK for Android

3.6.1. Interface

```
public class QRCodeReader {

   static {

      System.loadLibrary("QRCodeReader");

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

   }

     public native static byte[] decode(byte []gray,,int width,int height);

     public native static boolean RegisterDecoder(String mailBox, String

regCode);

 }
```

3.6.2. Sample

```
#import com.aipsys.QRCodeReader;

QRCodeReader nativeLib;

…..

byte[] r = nativeLib. decode (gray, 480, 320);

….
```

3.6.3. Library

libQRCodeReader.so

3.7. SDK for iPhone platform

3.7.1. Interface

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

typedef struct tagResult
{
  char sBarcodeType[32];
  unsigned char *pData;
  int nSize;
  POINT ptsBarcode[4];
}Result;

Result *DecodeGrayImage(unsigned char *pGray,int width,int height, int *pCount);
void   FreeResult(Result *pResult, int nCount);
int RegisterDecoder(char *mailBox, char *regCode);
```

3.7.2. Samples

```
#include “QRcodeReader.h”
```



```
……

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

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

3.7.3. Library

libQRCodeReader.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

As to Price Information, please visit http://www.aipsys.com for details.

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 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	1 Developer	5 Developer	Unlimited Developer	Version

		Developer				
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	30022f38	300222339	300222340	300222341	1.2
ASP Component		300222372	300222373	300222374	300222375	1.2
Aztec Encode SDK						
Static Library		300222288	300222414	300222289	300222290	1.2
Dynamic Library	300222323	300222327	300222329	300222330	300222331	1.2
ActiveX	300222362	300222363	300222364	300222365	300222366	1.2
ASP Component		300222384	300222385	300222386	300222387	1.2

6. Support Information

Sales information

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

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

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

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

Barcode resources reference information

Introduction to common barcode types

RSS barcodes renamed GS1-DataBar

Recommended sizes for barcodes

Barcode specifications & Standards

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

7. Product Information Link

. QRCode encoder SDK

. PDF417 encoder SDK

. DataMatrix encoder SDK

. Aztec encoder SDK

. Linear 1D barcode encoder SDK