Data Encoding

Each encoding mode is designed to create the shortest possible string of bits for the characters that are used in that mode. Each mode uses a different method for converting the input text into a string of bits. This page explains the entire data encoding step.

Step 1: Choose the Error Correction Level

Before encoding the data, select an error correction level. As mentioned in the introduction, QR codes use Reed-Solomon error correction (http://en.wikipedia.org/wiki/Reed–Solomon_error_correction). This process creates error correction codewords (bytes) based on the encoded data. A QR code reader can use these error correction bytes to determine if it did not read the data correctly, and the error correction codewords can be used to correct those errors. There are four levels of error correction: L, M, Q, H. The following table lists the levels and their error correction capabilities.

Error Correction Level	**Error Correction Capability**
L	Recovers 7% of data
M	Recovers 15% of data
Q	Recovers 25% of data
H	Recovers 30% of data

Be aware higher levels of error correction require more bytes, so the higher the error correction level, the larger the QR code will have to be.

Step 2: Determine the Smallest Version for the Data

The different sizes of QR codes are called **versions**. There are forty versions available. The smallest version is version 1, and is 21 pixels by 21 pixels in size. Version 2 is 25 pixels by 25 pixels. The largest version is version 40, and is 177 by 177 pixels in size. Each version is 4 pixels larger than the previous version.

Each version has a maximum capacity, depending on the mode in use. In addition, the error correction level restricts the capacity further. The character capacities table (character-capacities) lists the capacities of all QR versions for a given encoding mode and error correction level.

How to Determine the Smallest Version

At this point, count the number of characters to be encoded, and determine which is the smallest version that can contain that number of characters for the encoding mode and desired error correction level.

For example, the phrase **HELLO WORLD** has 11 characters. If encoding it with level Q error correction, the character capacities (character-capacities) table says that a version 1 code using level Q error correction can contain 16 characters in alphanumeric mode, so version 1 is the smallest version that can contain this number of characters. If the phrase were longer than 16 characters, such as HELLO THERE WORLD (which is 17 characters) version 2 would be the smallest version.

Upper Limits

The highest capacity QR code is 40-L (version 40, error correction level L). Below is a table that lists the capacity of a 40-L QR code for the four encoding modes. This is the maximum possible number of characters that a single QR code can contain. Versions 40-M, 40-Q, and 40-H have lower capacity because they require more space for more error correction codewords. For a table of the capacities of all versions, please see the character capacities table (character-capacities).

Encoding Mode	Maximum number of characters a 40-L code can contain in that mode
Numeric	7089 characters
Alphanumeric	4296 characters
Byte	2953 characters
Kanji	1817 characters

Step 3: Add the Mode Indicator

Each encoding mode has a four-bit mode indicator that identifies it. The encoded data must start with the appropriate mode indicator that specifies the mode being used for the bits that come after it. The following table lists the mode indicators for each mode.

For example, if encoding **HELLO WORLD** in alphanumeric mode, the mode indicator is 0010.

Mode Name	Mode Indicator
Numeric Mode	0001
Alphanumeric Mode	0010
Byte Mode	0100
Kanji Mode	1000
ECI Mode	0111

Step 4: Add the Character Count Indicator

The character count indicator is a string of bits that represents the number of characters that are being encoded. The character count indicator must be placed after the mode indicator. Furthermore, the character count indicator must be a certain number of bits long, depending on the QR version.

Count the number of characters in the original input text, then convert that number into binary. The length of the character count indicator depends on the encoding mode and the QR code version that will be in use. To make the binary string the appropriate length, pad it on the left with 0s.

The following lists contain the sizes of the character count indicators for each mode and version. For example, if encoding **HELLO WORLD** in a version 1 QR code in alphanumeric mode, the character count indicator must be 9 bits long. The character count of **HELLO WORLD** is 11. In binary, 11 is 1011. Pad it on the left to make it 9 bits long: 000001011. Put this after the mode indicator from step 3 to get the following bit string: 0010 000001011

Versions 1 through 9

- Numeric mode: 10 bits
- Alphanumeric mode: 9 bits
- Byte mode: 8 bits
- Japanese mode: 8 bits

Versions 10 through 26

- Numeric mode: 12 bits
- Alphanumeric mode: 11 bits
- Byte mode: 16
- Japanese mode: 10 bits

Versions 27 through 40

- Numeric mode: 14 bits
- Alphanumeric mode: 13 bits
- Byte mode: 16 bits
- Japanese mode: 12 bits

Step 3: Encode Using the Selected Mode

The previous page, data analysis (data-analysis), explains how to select the appropriate encoding mode for a given string. The process for each encoding mode is explained on its own page. Click a link below to read about the encoding process for each mode.

- Numeric Mode Encoding (numeric-mode-encoding)
- Alphanumeric Mode Encoding (alphanumeric-mode-encoding)
- Byte Mode Encoding (byte-mode-encoding)
- Kanji Mode Encoding (kanji-mode-encoding)

HELLO WORLD is encoded on the alphanumeric mode encoding (alphanumeric-mode-encoding) page. Continuing the HELLO WORLD example, the bit string so far is:

Mode Indicator	**Character Count Indicator**	**Encoded Data**
0010	000001011	01100001011 01111000110 10001011100 10110111000

		10011010100 001101

Step 4: Break Up into 8-bit Codewords and Add Pad Bytes if Necessary

After obtaining a string of bits that consists of the mode indicator, the character count indicator, and the data bits as described in steps 1 through 3 on this page, it may be necessary to add 0s and pad bytes, because the QR code specification requires that the bit string must completely fill the total capacity of the QR code. The following sections explain the process of adding 0s and pad bytes to the bit string.

Determine the Required Number of Bits for this QR Code

To determine how many data bits are required for a particular QR code, refer to the error correction table (error-correction-table). Find the version and error correction level that is in use for the QR code being encoded, and find the number in the column that is labeled "Total Number of Data Codewords for this Version and EC Level". Multiply this number by 8 to obtain the total number of data bits required for this version and error correction level.

For example, according to the table, a version 1-Q code has 13 total data codewords. Therefore, the total number of bits required for this QR code is 13 * 8, or 104 bits.

Add a Terminator of 0s if Necessary

If the bit string is shorter than the total number of required bits, a terminator of up to four 0s must be added to the right side of the string. If the bit string is more than four bits shorter than the required number of bits, add four 0s to the end. If the bit string is fewer than four bits shorter, add only the number of 0s that are needed to reach the required number of bits.

For example, if encoding **HELLO WORLD** in a version 1-Q QR code, the total number of required bits as mentioned in the previous section is 104 bits. The data bit string that is shown in step 3 on this page is 74 bits long. The terminator must only be at most 4 bits long, so add four 0s to the right of the string. The resulting string is still too short to fill the 104 bit capacity, but the QR code specification requires that the terminator be at most four 0s in length. If the string had been 102 bits instead, the terminator would only be 2 bits in length.

Here is the example **HELLO WORLD** string with terminator added:

Mode Indicator	**Character Count Indicator**	**Encoded Data**	**Terminator**
0010	000001011	01100001011 01111000110 10001011100 10110111000 10011010100 001101	0000

Add More 0s to Make the Length a Multiple of 8

After adding the terminator, if the number of bits in the string is not a multiple of 8, first pad the string on the right with 0s to make the string's length a multiple of 8.

For example, after adding the terminator to the HELLO WORLD string, the length became 78 bits long. This is not a multiple of 8. The bit string is shown here broken up into 8-bit binary bytes:
00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 010000

There are six bits at the end. Add two 0s to make it an 8-bit binary byte:

00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000**00**

Add Pad Bytes if the String is Still too Short

If the string is still not long enough to fill the maximum capacity, add the following bytes to the end of the string, repeating until the string has reached the maximum length:
11101100 00010001

These bytes are equivalent to 236 and 17, respectively. They are specifically required by the QR code specification to be added if the bit string is too short at this stage.

For example, the HELLO WORLD string above is 80 bits long. The required capacity for a 1-Q code, as stated earlier on the page, is 104 bits. The number of bits that must be added to fill the remaining capacity is 104 - 80, or 24. Divide this by 8: 24 /8 = 3. Therefore, three pad bytes must be added to the end of the data string. This is shown below:
00100000 01011011 00001011 01111000 11010001 01110010 11011100 01001101 01000011 01000000 **11101100 00010001 11101100**

Next: Error Correction Coding

Now that the raw data bits have been obtained, the next step is to generate error correction codewords (error-correction-coding) for the data.

Page last updated 2017-03-07T15:49:56-06:00