The Hidden Language of
Computer Hardware and Software

CODE

SECOND EDITION

CHARLES PETZOLD

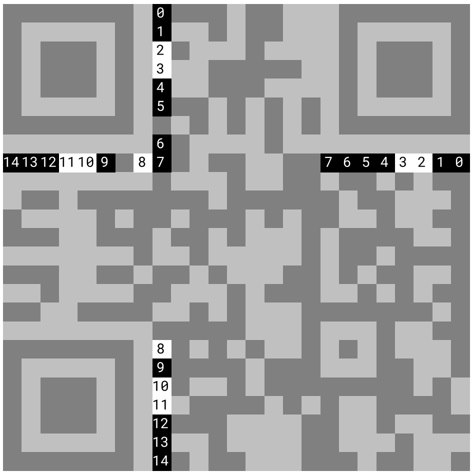
The Hidden Language of
Computer Hardware and Software

C O D E

1000011 1001111 1000100 1000101

SECOND EDITION

CHARLES PETZOLD



Bits are sometimes labeled with numbers like this to indicate how they constitute a longer value. The bit labeled 0 is the least significant bit and appears at the far right of the number. The bit labeled 14 is the most significant bit and appears at the left. If white cells are 0 bits and black cells are 1 bits, here is that complete 15-bit number:

111001011110011

Why is bit 0 the least significant bit? Because it occupies the position in the full number corresponding to 2 to the zero power. (See the top of page 109 if you need a reminder of how bits compose a number.)

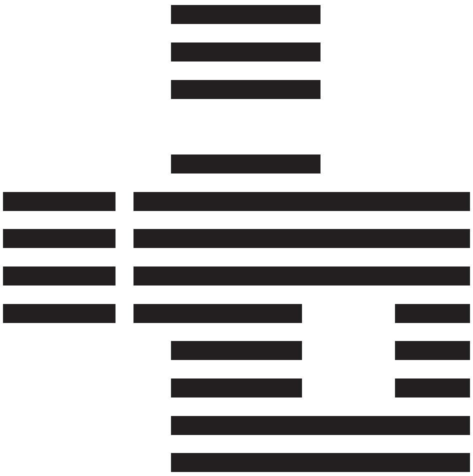
The actual numeric value of this 15-bit number is not important, because it consolidates three pieces of information. The two most significant bits indicate one of four error-correction levels. The ten least significant bits specify a 10-bit BCH code used for error correction. (BCH stands for the inventors of this type of code: Bose, Chaudhuri, and Hocquenghem. But I promised I wouldn't discuss the QR code error correction!)

In between the 2-bit error-correction level and the 10-bit BCH code are three bits that are *not* used for error correction. I've highlighted those three bits in bold:

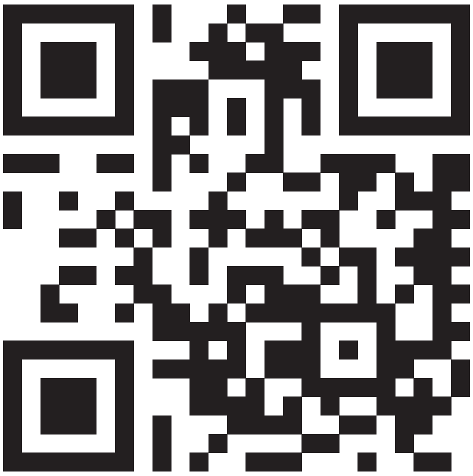
11**100**1011110011

It turns out that QR code readers work best when there are approximately an equal number of black and white squares. With some encoded information, this will not be the case. The program that creates the QR code is responsible for selecting a *mask pattern* that evens out the number of black and white squares. This mask pattern is applied to the QR code to flip selected cells from white to black, or black to white, and hence the bits that they represent from 0 to 1 and from 1 to 0.

The documentation of the QR code defines eight different mask patterns that can be specified by the eight 3-bit sequences 000, 001, 010, 011, 100, 101, 110, and 111. The value in the QR code that we're examining is 100, and that corresponds to a mask pattern consisting of a series of horizontal lines alternating every other row:

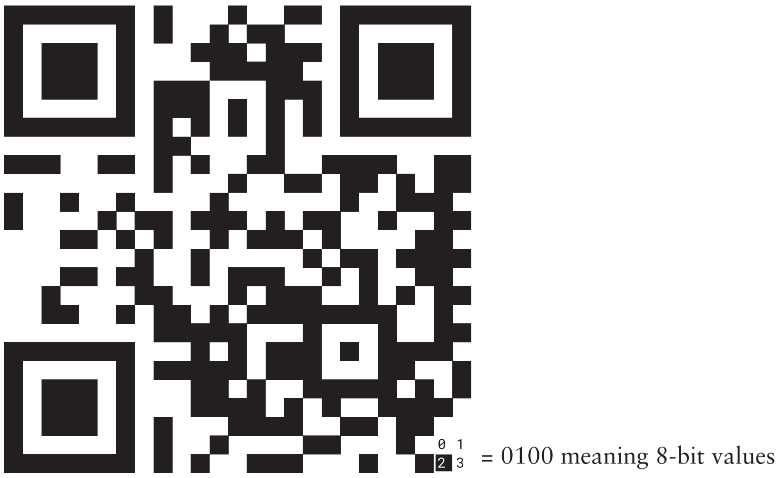


Every cell in the original QR code that corresponds to a white area in this mask remains unchanged. Every cell that corresponds to a black area must be flipped from white to black, or from black to white. Notice that the mask avoids altering the fixed areas and the QR information area. Here's what happens when this mask is applied to the original QR code:



The mask doesn't change the fixed and information areas. Otherwise, if you compare this image with the original QR code, you'll see that the top row is reversed in color, the second row is the same, the third row is reversed, and so on.

Now we're ready to start digging into the actual data. Begin with the four bits in the lower-right corner. In the following image, those cells are numbered 0 through 3, where 3 is the most significant bit and 0 is the least significant bit:

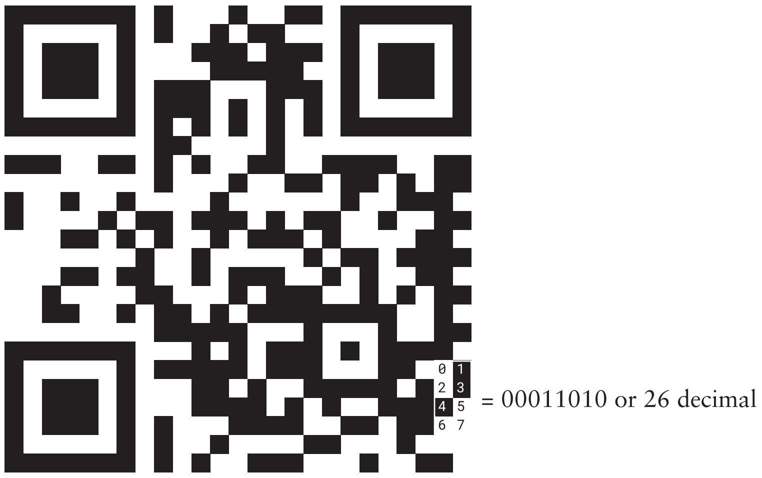


These four bits are known as the *data type* indicator, and they indicate what kind of data is encoded in the QR code. Here are a few of the possible values:

Date-Type Indicator	**Meaning**
0001	Numbers only
0010	Uppercase letters and numbers
0100	Text encoded as 8-bit values
1000	Japanese kanji

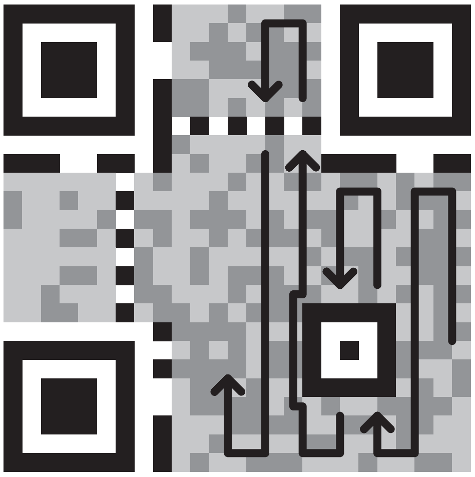
The value for this QR code is 0100, meaning that the data consists of 8-bit values that encode text.

The next item is stored in the eight cells above the data type indicator. These eight bits are numbered 0 through 7 in this illustration:

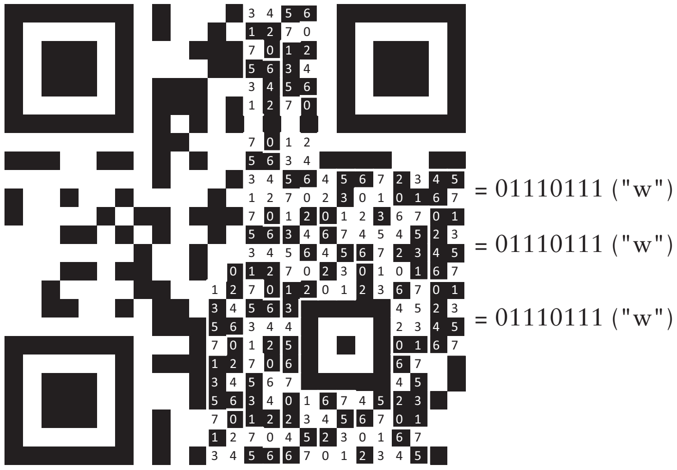


This value is 00011010, which is 26 in decimal. That's the number of characters encoded in the QR code.

The order of these characters is systematic but weird. The characters begin right above the character count. Each character usually—though not always—occupies an area that is two cells wide and four cells tall, and the characters wind through the grid like this:



Not all characters occupy areas that are two cells wide and four cells tall. Fortunately, the official QR specification is quite precise about how the bits are oriented when the area is not rectangular. In this next image, the cells for each of the 26 characters are outlined in red, and the cells are numbered 0 through 7, where 0 denotes the least significant bit and 7 the most significant bit:



The QR specification indicates that text is encoded in the QR code using 8-bit values defined in a standard known as ISO/IEC 8859. That's a fancy term for a variation of the American Standard Code for Information Interchange (ASCII), which I'll be discussing in more detail in Chapter 13.

The first character is 01110111, which is the ASCII code for *w*. The next character up is the same. The next character extends to the left, but it is also another *w*. Now proceed down the next two pairs of columns. The next character is 00101110, which is the period, then 01000011, the uppercase C followed by 01101111, *o*. The next character straddles the next pair of rows. It's 01100100: *d*. The next character begins below the alignment pattern and continues above it. The ASCII code is 01100101, which is *e*. Continue in this way to spell out www.CodeHiddenLanguage.com.

That's it. Most of what's left in the QR code is devoted to error correction.

Codes such as the UPC and QR certainly look forbidding at first glance, and people might be forgiven for assuming that they encode secret (and perhaps devious) information. But in order for these codes to be widely used, they must be well documented and publicly available. The more that they're used, the more potentially valuable they become as another extension of our vast array of communication media.

Bits are everywhere, but toward the end of my discussion of the QR code, I referred to "8-bit values." There's a special word for 8-bit values. You may have heard of it.

Chapter Twelve

Bytes and Hexadecimal

Individual bits can make big statements: yes or no, true or false, pass or fail. But most commonly, multiple bits are grouped together to represent numbers and, from there, all kinds of data, including text, sound, music, pictures, and movies. A circuit that adds two bits together is interesting, but a circuit that adds multiple bits is on its way to becoming part of an actual computer.

For convenience in moving and manipulating bits, computer systems often group a certain number of bits into a quantity called a *word*. The length or size of this word—meaning the number of bits that compose the word—becomes crucial to the architecture of the computer because all the computer's data moves in groups of either one word or multiple words.

Some early computer systems used word lengths that were multiples of 6 bits, such as 12, 18, or 24 bits. These word lengths have a very special appeal for the simple reason that the values are easily represented with octal numbers. As you'll recall, the octal digits are 0, 1, 2, 3, 4, 5, 6, and 7, which correspond to 3-bit values, as shown in this table:

Binary	**Octal**
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7