

READING QR CODES WITHOUT A COMPUTER!

Made in love by Piko and blinry for 37C3

Did you ever wonder how QR codes work? You've come to the right place! This is an interactive explanation that we've written for a workshop at [37C3](#), but you can also use it on your own. You will learn:

- The anatomy of QR codes
- How to decode QR codes by hand (using our [cheat sheet](#))

TABLE OF CONTENTS

1. [Anatomy of a QR code](#)
2. [Mask pattern](#)
3. [Encoding mode](#)
4. [Reading order](#)
5. [Decoding the length](#)
6. [Decoding the content](#)
7. [Congratulations!](#)

PICK YOUR QR CODE

Before we start, we need a QR code to work with! You have three options here. First, you can enter any text, and it will update the QR code that's explained below:

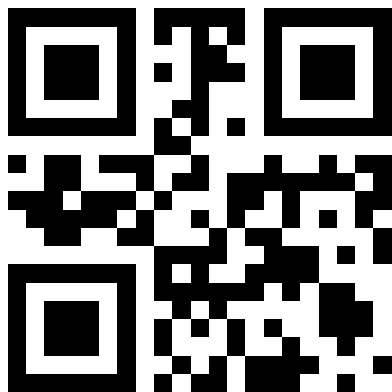
Second, you can scan any existing QR code using your camera, to learn how to read it:

And third, you can practice on random English words! The word will not be shown, so

that you can use it for training:

ANATOMY OF A QR CODE

Here's the QR code you picked:



The code is has a size of 21×21 *modules* (which is the QR code name for pixels).

QR codes come in different sizes, which are called *versions*. The smallest version is 21×21 modules, and for every version after that, the size increases by 4. This means that our QR code has a version of 1!

Here are the different parts:

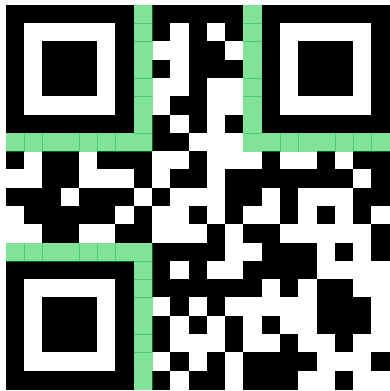
FINDER PATTERNS

They help scanners identify QR codes in the first place.



SEPARATORS

Empty lines around the finder patterns, it's important that you skip them when reading the content.



ALIGNMENT PATTERNS

This QR code has version 1, so it doesn't have any alignment patterns (which are additional patterns that help scanners read the QR code correctly).

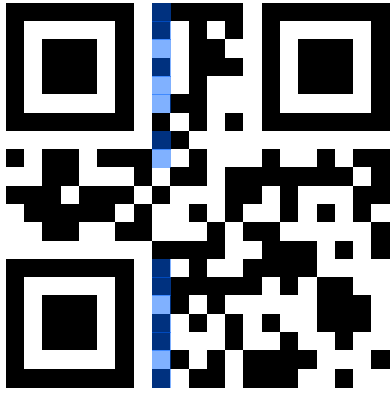
TIMING PATTERNS

Two always alternating lines of black/white modules – another way to help scanners read the QR code correctly.



FORMAT INFORMATION

This determines the *mask pattern* and the *error correction level*. There's a vertical version:



And a horizontal one. Note that the timing pattern interrupts both lines!



HERE'S ALL AREAS TOGETHER

Everything that's not highlighted is the actual content!



OTHER AREAS

There is one other area that is not shown here: The *quiet zone*, which is a white border around the QR code. It helps scanners identify where the QR code starts and ends.

MASK PATTERN

To avoid big blobs of black or white modules, QR codes use a *mask*. There are eight different options.

Here's where the mask pattern is specified:

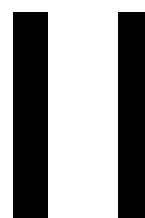


Black modules are a 1, and white boxes are a 0. In this case, the mask pattern consists of the bits which are the binary numbers **110**. They correspond to the decimal number 6.

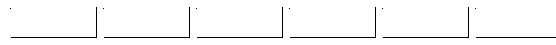
There is a predefined pattern for each type. To better remember those patterns, you can use the following story.

MASK PATTERN STORY

See, you have broken the law. I'm really sorry, but I see black for your future, you are going behind **bars**:



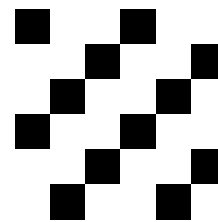
But then, things happen, a story unfolds. It follows the black pixels moving from left to right, like this:



So, what happens? First , you get the cliché **striped prison clothing**.

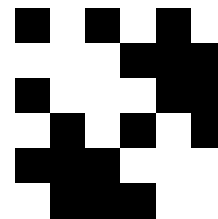


But you don't lose hope yet , and a **rainbow** helps you keep your spirit up.

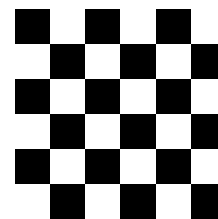


Half time into your time in prison (the symmetric codes: and) , you start to have hobbies:

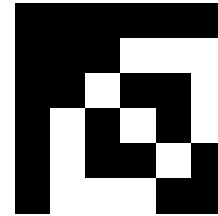
When you are alone , you paint, for example [M.C. Escher's Lizards](#).



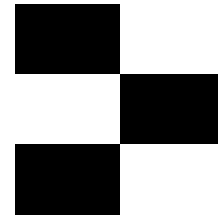
Or you watch with your two eyes the games of **chess** the other inmates are playing.



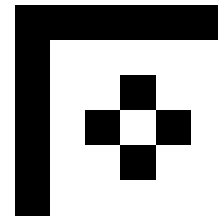
But the time drags on _____ and you start to make break-out plans: You will cut your way out with an **axe**.



And finally _____ you manage to get out! This is the finishing line for your time in prison, so you get a **chequered flag**.

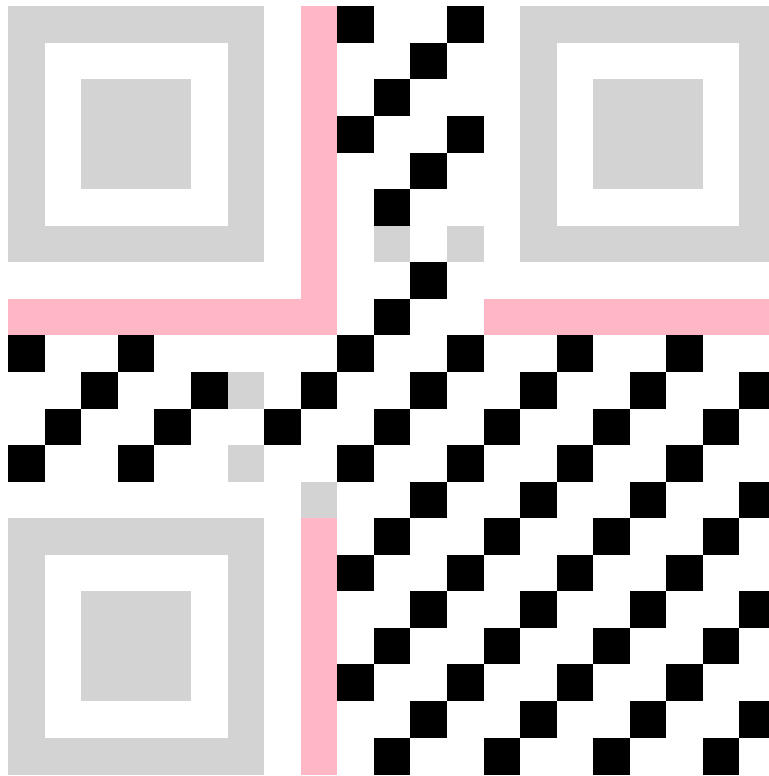


And now, everything is bright _____ and happy, like a **flowered wall paper**!



In our case, the pattern _____ is the one that looks like **part of a black-and-white rainbow**!

If you repeat this pattern over the data parts of the QR code, you get this:



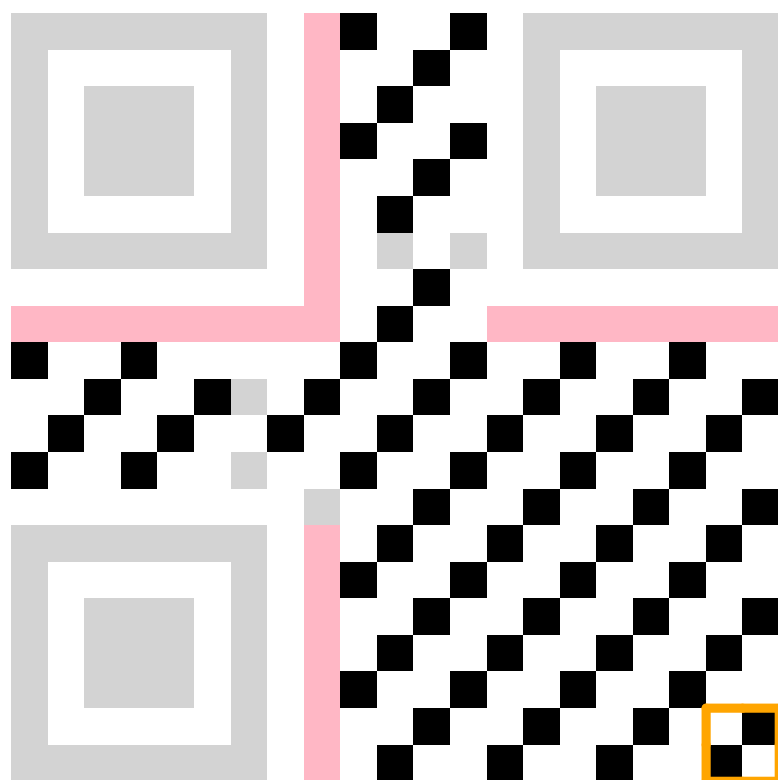
For each black module in the mask, you flip the module in the QR code! The next steps will show you how. We prepared a [DIY zine](#) that contains pre-made cheat sheets for all mask types! You can fold it into a little booklet, [like this](#)!

ENCODING MODE

Let's start decoding the content! First, we need to know the encoding of the content. That information is always given in the bottom-right corner.



But – remember! You first have to apply the mask! Let's look at our cheat sheet to see how.



Let's XOR the two boxes together:

$$\begin{array}{c} \blacksquare \\ \blacksquare \end{array} \text{ XOR } \begin{array}{c} \blacksquare \\ \blacksquare \end{array} = \begin{array}{c} \blacksquare \\ \blacksquare \end{array}$$

Again, for each pattern, there's a different encoding mode.



Numeric



Alphanumeric



Byte



ECI (Extended Channel Interpretation)

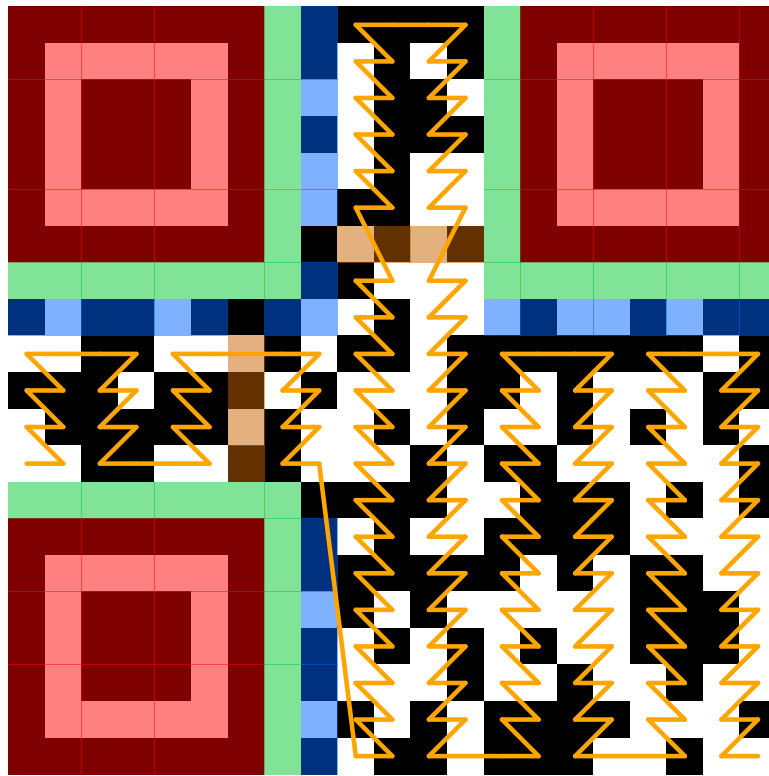


Kanji

For our code, the encoding mode is **Byte** (4), so we can proceed!

READING ORDER

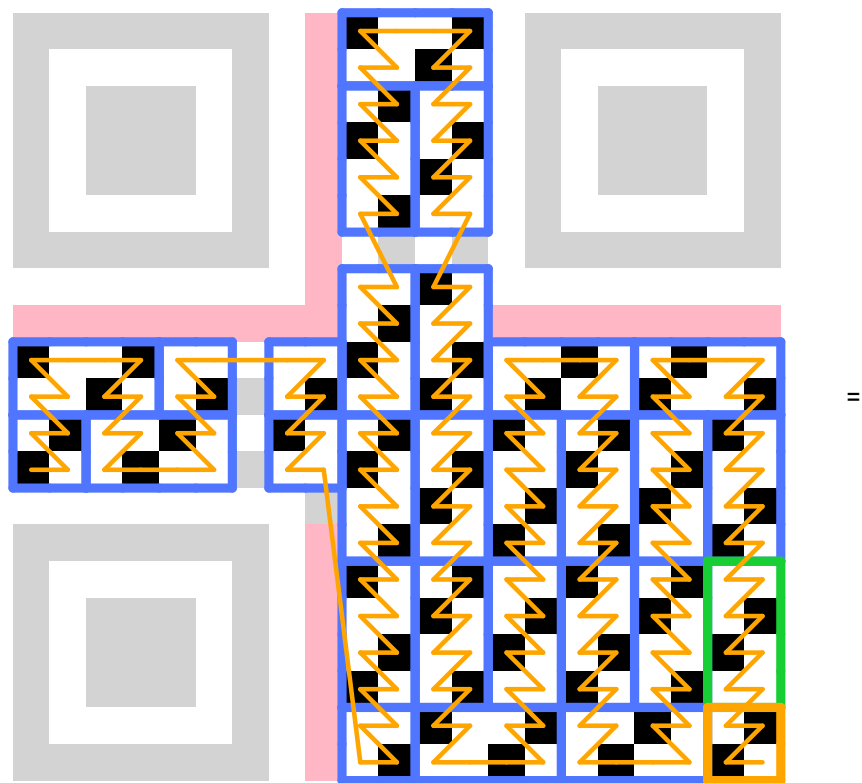
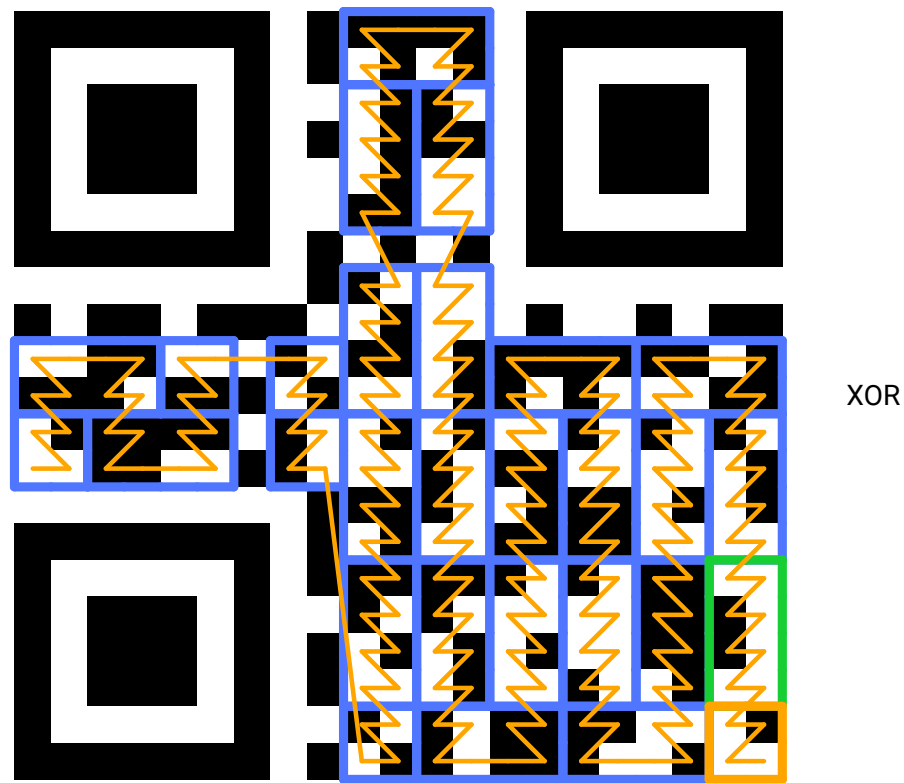
The content starts in the bottom-right corner. You go up, zig-zagging between two columns, and then zig-zag down the next to to the left. You *always* skip all special areas!

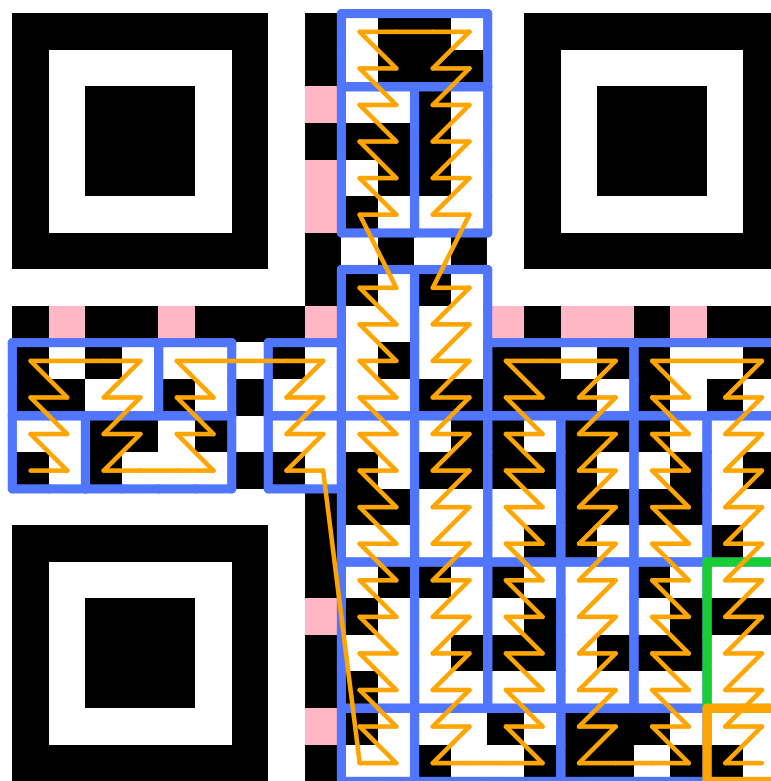


BYTES

The first **four bits** are the encoding mode. For the ASCII encoding mode, the next **eight bits** specify the content length. After that, every eight bits is a content byte.

And again, we have to apply the mask:





This is the (unmasked) length information:



These bits are in decimal: **12**

This value is the total length of the content in bytes!

DECODING THE CONTENT

And finally, we can proceed decoding the content of the QR code!

This is the first (unmasked) byte:

These are the bits: **01001000**



These bits are in hexadecimal: **0x48**

And this is the corresponding ASCII character: **H**

This is the second (unmasked) byte:

These are the bits: **01100101**



These bits are in hexadecimal: **0x65**

And this is the corresponding ASCII character: **e**

...and so on! The remaining bytes yourself are for you to practice on. You can stop when you've read the number of characters indicated by the length field previously.

ASCII TABLE

You can use this table to translate the hexadecimal numbers into ASCII characters! You can also try to memorize parts of this table, if you're that kind of person: The lowercase letters start with an "a" at 0x61, and end at "z" with 0x7A. The ones in between you can count.

	...0	...1	...2	...3	...4	...5	...6	...7	...8	...9	...A	...B	...C	...D	...E	...F
0x2...	_	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
0x3...	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4...	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5...	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6...	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x7...	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

ERROR CORRECTION

So is the rest all content? Not quite! There is also error correction, which is used to

make sure that the QR code can still be read even if some parts are damaged, blurry, or missing. The error correction is generated by some fancy math, and we don't care about it here for the purpose of reading it by hand.

CONGRATULATIONS!

This should be everything you need to decode simple QR codes by hand. You can now either press the "Random code" button at the top to practice on short English words, or go find a QR code in the wild, and scan it using the "Scan code" button!

FOUND A BUG? FEATURE REQUEST?

You can find the source code on [Codeberg](#). We'd be especially happy about PRs that explain how UTF-8 encoding modes work!