Bringing together the digital and physical worlds

Barcodes with iOS 7

Oliver Drobnik

MEAP

MANNING

MEAP Edition
Manning Early Access Program
Barcodes with iOS 7
Version 1

For more information on this and other Manning titles go to
www.manning.com

Welcome

In June 2014 it will be exactly 40 years since the first commercial barcode was scanned. That is to say that barcodes are a ubiquitous and incredibly mature technology. The UPC you find on all products sold in your supermarket was just the beginning. Just look at any Apple product box you have brought home lately. You'll find several additional barcodes on the sticker. This book will open your eyes to the sheer number of different usage scenarios involving barcodes.

With the release of iOS 7, Apple put a barcode scanner into all iPhones. Together with always-on mobile Internet and built-in device sensors, this enables a new breed of product-centric apps which have not been feasible before. This means exciting new apps that you and many other readers of this book will be creating, bringing together the digital and physical worlds.

I am writing *Barcodes with iOS 7* for intermediate level iOS developers who know their way around Xcode and have made a few apps already. If you are an absolute beginner then I recommend you look at something more basic before reading this book. This book, unlike most of the Manning catalog, is written in more of a tutorial style as readers of my blog Cocoanetics.com have come to appreciate.

The first chapter introduces you to the various kinds of barcodes that iOS 7 supports natively. It features a condensed history of the UPC with information that is hard to find anywhere else in one place. You will become a barcode guru who can hold his own in any conversation about barcodes, their promises and their limitations. This understanding will be the main eye opener in your daily life. Barcodes will no longer be visual noise to you but represent app opportunities everywhere you spot them.

Chapters 2 and 3 give you a solid introduction to AV Foundation media capture and how to use the new metadata detectors for scanning barcodes.

Looking ahead, Chapter 4 deals with Apple's main reason for pushing forward with barcodes support in iOS: Passbook. Chapter 5

looks at how you can generate barcodes for display on devices and how to print them to physical media via AirPrint. Chapter 6 looks at networking to retrieve metadata for scanned barcodes, in particular NSURLSession for background fetching. Chapter 7 then rounds off the book by adding contextual information about the user scanning barcodes to the mix, with Core Location and iBeacon.

By purchasing the MEAP for *Barcodes with iOS* 7 you are showing your support for my work on this book, and I thank you very much for your trust and interest! I encourage you to connect with me and other readers on the Author Online forum. Your feedback will inform the road ahead for this book, and I would love to hear your real life barcode-related stories.

This book will change your life as an iOS developer. I know, because writing it changed mine.

—Oliver Drobnik

brief contents

1 Barcodes, iOS 7 and you

This chapter covers:

- Why the nexus of barcodes and mobile technologies is creating new exciting opportunities for app makers
- The barcode symbologies in iOS 7 you should know about
- The distinctions between 1D and 2D barcodes
- A brief history of the UPC/GTIN, the mother of modern barcodes

If you wanted to add barcode scanning to your apps in the past, you had to either fight your way through open source projects or license a commercial barcode scanning library. None of those projects were written in Objective-C, documentation would be lacking and commercial solutions would require payment of license fees for each downloaded copy of your app. All of these made barcode scanning impractical for all but the most skilled iOS developers, or too expensive to make economic sense for free or low-income apps.

When Apple added Passbook to iOS 6 they needed the ability to *display barcodes* on Passbook passes. With iOS 7, Apple made these APIs public and added the ability to *scan barcodes*. This allowed them to add such functionality to several of their first-party apps:

- The **Passbook app** lets you add new passes to your device by scanning special QR codes.
- The **iTunes app** has the ability to redeem a iTunes credits via scanning the voucher.
- The **Apple Store app** has an in-store UI that lets you scan the barcodes of accessories for unassisted checkout.

Figure 1.1 Barcode scanning in Apple Store app

In June 2014 the mother of all barcodes, the UPC, will be celebrating its 40th annivesary. This makes it an incredibly well understood and ubiquitous technology. Throughout these decades different kinds of usage scenarios prompted the development of a variety of barcode symbologies which were more or less all informed by the UPC. Apple selected the most prevalent and useful kinds of barcodes to support in iOS 7. To grasp the full potential of these differing symbologies - as they are relevant for you as iOS app developer - you will be learning for which purposes they are used best in.

This introductory chapter will give you a solid understanding of barcode technology. Seeing how the many symbologies relate to each other should eleviate any anxiety you might feel right now. You'll no longer shiver in fear from not knowing the difference between UPC, EAN, GTIN, Code39, Code93 and Code128. Especially when dealing with the GTIN family of barcodes the plethora of abbreviations is daunting. A brief history of the UPC/GTIN will let several lights go on for you. Not only has the long history of it been quite amusing at times, it will also aide greatly in your appreciation of the current state of the GTIN. You will become a *barcode guru* who can hold his own in any discussion about barcodes.

Two chapters give an in-depth tutorial about he parts of **AVFoundation** as they are required for scanning bar codes. One will be a more general tutorial about the intricacies of media capture via the iPhone cameras. The other will build on this

functionality and add metadata scanning. The goal is that you end up with reusable classes for barcode scanning which you truly understand. This you can add - with little modification - to any app project to support barcode scanning.

Apple's poster child for the use of barcodes in iOS is the **Passbook** system and app. Passbook passes are made possible by 2D barcode support in iOS, barcodes are the means of data transfer between a pass that is presented for scanning and the app that validates said passes. You will learn how to create valid signed Passbook passes. We will teach you a simple method for validating scanned passes without the need for a server. You will see that Passbook is a great idea and useful not only for large corporations. You might have several great potential uses right in your backyard!

The other side of the barcode coin is the ability to **generate barcodes**. iOS7 has the built-in ability to generate 2D barcodes, as this is used by Passbook to display them on passes. Readers of this book get a free license to our commercial **BarCodeKit** component. This lets you generate a wide variety of 1D barcodes for display or printing. In addition to being able to "consume" barcodes by scanning them you gain the ability to also "create" barcodes to contribute to the barcode ecosystem.

Other mobile technologies also lend themselves to be used synergistically with barcodes, some of which have been updated or greatly enhanced in iOS7. Having always-on Internet connectvitiy in iPhones allows you to look up metadata for physical items that have a barcode. We will specifically look at **modern iOS networking** for this purpose and dive into the new methods for smart background fetching of data.

The final piece in the barcode technology puzzle is about **user context**. iPhones are people's permanent companions and via their sensors they posess contextual information about what situation the user is in. These "smarts" let you tailor your app user experience based on whether a user is at home, in the vicinity of a brick-and-mortar store or if he's inside it next to a specific shelf. Barcodes provide the most fine-grained context as they are either about a product model (scan UPC/GTIN) or a specific instance of a product (scan serial number). iBeacons and GPS location form two additional layers of user context to personalize the user experience.

Barcodes are the identity technology of choice for objects in the physical world. Mobile technologies - as they are found in iOS devices - compliment it beautifully in the digital world. This is why this book has the tag line "bringing together the

physical and digital worlds".

1.1 The evolution of barcodes

The first barcode in wide use was the Universal Product Code (UPC) combining the semantic meaning of a number of 12 digits with a machine-readable scheme for representing these number as bars. It was designed only for automated handling of physical products and therefore limited to represent numerical product codes. Section 1.3 will walk you through the history of the UPC and how it became the GTIN as it is referred to nowadays by people "in the know".

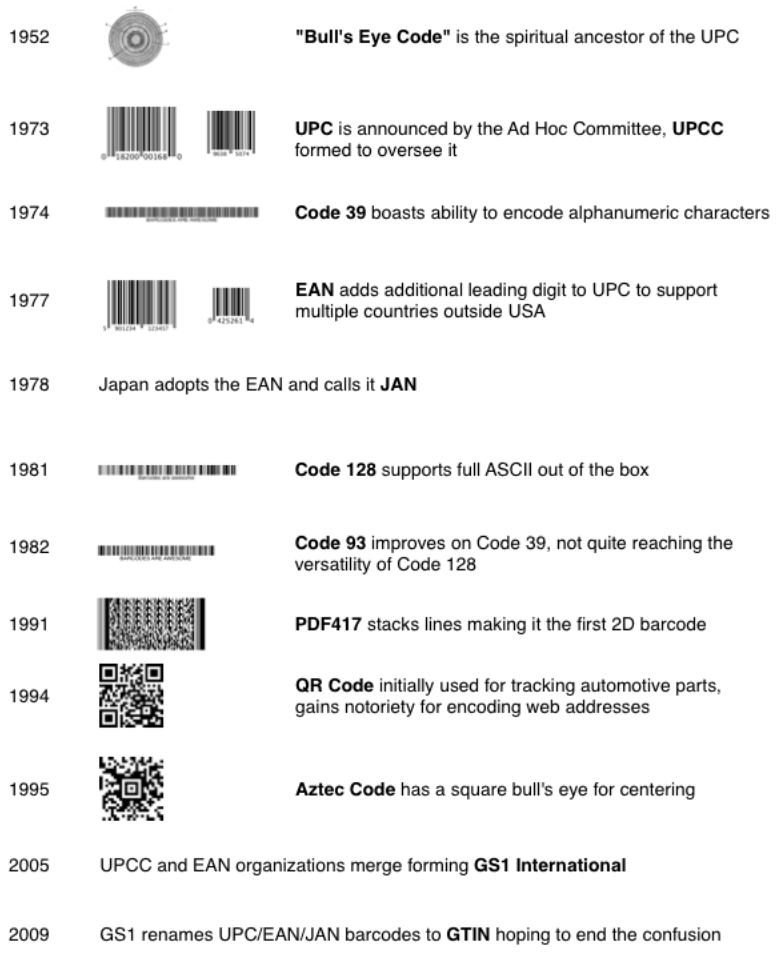


Figure 1.2 Timeline of Barcodes

Having taken the hurdle of having a machine recognize visual markings with a laser beam a plethora of additional kinds of bar codes started to appear, all with more or less specific fields of application.

The post office found that adding markings to mailed items would allow them to automatically sort the items. Luggage for airline travel was kept track of with numeric codes. Many other industries had their own standards that worked the best

for them.

A combination of several bars that make up an individual character/digit is often called a **symbol**. The set of symbols available for a specific barcode standard is referred to as **symbology**.

1.1.1 One dimension: laser

All these different symbologies had in common that they could be read with a **laser beam**. Think of the laser beam as cutting out a single horizontal slice of the vertical code bars. Then integrated circuitry compares the widths of dark areas against light ones and thus can decode it.

The line of the laser beam is also the reason why these kinds of barcodes are referred to as being **one-dimensional (1D)**.

If you have more complex encoding schemes you can also represent more different characters, like letters and special characters. Some 1D barcode types were created to also be able to represent short texts.

The major advantage of 1D barcodes has for the longest time been that they can be decoded extremely reliably even when the items tagged with such codes are moving at high speed. Some schemes employ checksums to recognize if something was misread to even increase this reliability.

The second advantage is one of cost. Because of the technology being around for 40 years now the components necessary (laser diode, decoding electronics) has become cheep and reliable. This is ideally suited for high volume deployment as well as use in environments requiring to scan a great deal of codes in short succession.

1.1.2 Two dimensions: CCD

The **Charge-coupled Device (CCD)** got invented at **AT&T Bell Labs** by Williard Boyle and George E. Smith in 1969. This is the chip at the heart of any kind of digital camera. Curiously the technology for CCDs was invented round about the time of the first 1D barcode was introduced, but it took decades to achieve a level of fidelity that would allow CCDs to compete with the technically much simpler predecessor.

A CCD is essentially a matrix of pixels that is able to read different kinds of binary values for each pixel depending on the light intensity shining on it. Because of this it can do the same thing as a laser beam if you have sufficient pixels (aka resolution). Since the CCD pixels are laid out in two dimensions this is also able to recognize a new kind of barcode, the **two-dimensional barcode (2D)**

Initially CCDs did have neither the necessary resolution nor did CPUs have the decoding power required for barcode recognition. A CPU essentially needs to look at each individual frame of video coming from the CCD and look for patterns that constitute a code. Significant advances in electronics and Computer Vision were necessary to be able to do that.

Like all computer technology we witnessed Moore's Law working its magic on CCDs as well to eliminate these hurdles over time. Modern smartphones have resolutions measured in Megapixel which are more than sufficient for scanning 1D and 2D barcodes.

If there is any reason left why laser-based scanners are still more widely used at supermarket check-stands than CCD-based scanners then I can only think of two reasons: CCD hardware is more complex and expensive than laser scanners. And CCDs had a limit in the past how many frames per second of video they are able to capture. Move the scanned code to quickly then all the decoder sees are blurred images and will be unable to recognize any barcodes.

1.1.3 Versatility winning

Imagine setting up a competition for accuracy and speed between a laser- and CCD-based scanner, of course the latest and greatest models. Our test scenario is a check-stand at a supermarket. The contestants are professional cashiers with years of experience scanning products. Whoever scans 1000 products first, distributed to 100 shopping carts each, will be the winner.

Who do you think would win?

I would still bet on the laser winning the race because of its history and the fact that is had been optimized for exactly these kinds of high speed scenarios. However any relevant difference in technical or cost will continue to melt away for the next years.

One disadvantage of laser-based scanners though remains that it will never be able to scan 2D barcodes whereas CCD-scanners are the most versatile. There are many scenarios where versatility trumps speed. Consider yourself and your cellphone: you are never going to compete for speed with a professional cashier, but having the ability to scan barcodes might be very welcome function in your pocket.

All modern smartphones have a camera built-in for taking photos. With a bit of software intelligence all those cameras gain the ability to scan barcodes at no extra cost.

1.1.4 Where are the bars?

You might come to wonder why we are referring to the two-dimensional technology as "2D barcodes" even though there are no bars in sight. All 1D barcodes really do have bars representing a code. Most 2D barcodes have small squares. It would be grasping at straws to argue that squares are really bars that are very fat and not very tall.

It is the same technical inaccuracy that lets hyper-precise Apple engineers refer to these codes as **"machine-readable codes"** instead.

But the fact of the matter is that nobody so far was able to suggest a term for 2D barcodes that stuck and found wide usage. "Barcode" is easy to say fluidly with the B being formed with your lips and the C coming from the back of your throat.

Any other terms you sometimes hear - like "QR code" - refer to a single member of the family of 2D barcodes and thus are just as inaccurate. If you wanted to be extra-precise then the correct technical term would be something like: "machine-readable codes that use markings forming a matrix grid".

I hope you agree with me that "2D barcode" is still the best option, however inaccurate it may be. It simply rolls better off your tongue.

1.2 Barcode symbologies in iOS 7

There are way too many types of 1D and 2D barcodes to cover all of them in this book. This book is about Apple's support for barcodes and therefore we will only look at the types of barcodes that the operating system actually supports.

We are not going to the details of how the individual symbologies are constructed. This overview gives you the basics you need to understand those supported types, their abilities and where they are primarily used.

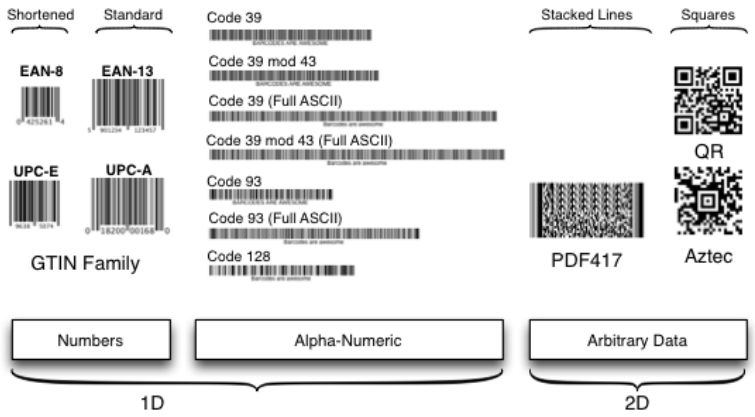


Figure 1.3 Barcode types supported by iOS 7

1.2.1 1D Barcodes in iOS 7

Barcodes are said to be having one dimension if there is a single line (e.g. traced by a laser of a scanner) that can cross all lines of the symbol. iPhones don't have a built in laser for scanning, but the single-line scanning can be emulated with the camera. The SDK scans multiple horizontal and vertical lines on the images coming from the camera and recognizes a 1D barcode as soon as one such scan line crosses all bars of the code.

THE GTIN FAMILY

The GS1 organization maintains the standards related to the **Global Trade Identification Number** (GTIN). There are several symbologies which belong to this family, all of them representing a product code, all of them using the same kind of barcode symbols.

- **UPC-A** - The "classic" first product barcode (12 digits)
- **UPC-E** - A less wide version for compressed product numbers (8 digits)
- **EAN-13** - The "classic European" barcode with an extra digit over the UPC-A, in Japan referred to as "Japanese Article Number (JAN)" (13 digits)
- **EAN-8** - A less wide version of the EAN for compressed product numbers (8 digits)

For the less wide symbologies product numbers can be are *compressed* by suppressing a string of zeroes inside. Manufacturers love to use these for products with little packaging real estate.

All product codes represented by barcodes of the GTIN family expand to **13-digit numbers**. UPC-A just gets an extra zero prefix. Uncompressing UPC-E and EAN-8 to 13 digits is slightly more complicated, but still possible.

Note: Paper magazines often have an additional barcode next to the GTIN. This is used to encode the issue number with EAN-2 or the price with EAN-5. Those are currently not supported by iOS and ignored when scanning the EAN-13.

Of these 13 digits the rightmost is the **check digit**. After scanning the check digit must match a fresh calculation from the numbers, otherwise it is considered corrupted.

The **leading first digit** is not directly encoded as bars, but represented by a certain pattern of odd and even bars. Thus US GTINs always use the same pattern, tied to the prefix 0 whereas international GTINs have 9 other patterns to choose from. When scanning a barcode the scanner infers the leading digit from the pattern of odd and even symbols.

GTINs are typically made up of a **manufacturer prefix** which gets assigned by GS1. Each manufacturer decides on the numbers he assigns to his products. If he were to assign a very low number, with several leading zeroes, then this allows for the above mentioned number compression.

The **International Standard Book Number (ISBN)** is a variant of the GTIN-13 used for books and book-like publications. Hereby the fixed prefix 978 or 979 replaces the manufacturer prefix, the remaining digits are assigned by national ISBN agencies. ISBNs have their own check digit which is calculated differently than the one in the GTIN.

The **International Standard Music Number (ISMN)** is a unique number for the identification of all notated music publications from all over the world. It uses fixed prefix 9790. ISMNs are also assigned by national agencies.

The third kind of special numbering scheme sitting on the back of GTIN is the **International Standard Serial Number (ISSN)** used for printed or electronic periodical publications. They share the 977 prefix and are also assigned.

Note: There are several additional ranges used for products that are packaged in stores, like cheese or meat. Also you can infer the country a product manufacturer is located in from these prefixes. Please refer to the table in the GTIN Prefix List in Appendix X.

Figure 1.4 Example showing all 4 members of the GTIN Family of Barcodes

CODE 39 AND CODE 39 MOD 43

Code 39 was developed by Dr. David Allais and Ray Stevens of **Intermec** (now a subsidiary of Honeywell Scanning and Mobility) in 1974 to overcome the limitation of the just-released UPC-A only being able to represent digits. They saw the promise of automating identification and data capture but wanted to be able to represent letters as well.

Code 39 is also known as Alpha39, Code 3 of 9, Code 3/9, Type 39, USS Code 39 and USD-3. It derives its name from initially being able to represent 39 different characters (plus a symbol to represent start and stop), generally only uppercase letters. With the help of control characters it can also represent the full **ASCII** set of characters.

Another advantage that Code 39 has over the GTIN is that is can be used for any length of text. As the number of characters increases so does the width of the barcode representation.

The plain version of Code 39 does not have a checksum to detect scanning errors. A variant of Code 39 exists that does have such a check digit, involving a modulo 43 operation. Because of this operation this variant is referred to as **Code 39 mod 43**. iOS supports both variants.

A human-readable version of the represented text is usually displayed beneath the code, with asterisks representing the start/stop character.

Note: Being as old as the UPC means that virtually all existing barcode scanners are able to read Code 39 codes.

Example showing how Netgear combining a UPC with 3 Code 39 to simplify adding this router device to a corporate tech inventory tracking system.

Figure 1.5 Example showing how Netgear combining a UPC with 3 Code 39 to simplify adding this router device to a corporate tech inventory tracking system

CODE 93

Code 93 could be considered the big brother of Code 39 if it weren't for the fact that it got developed by **Intermec** 8 years *after* its predecessor, namely in 1982. Its design goal was to reduce the horizontal space required for representing the same amount of text and improve error detection by adding a checksum right from the start.

The origins of the name are unknown, there is nothing in the standard with the number 93. Maybe it is just a play on the Name of its predecessor. By placing the 9 in front it sounds newer, bigger and better. That's marketing for you.

Code 93 at its core has 43 characters and 5 special characters. Similar to Code 39 it can represent all 128 ASCII characters with the help of control characters. This feature was tacked onto Code 39 by reusing codes, but in Code 93 dedicated control characters are used.

The main benefit is that if you scan a Code 93 with iOS you already get the ASCII characters decoded whereas with Code 39 you have to find the control sequences and do the decoding yourself.

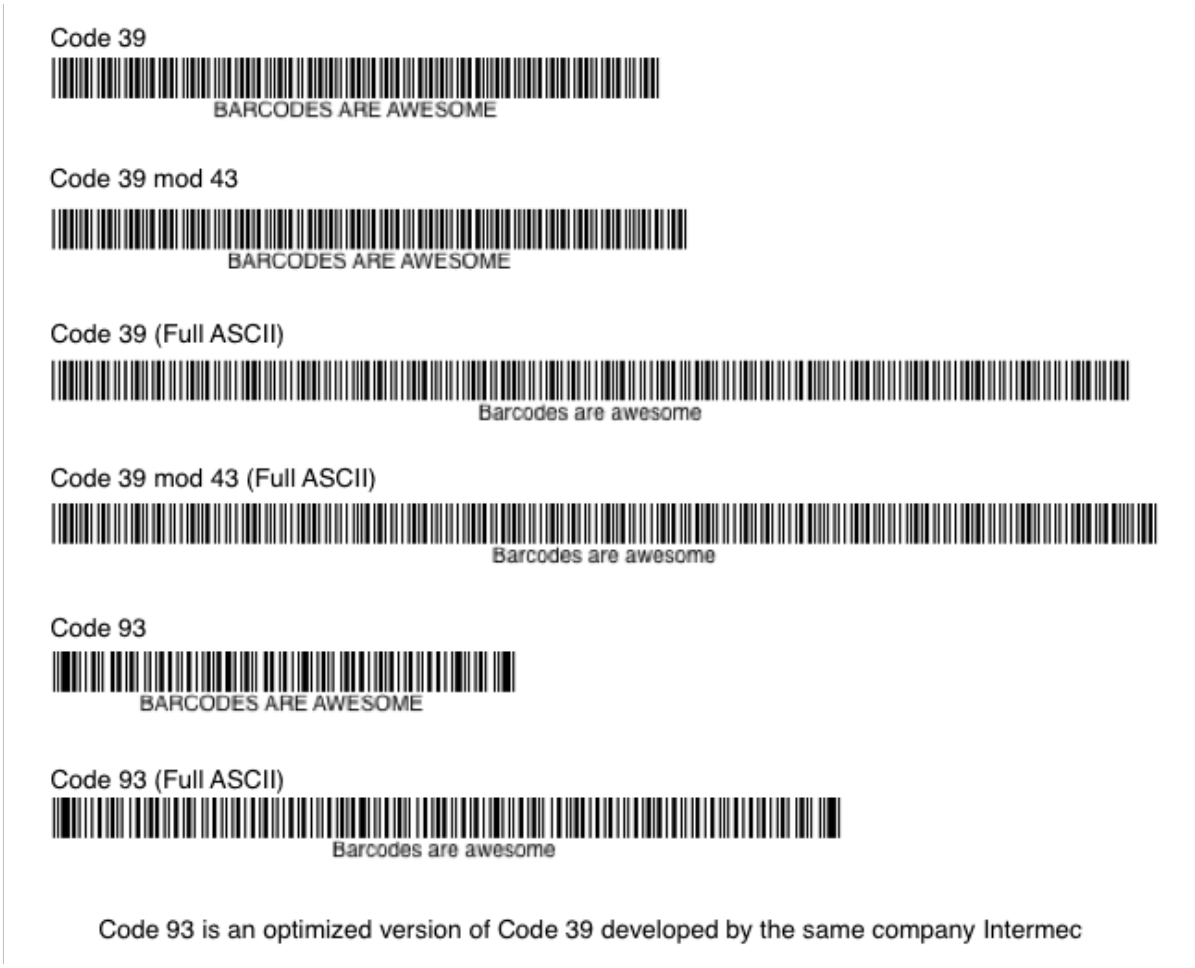


Figure 1.6 Code 93 is an optimized version of Code 39 developed by the same company Intermec

CODE 128

Code 128 was developed in 1981 by **Computer Identics Corporation** in 1981 and is named for being able to represent the 128 characters of the full ASCII code.

Data density in Code 128 is comparable to Code 93, and also the features are quite similar. This symbology has a more sophisticated mechanism for extending the range of characters by 3 different sets of codes via control characters. Those character tables are referred to as 128A, 128B and 128C.

Code 128 has a mechanism that allows it to save horizontal space. The 128C symbol set allows encoding two neighboring digits to be put in one code symbol. This compression mechanism was added to reduce the distance that the laser would have to travel when scanning such a code, which could become rather long depending on the number of characters contained.

This code was adopted by GS1 to represent supplementary product information like product weights, dimensions, expiration dates and many others. When used in

this context the concrete application of the Code 128 specification is referred to as **GS1-128**.

On the visual representation of such a GS1-128 the **application identifiers** are enclosed in round brackets. The full list of defined identifiers is in appendix X. On this list there is an interesting way employed to specify the position of the decimal comma.

For example 311y is the application identifier for "Product Length/1st Dimension, in meters". y can be any digit 0-9. The following integer is to be divided by 10 to the power of y.

You can easily spot a GS1-128 by looking for 2-3 digits enclosed in round brackets. Think of what cool applications you could build that read these codes of physical items.

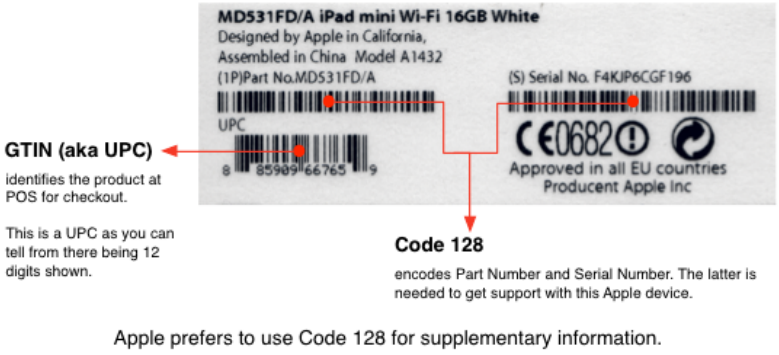


Figure 1.7 Apple prefers to use Code 128 for supplementary information

1.2.2 2D barcodes in iOS 7

A problem of barcodes encoding text strings of variable length is that there is a maximum useful width that could be used for the markings which is dictated by what they are printed on. You cannot put a longer string onto an envelope than fits on the paper. The obvious solution for this, represented by PDF417, was to wrap the single scanning dimension into multiple lines. Such a "stacked linear code" could be read by specialized (and thus more expensive) laser-based scanners.

When there is no single line that can cross all parts of a barcode in one scan then this code is being referred to as 2D and as a rule of thumb you need a camera for reading their digital contents. Several barcode symbologies were designed to be forming a two-dimensional square, most notably QR and Azetec.

Apple needed to add support for the following 3 barcode types to iOS to be able to represent modern digital boarding passes and tickets in **PassKit**.

PDF417

PDF417 was invented by Dr. Ynjiun P. Wang at **Symbol Technologies** in 1991. This company was founded in 1975, a mere 2 years after the initial UPC was announced, with its primary goal to go after the blooming retail and inventory management market.

The name PDF417 is in no way related to Adobe's popular document format sharing the same abbreviation, which adds some confusion amongst people first dealing with it. Rather it is short for *Portable Data File*. It consists of symbols that contain 4 bars and spaces each, with each symbol being 17 units long.

The greatest advantage of PDF417 is that you can decide how wide each line can be and how high each individual line should be. PDF417 - even though being patented - is fully in the public domain and thus free of all user restrictions, licenses or fees.

It is because of this versatility and being free to use that it became the 2D barcode of choice for a great variety of use cases.

PDF417 one of the formats accepted by the **United States Postal Service** for printing postage. More than 200 airlines have settled on using it as the **Bar-coded Boarding Pass** standard since 2005. It is also used by package services and on ID cards.

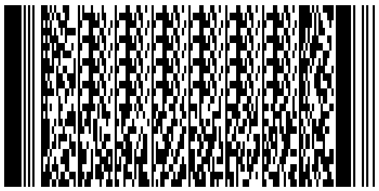


Figure 1.8 Example showing PDF417 2D Barcode

QR CODE

In contrast to PDF417 the QR Code never had roots in the one-dimensional space. It was designed from the ground up to be read by CCDs.

Toyota's subsidiary **Denso Wave**, an automotive components manufacturer, invented the QR Code in 1994 for tracking parts around the car factories. QR codes can represent any kind of data with great density. As the length of data encoded grows so does the square area of the code or the individual squares shrink. This means that the data density is only limited by the resolution of the scanner camera.

The QR code gained notoriety outside of the auto industry being used to encode

website addresses, mostly in print media and advertising. For the longest time it was made fun of for this reason, as being a Japanese fad where almost all phones have the ability to read QR codes nowadays.

Since QR codes can represent any arbitrary data use cases range from encoding vCards to audio files. For example visually impaired people could scan a QR sticker on a box containing an audio recording of verbal instructions.

There is no universal standard how certain kinds of complex data should be represented in QR, it is agnostic to what data you want to put in there.

QR Codes embed multiple error-correction symbols which make them extremely resilient.

Figure 1.9 Example showing QR Code

AZTEC CODE

Aztec Code was invented in 1995 by Andrew Longacre, Jr. and Robert Hussey. It was published by **Automated Industry Machines (AIM)** two years later. Its original purpose inside this company is not publicly known.

You can tell Aztec codes apart from QR codes if you look for the number of concentric squares. Aztec has one such square in the center whereas QR has these boxes at 3 of the 4 corners of the code. Those squares are used by scanners to align themselves on the code and determine its orientation.

The name of the code derives from the center "square bulls eye" being reminiscent of an Aztec pyramid. Of course the Aztec Code as built-in error correction as well.

This code is also one of the 3 barcode types that can be used on boarding passes following the **Bar-coded Boarding Pass** standard. On top of that it is used by a great number of railway companies for digital train tickets.

Just like with the QR code you can represent any kind of arbitrary data with

Aztec codes so in practice it is a matter of preference by the developer which of the two you would prefer to represent your data.

Figure 1.10 Example showing Aztec Code

SO MANY CHOICES: WHICH BARCODE SHOULD I USE?

Out of a large amount of different barcode symbologies Apple picked the ones that make the most sense for mobile applications. If you are building apps to read existing codes then it is very likely that one of the mentioned barcodes has you covered.

- Physical products, all items that are scanned at the point of sale: **GTIN Family**
- relatively short alphanumeric texts, serial numbers: **Code 93** or **Code 128**
- Digital tickets, boarding passes, loyalty cards, PassKit: **PDF417**, **QR** or **Aztec**
- Arbitrary data in a square space: **QR** or **Aztec**
- Arbitrary data using less height than width: **PDF417**

iOS 7 can generate bitmaps of all mentioned 2D barcodes and provides scanning ability of all mentioned 1D and 2D barcodes. To fill the niche of generating 1D barcodes I created a library dubbed **BarCodeKit**. This commercial library is available to readers of this book at no charge. All you need to do is find a QR code with the secret ordering link which I have hidden in the pages of this book.

1.3 UPC, the mother of all barcodes

When getting acqainted with barcodes used for tagging products you will get in contact with a plethora of abbreviations. Some of of these will be names of barcode symbologies while others are names of companies or organizations. Particularily the barcodes representing product numbers had several different names in the course of their history as well as multiple organizations developing and maintaining the related standards. Most people find this quite confusing initially.

This section provides an overview of the timeline of the very first barcode, the Universal Product Identifier (UPC). This history has several informative as well as amusing lessons for us modern people. And besides of being fun to know how the modern barcode came to be, this brief history also reveals the connections between several actors, standards and organizations.

While your colleagues will still be confused as the difference between UPC-A, EAN-13 and GTIN, you will be able to confidently sound the part of the *Barcode Guru* when discussing barcodes for use in your company or apps.

1.3.1 Bull's eye origins

An old saying is "war is good for business". Indeed the United States of America experienced an unprecedented economic upturn during the second World War (1939-1945) as soldiers were removed from their normal workplace to fight abroad, and jobs at home were filled by formerly unemployed.

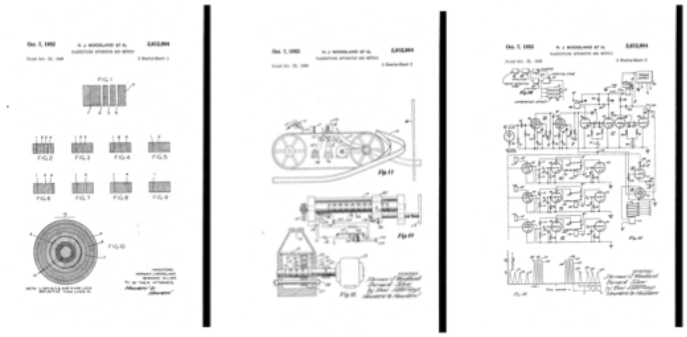
The grocery industry forcing to take a hard look into how it could scale its processes to cope with the increased demand. So one day in 1940 an enterprising grocery executive visited the Engineering College at Drexel University in Philadelphia. He hoped to challenge them to develop a method for automating product identification at checkout, possibly gaining a commercial advantage over his competitors with academia doing the research at low cost.

Drexel University declined the challenge. **Bernhard Silver**, being a graduate student there at this time, overheard the request and brought the idea to his friend **Joe Woodland**. They saw an opportunity for inventing something revolutionary and they went to work.

Automating anything in the middle of the 20th century meant physical machinery. So there was a need to develop some marking scheme or identifier that could be affixed to products which could be "read" by a machine. Once the item was identified changing the charged price accordingly at the checkout counter would be an easy second step.

Joe Woodland got hit by a stroke of genius when he absentmindedly ran his fingers through the sand on a beach. He realized that by varying the thickness of lines he could represent different numbers, much like in Morse Code but with more "symbols" than just a dot and a dash. Woodland had learned the Morse Code when he was a Boy Scout.

The two friends Silver and Woodland continued to work on their technology for 9 more years. They applied for a US Patent in 1949 and it was granted as #2612994 in 1952.



Patent US2612994 is the earliest documented form of a machine-readable code

Figure 1.11 Patent US2612994 is the earliest documented barcode

1.3.2 Startup story

The patented approach was to have circular lines around a center point printed in reflective ink. The reasoning behind this was that you could scan such symbols in any direction always arriving at the same result. This design being reminiscent of a dart board earned it the unofficial name "Bull's Eye Symbol".

Even though Woodland tried hard to find investors for their technology he had no tangible success. Their "proof of concept" experimental system - installed in a back room of a grocery store - did not convince anybody. Finally the **Philco** company bought the patent from them for $15,000 which equate to $132,000 of today's dollars.

Their invention did not yield immeasurable riches to comfortably retire and so Woodman signed up as an engineer with IBM who was not interested in the patent at that time. Bernard Silver passed in 1962.

It might have been lack of marketing or lack of practical applicability of their solution, but the Bull's Eye Symbol remained unused for several decades.

1.3.3 Military efficiency improving the grocery industry

The US grocery industry had never ceased to keep improving their distribution channels and became quite efficient in the early 1960s. Human manual work remained the single most limiting factor slowing their growth.

- Labor cost of retail checkout was a major cost factor. About 40% of labor expenditures went to checkers and baggers.
- Inventory Tracking remained a manual process. Cash registers then were only able to track the amount paid and which of 5-6 departments the item was from.
- The high rate of errors and slow transaction speed of humans caused a lot of problems. Retailers and manufacturers all had their own product codes, requiring manual copying and translating of these on invoices and other forms.

Several independent projects were started during the 1960s to address these issues and improve productivity at checkout. The US military had logistic efficiencies in place due to the fact that it employed many contractors developing related technologies to increase efficiency. Having a strictly hierarchical structure and central place to work with also made it more attractive for contractors to work for.

The Kroger Company of Cincinnati, Ohio realized this and sponsored a high tech conference in 1966. Their aim was to educate military contractors about the grocery distribution environment. Technologies originally developed for military logistics could also benefit the grocery industry to become more efficient and save cost.

Amongst the attendees of this conference was **RCA** who had purchased the Bull's Eye barcode patent from **Philco** a few years earlier. RCA formed a partnership with Kroger to develop a laser-based scanner prototype by 1968. This prototype was used to analyze performance of various configurations of checkout machines in their Princeton, New Jersey laboratory.

1.3.4 Confusion in numbers

Getting a grip on the basic technology of using a laser for reading barcodes was the first hurdle. The second was the need for agreement on a common scheme for identifying products.

With individual retailers, grocery manufacturers, the National Association of Food Chains (NAFC) and the Grocery Manufacturer's Association (GMA) all having ideas and concerns there was no common ground nor leadership to establishing a standard.

Manufacturers - being the first in the chain of products - already had case codes and they favored simply adding a prefix, with eleven or more characters per code. Retailers on the other hand - worried about wasting too much packaging space with barcode symbols - wanted to go for seven or less.

In all this disagreeing, the groups agreed on one thing: to form the **Ad Hoc Committee on a Uniform Grocery Product Code** in August 1970 It was given the goal would be to create a standard for the numbering as well as the machine-readable representation.

1.3.5 CEO-driven development

Fortunately the arguing trade association members had the wisdom take a back seat and let a good dozen key grocery manufacture and retail executives form this committee themselves. The committee selected **Burt Gookin, CEO of H.J. Heinz Company** to be chairman and **McKinsey and Company** were hired to manage the project.

McKinsey took a rather scientific approach to the problem. They developed a model which allowed participants to measure and evaluate the economic impact of different proposals. Having such a fact-based common evaluation framework let the industry see that this would eventually benefit everybody.

A study done by the GMA had shown that 95% of products sold in the USA had 5 or less numerical characters in their case code. Airplane travel at this time still involved good food and lots of beverages. The folk tale goes that, on a flight returning from a west coast brainstorming session, the members of the Ad Hoc Committee realized the solution to the numbering dilemma.

They would assign a 5-digit prefix to each manufacturer and let him select another 5 digits forming a 10 digit code. Later an 11th digit was added to discern 10 different numbering ranges.

Another smart move was that no single executive from the committee wanted to claim responsibility for this idea. Rather, McKinsey presented the approach to hundreds of companies and all but two readily stated in writing that they would support it.

This was smart because coming from a single person or company there might have been severe backlash, due to fears that a competitor might gain an unfair advantage. Presenting it as the best-of-breed idea that the entire committee came up with prompted an almost-universal "we too" response.

At the May 1971 Supermarket Industries trade show, the Ad Hoc Committee presented the general agreement on a coding format - a major milestone.

1.3.6 Making it machine-readable

The second part of this undertaking was still missing - the numbers of the coding format would have to be made readable by machines in order to allow for automation.

Over the next two years following the 1971 announcement, companies busily worked on their proposals for the committee. This ended in a three-day meeting held in January 1973 where each contender got a 20 minute slot to present its approach.

The two strongest contenders turned out to be RCA Bull's Eye Code and IBM's solution. The circular code still seemed to have the upper hand because of the omnidirectional readability and an existing laser-based scanning technology. Also IBM - being extremely secretive about under-development projects - had not shown anything public before this time.

Originally IBM had stated that they were not interested in participating. But IBM's **George J. Laurer** had recognized a fatal flaw in RCA's symbol. Printing it on product packages would often smear the circles in the direction of the paper feed. This rendered the code unreadable.

Laurer's barcode design was linear, and you could orient the code such that the smearing during printing would only elongate the lines slightly but preserve readability. Woodland who was still employed by IBM at this time provided valuable input.

When IBM's Senior VP Bob Evans made the presentation he said:

> I know you may have concerns about what computer could keep up with scanning this symbol in a store checkout

He then reached into his pocket and produced a disk containing schematics for micro-circuits.

> Each circuit on this disk is equivalent to a moderate size existing computer. If IBM were to develop a system, we would put one of these circuits in each checkstand.

Evans spoke this words after Intel had created the 4004 processor in 1971 and 8008 processor in 1972. Intel's co-founder Gordon E. Moore had observed that the number of transistors of CPUs would be doubling in number every two years. This

statement was given the name "Moore's Law" in 1970.

The computer revolution had just begun and made IBM's barcode feasible.

1.3.7 And the winner is ...

As before with the number format, the Ad Hoc Committee didn't want to make it sound like IBM was the winner of the contest, and conversely everybody else a loser. So, they slightly modified IBM's design by trimming off the tops of the code's longer marker lines. They also wanted to have somebody neutral to evaluate the proposal.

MIT in Cambridge performed the evaluation and recommended changing the numbers at the bottom to the OCR-B font. They argued that in a few years the bars would not be needed any more since then computers probably would be able to read the numbers directly.

Finally - after 3 years - the **Universal Product Code** (UPC) was announced in a press release in April 1973. The Winner being *everybody*.

The first UPC-marked item ever scanned was a 10-pack (50 sticks) of Wrigley's Juicy Fruit chewing gum which is on display at the Smithsonian Institution's National Museum of American History in Washington, D.C. This historic occasion took place on June 26, 1974, a mere month before this author's date of birth.

Joe Woodland of IBM was honored with the National Medal of Technology in 1992 for "inventing the barcode", incidentally the same year that Bill Gates of Microsoft received it, too.

1.3.8 UPC plus EAN equals GTIN

Shortly thereafter the **Uniform Product Code Council** (UPCC) was formed to oversee the administration of the UPC system. It was founded as a not-for-profit standards organization.

3 years later a European counterpart to the UPCC was formed, the **European Article Numbering (EAN) Association** based in Brussels, Belgium. To further increase the range of numbers a 13th digit was added. What followed was a meteoric rise of the adoption of the UPC/EAN code around the world.

At this point the confusion reached its climax, since the term UPC was colloquially used to refer to the UPC barcode, the UPC product number as well as the UPC organization. The same happened in Europe where the EAN abbreviation could mean the EAN barcode, the EAN product number or the EAN organization.

In 1978 Japan joined the EAN and adopted the EAN for use in Japan, calling it the **Japanese Article Number (JAN)**. Originally founded by 12 European

countries the EAN got joined by many countries outside of the European continent. For many years the European and US organizations worked alongside each other until in 1990 they signed a formal agreement to jointly manage the standard.

15 further years passed until UPCC and the EAN Association decided to merge. EAN was renamed to **GS1 International** the UPCC became GS1 US. This merge greatly reduced the confusion over the many organizations being in charge of the same numbering and barcode standards.

The true origins of the name GS1 of shrouded in mystery. One can only assume that GS1 means something like "Global Standard Number 1". A bold statement.

The second simplification step occured in 2009 when the UPC/EAN/JAN codes were renamed to **Global Trade Identification Number (GTIN)**. GS1 would love if you would only refer to their product barcodes to GTINs from here on, but because the other names had been in use for over 40 years you still see UPC and EAN used colloquially.

On the UPC's 40th anniversary in 2013 GS1 International had presences in 111 countries and standardizes many other things - mostly related to commerce - besides the bar code. Most barcode standards are actually being maintained by the **International Standards Organization (ISO)** at the lowest technical level.

GS1 bases their standardization work on these but adding the semantics necessary for using these bar codes in the context of commerce communication. This is why the GS1 tagline reads "The global language of business".

TIP	**Barcode Guru Tip**
	Use the name **GS1** to refer the standards organiziation. When referring to product numbers or barcodes call them **GTIN**s.

1.4 Summary

2013 marked the 40-year anniversary of the first widely used barcode which eventually became the GTIN. It's original goal was to increase the efficiency of the grocery industry by enable automatic product identification at the point of sale. And this goal it fulfilled many times over as the entire world adopted it. The international organization GS1 maintains it. Most barcode symbologies are ISO standards at the lowest technical level. GS1 put itself in charge of defining the semantic meaning of contents represented as GTINs and in Code 128.

Other industries had different needs and this lead to the development of barcode symbologies which could represent alpha-numeric characters as well. Code 39

being the oldest amongst the barcode types supported by iOS. Code 93 and Code 128 representing two more advanced symbologies.

The advancements in digital image processing gave rise to a new kind of barcodes using more than one dimension. Small, inexpensive cameras on a chip, called CCD, were able to scan 2D barcodes as well as the older 1D barcodes which initially could only be scanned with a laser beam. As it became a de facto standard for smartphones to have a built-in camer this put a fully fledged barcode scanner in everyone's pocket.

Before the rise of the mobile phone, barcodes were only useful for places that had scanning equipment installed. Point-of-Sale (POS) systems had bulky cash registers with built-in laser scanners and a database for looking up price information. Much of these previously exclusive technologies are now available in modern smartphones. Not only can users now scan barcodes with a device they are carrying with them everywhere they go. But always-on Internet connectivity and devices sensors detecting the user's current context add degrees of utility which have never been possible before.

iOS7 is truly democratizing the barcode.