Published November 1, 2001. Distribution restricted to sponsors until February 1, 2002.

AUTO-ID CENTER

WHITE PAPER
The Compact Electronic Product Code
A 64-bit Representation of the Electronic Product Code

David L. Brock

MIT AUTO-ID CENTER MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 77 MASSACHUSETTS AVENUE, BUILDING 3-449G, CAMBRIDGE, MA 02139-4307

ABSTRACT

The Electronic Product Code (EPC) was conceived as a means to identify all physical objects. The EPC was intended to be a short, simple and extensible code designed primarily to reference networked information. A 96-bit Electronic Product Code has been defines; however, the incremental cost of encoding additional bits on electronic tags prompted the investigation of a reduced size, or "compact," Electronic Product Code. This paper explores a reduced bit count from the original 96-bit version of the EPC and proposes a specific 64-bit variant of the Electronic Product Code.

Published November 1, 2001. Distribution restricted to sponsors until February 1, 2002.

WHITE PAPER

The Compact Electronic Product Code

A 64-bit Representation of the Electronic Product Code

Biography

by David L. Brock
Co-Director

Dr. David Brock received Bachelors degrees in theoretical mathematics and mechanical engineering from MIT, and his Masters and Ph.D. degrees from the Department of Mechanical Engineering at MIT with an affiliation to the Artificial Intelligence Lab. He is currently a Principal Research Scientist in the Laboratory for Manufacturing and Productivity and Co-Director of the MIT Auto-ID Center. Dr. Brock is also the Founder of Brock Rogers Surgical, a manufacturer of robotic medical devices. Dr. Brock has worked with a number of organizations including the Artificial Intelligence Laboratory, the Massachusetts Eye and Ear Infirmary, DARPA, Lockheed-Martin, Loral, BBN and Draper Laboratories.

Published November 1, 2001. Distribution restricted to sponsors until February 1, 2002.

WHITE PAPER

The Compact Electronic Product Code

A 64-bit Representation of the Electronic Product Code

Contents

The Electronic Product Code (EPC) was conceived as a means to identify all physical objects. The EPC was intended to be a short, simple and extensible code designed primarily to reference networked information. A 96-bit Electronic Product Code has been defines; however, the incremental cost of encoding additional bits on electronic tags prompted the investigation of a reduced size, or "compact," Electronic Product Code. This paper explores a reduced bit count from the original 96-bit version of the EPC and proposes a specific 64-bit variant of the Electronic Product Code.

1. INTRODUCTION

The Electronic Product Code (EPC) was conceived as a means to identify all physical objects. The primary purpose of the EPC was to serve as a reference to networked information [3]. Used in conjunction with the Object Name Service, the EPC associates the physical object with information about the object – written in the Physical Markup Language (PML) [4]. Together these components allow physical objects to be networked together – creating essentially an **'Internet of Things'** [1,2].

Since the EPC identifies 'all physical objects,' it must be sufficiently large to enumerate at least those objects of interest for purposes of tracking and identification. The 96-bit version of the EPC code allows approximately 8×10^{28}, or 80 thousand trillion trillion objects – more than sufficient for man-made physical products [3].

Although 96-bits is small and fits easily into many commercial Radio Frequency Identification (RFID) tags, there is a desire to reduce this number still further to continue to cut the cost of the electronic identifier. This paper presents a reduced size identification scheme – specifically a set of 64-bit Electronic Product Code (EPC-64) – which is a proper subset of the 96-bit version.

The following sections provide a brief background, a design strategy and a specific proposal for the EPC-64.

2. BACKGROUND

The Electronic Product Code is a unique identification scheme containing four partitions: header, EPC manager, object class and serial number. The 96-bit version of the Electronic Product Code (EPC-96) is shown in Figure 1 [3]. The code includes a fixed, 8-bit header. This header defines the number, type and length of all subsequent data partitions. Thus, the single byte provides 256 possible partitioning schemes.

The first scheme, EPC-96 Type I, was intended as a public object identification number. It is used in the same way as the current Uniform Product Code (UPC), as well as UCC.EAN Shipping Container Codes [5-8].

The EPC-96 Type I has three data partitions, shown in Figure 1. The first data partition, after the header, identifies the EPC manager; that is the manufacturer, or entity, responsible for maintaining the subsequent codes. The EPC manager is responsible for maintaining the object type codes and serial numbers in their domain. The EPC manager must also ensure reliable operation of the Object Name Service (ONS) and for maintaining and publishing associate product documents [2].

The EPC-96 manager partition spans a 28-bit section, encoding a maximum of $2^{28} = 268,435,456$, or approximately 268 million, manufacturers. This exceeds the 100,000 managers possible with the UPC-12 and the 1,000,000 for the EAN-13.

The next partition, object class, occupies the next 24-bits of the EPC-96. The object class may be considered the product skew or stock keeping unit (SKU). It may also be used for lot number, or any other object grouping scheme developed by the EPC manager. Since each manufacturer is allowed more than 16 million object types, this partition could encode all the current UPC SKUs, as well as many other object classes.

The final partition encodes a unique object identification number. For all objects of a similar type, the EPC-96 serial number provides 36-bits, or 2^{36} = 68,719,476,736, unique identifiers. Together with the product code, this provides each manufacturer with 1.1×10^{18} unique item numbers — currently beyond the range of all identified products.

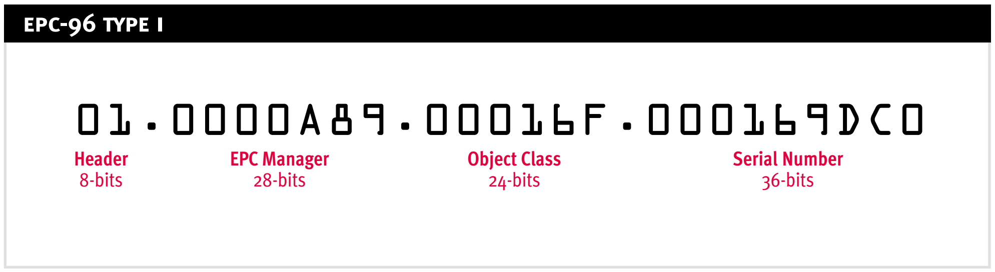


Figure 1. The EPC-96 Type I encodes is public object identification numbers.

3. DESIGN STRATEGY

In this section, we consider the design strategy for a compact identification number, which will serve the same function – albeit with limited capacity – as the 96-bit EPC. We include underlying assumptions, theoretic constraints and practical implementations, which lead to a particular design approach.

3.1. Object Identification

It is assumed the smaller identification code proposed here would serve the same function as the 96-bit version of the Electronic Product Code; that is the identification of physical objects. Therefore the same reasoning and assumptions, which went into the original design of the EPC-96 apply here [3]. Thus the same types of partitions exist. These include a header, EPC manager, object class and – for unique identification – a serial number.

Since minimum code size is critical, an optimal, variable partitioning scheme should at least be considered. The disadvantages of variable partition schemes – more complex descriptions, parsers, routers and software – should, however, limit the extent of partition variations.

3.2. Subset of the Electronic Product Code

Any reduced size, or compact, object identification code should be a proper subset of the 96-bit Electronic Product Code. In this way the smaller code can map easily into the EPC-96. This also implies partitions not included in the 96-bit version of the EPC cannot be present in the reduced identification scheme. Similarly there can be no partitions in the smaller code, which are larger than the corresponding partitions of the 96-bit EPC. This assumption reduces the range of possible identification code designs, as illustrated in Figure 2.

3.3. Header

We will want some type of header in any reduced code design. Without a header it will be impossible to expand, change or modify the coding scheme for a particular size.

The smallest possible header partition size is a single bit, 1-bit, and the largest, given our assumptions, is 8-bits. However, since we are attempting to reduce the size of all partitions, it makes sense to define a lower, maximum bound for the header size. Therefore, we will set a maximum header size to one-half byte – a nibble or 4-bits. Thus the minimum number of partitioning schemes is two (for a single bit header) and the maximum is 16 for a 4-bit header, as illustrated in Figure 2.

3.4. Manager Partition

As in the original 96-bit EPC design, the reduced sized code will include a manager number. A reasonable lower bound on the size of the manager number is at least as large as the number of current EAN.UCC member company numbers. Since the EAN.UCC system includes approximately 1,000,000 companies, the manager partition should include at least 20-bits – allowing 1,048,576 numbers. The upper bound is set by the 96-bit EPC specification at 28-bits, or 268,435,456 companies. Thus the manager partition size for smaller code should range between 20 and 28-bits, as shown in Figure 2.

POSSIBLE DESIGNS FOR SMALLER SIZED OBJECTS

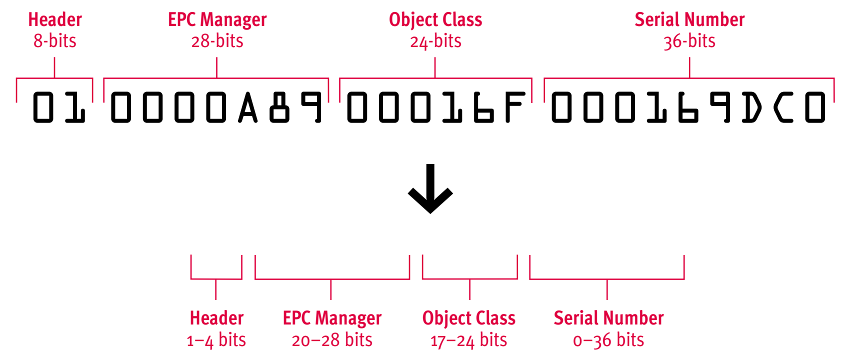


Figure 2. There is a range of possible designs for a smaller-sized object identification scheme consistent with the 96-bit Electronic Product Code (EPC) specification.

3.5. Object Class Partition

As with the manager number, the object class partition should provide at least the capacity of the EAN.UCC system. The Uniform Product Code (UPC) provides 5-digits, or up to 100,000, possible stock keeping unit (SKU) numbers for each company. The object identification number should, therefore, include at least 17-bits or $2^{17} = 131{,}072$ object class numbers.

As with the manager partition, the maximum size of the object class number is limited to the size of the corresponding 96-bit EPC partition, which is 24-bits. Therefore the object class partition should range between 17 and 24-bits, as shown in Figure 2.

3.6. Serial Number

Current UPC and EAN item numbers are not serialized. Therefore, the obvious minimum size for serial number of the reduced size object identification is zero.
If serialization is included it must be of practical benefit to the member companies. Given the extraordinary number of items manufactured by some companies (Gillette, for example, ships billions of razor blades each year) it is difficult to set a reasonable lower limit.

If serialization were applied to shipment or lot number the size of the serial number partition is reduced from many billions to millions. Thus, the practical lower bound on the size of the serial number will range from approximately 20 to 24-bits, or 2^{20} = 1,048,576 to 2^{24} = 4,194,304.

The upper bound is again defined by the 96-bit EPC specification at 36-bits or 2^{36} = 68,719,476,736. Thus the minimum size for a usable serial number should be 20 to 36-bits, as shown in Figure 2.

3.7. Multiple Versions

It is difficult for a **single** 64-bit EPC version to meet all the requirements of manager, object and serial numbers. (a more description of these requirements is given in [3]). Therefore a **set** of 64-bit versions may be necessary to more effectively **cover** the range of numbering requirements.

In other words, multiple verions could include a one with few manager numbers and many object types and serial numbers and another with many managers and few object types and serial numbers.
The partitions – header, manager, object and serial number – exist for multiple schemes, as well as the maximum partition sizes, as presented in the previous sections. The number of bits allocated to each partition differs between versions.

The minimum partition size for a particular scheme, however, may be smaller than those described above, as long as a least one of the versions meets the minimum requirements. In other words, the set of versions should cover all the minimum partition size requirements.

A set of versions for the 64-bit EPC code should all be proper subsets of a **single** 96-bit EPC code type – in this case the 96-bit EPC code type I as shown in Figure 1.

3.8. Application

In the design of the Electronic Product Code, we must consider how the code might be used and develop assumptions about the depth and breadth of its application.

First, we assume the 64-bit version of the EPC code will be transitional, that is it will be used for a time before the widespread use of a 96-bit or longer identifier. We do not assume the 64-bit version will be universally adopted or meet all the needs of all manufactures. Therefore we want at least one variant of the 64-bit code to cover the set of early adopters – understanding the longer 96-bit version will meet more universal demand. Furthermore we will ensure any version of the Electronic Product Code to be fully compatible with each other, so that a 64-bit, 96-bit or other version will be transparent to the system.

Second, we assume the 64-bit version will be used predominantly by larger manufactures, and those with more cost sensitive consumer products. As the cost of memory decreases and the production of electronic tags increases, the incremental cost of an additional 32-bits will have decreasing impact on users. Since we are assuming the application of a 64-bit Electronic Product Code particularly to high-volume, low-cost products, we must provide at least one type of the EPC-64 that can accommodates a large number of items.

Third, we assume the application of the Electronic Product Code to extend beyond traditional consumer products. These may include manufacturing, software and services. To encourage this development, we may want to include a variation of the 64-bit Electronic Product Code that allows a significantly greater range of companies and organizations. Most of these users will have far fewer products and items than larger consumer product companies.

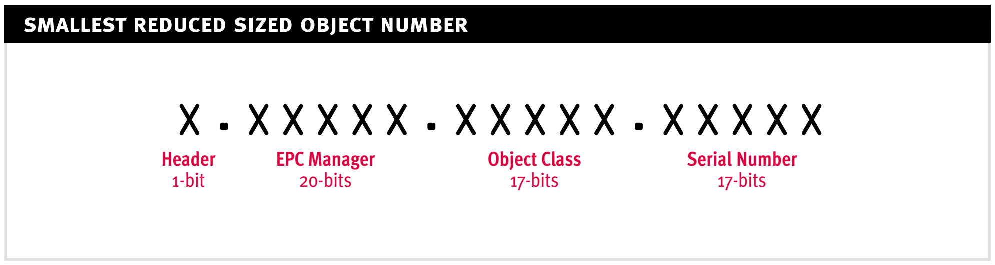
These underlying assumptions drive the design of the structure and organization of the 64-bit EPC. In the following section, we propose a series of 64-bit EPC types that attempt to cover these anticipated uses and meet the demand of a wide range of industries and applications.

4. DESIGN

Given the discussion in the previous section, a practical reduced-sized object identification number, which is consistent with the current 96-bit Electronic Product Code type I is defined by the partitions and partition sizes shown in Figure 2.

Using the minimum partition sizes, the smallest possible serialized identification scheme is 58-bits consisting of a single bit header, 20-bit manager, 17-bit object class and a 20-bit serial number, as shown in Figure 3.

Figure 3. Smallest reduced sized object number based on the assumptions given here is 58-bits.



4.1. The 64-bit Electronic Product Code (EPC-64) Type I

A 64-bit Electronic Product Code (EPC-64) could contain the minimum object identification number as described in Figure 3. We must therefore consider the allocation of the remaining 6-bits.

Since the conservation of bits is of prime importance in the smaller number, let us consider the allocation into the data partitions a higher priority than the header. A single additional bit in the header allows three data partition schemes, which cover a large range of industry needs, while leaving a fourth header value for expandability.

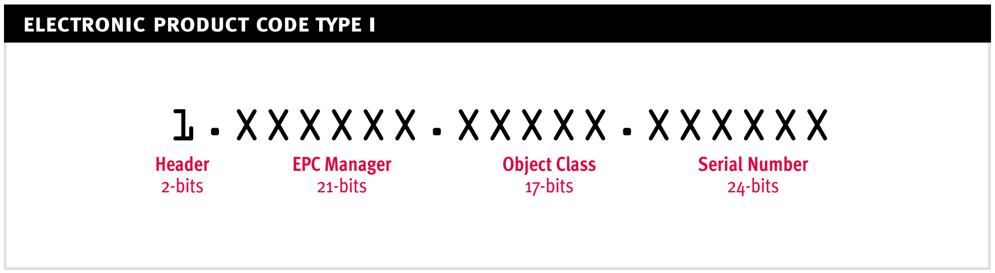
The 20-bit manager partition provides for only 1 million companies – barely adequate even for current usage among UCC.EAN member companies. Allocating an additional bit allows 2 million organizations that can use the smaller EPC-64 type I scheme.

The object class partition provides 131,072 stock keeping units – more than is provided for by the UPC code and more than most companies require.

The serial number, providing only 1 million individualized products, is insufficient for most companies that would want to use the EPC-64. By allocating the remaining 4-bits in this partition, the serial number grows to 24-bits allowing up to 16 million unique products (or lot numbers).

The 64-bit Electronic Product Code type I therefore provides a two bit header, 21-bit manager number, 17-bit stock keeping unit and 24-bit serial (or lot) number, as summarized in Figure 4.

Figure 4. The 64-bit Electronic Product Code (EPC-64) type I.



Given the assumptions from the previous section, we have proposed a 64-bit EPC scheme that accommodates up to 2 million manufactures – more than twice the current EAN.UCC membership. This should address all the potential “early adopters,” as well as a significant portion of latter developers.

With this scheme, the object type partition exceeds the current allocation provided by the Uniform Product Code. Therefore, we assume this partition to be sufficient for most companies.

Finally, although 16 million serial numbers may be inadequate for many manufacturers, we assume serialization by lot and date would still be practical. Furthermore, based on our assumptions, we include a 64-bit EPC type that specifically addresses those manufactures with a large number of serialized products.

4.2. The 64-bit Electronic Product Code (EPC-64) Types II

In addition to EPC-64 type I, other versions schemes may be added to cover a wider variety of companies, products and serial numbers. Specifically, we propose an EPC-64 type II to address the product identification needs of large volume, cost sensitive consumer producers.

For those companies who desire unique product identification and whose production exceeds the approximately 2 trillion unique codes (2,199,023,255,552 or 2^{17} = 131,072 product types together 2^{24} = 16,777,216 serial numbers) provided by the EPC-64 type I, we define an EPC-64 type II.

Beginning with the serial number, we propose a 34-bit partition, providing 17,179,869,184 identifiers for individual product identification, as shown in Figure 5. Combined with a 13-bit object class partition – allowing up to 8,192 stock keeping units – each manufacturer has 140,737,488,355,328 or over 140 **trillion** individual item numbers. This exceeds the total output of even the world’s largest consumer products manufacturer.

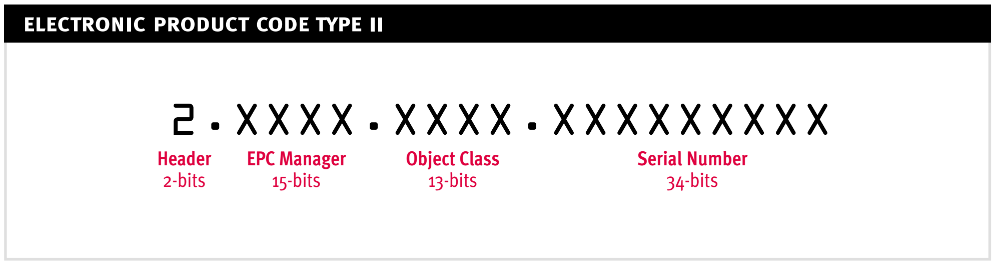


Figure 5. The 64-bit Electronic Product Code (EPC-64) type II.

The total number of companies who may use this particular EPC-64 type is 32,768, which is dictated by the 15-bit manager partition remaining in the 64-bit code. We can safely assume that fewer than 32,768 companies produce identifiable products in the multi-billion item range.

Again these restrictions are eliminated as we move toward the larger 96-bit versions of the Electronic Product Code.

4.3. The 64-bit Electronic Product Code (EPC-64) Types III

In addition to large manufacturers and those currently using the UCC.EAN numbering standards, we wish to expand the adoption of the Electronic Product Code to broader organizations and industries.

To facilitate this adoption, we wish to extend – even in the 64-bit version of the EPC – a partitioning scheme that accommodates smaller companies, service industries and organizations. Thus rather than expanding the number of items numbers, as we had for EPC-64 type II, we will increase the total number of companies.

By increasing the manager partition to 26-bits, as shown in Figure 6, we provide up to 67,108,864 companies who may use the 64-bit version of the Electronic Product Code. Since 67 million exceeds the number of the world’s corporations, it should be sufficient for now. Moreover, by accommodating a large number of companies and organizations, we hope to encourage a wider acceptance of the Electronic Product Code system.

As with the EPC-64 type II, we allow 13-bits for the object class partition providing 8,192 unique object types. The remaining 23-bits in the serial number partition allows over 8 million (2^{23} = 8,388,608) item numbers. Thus each of the 67 million companies are allowed over 68 billion (2^{36} = 68,719,476,736) unique product numbers.

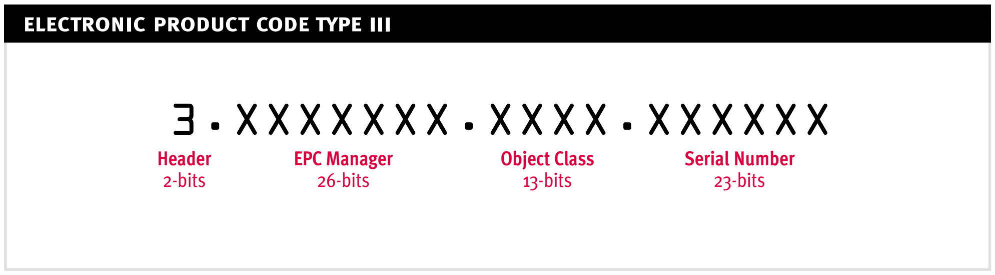


Figure 6. The 64-bit Electronic Product Code (EPC-64) type III.

5. CONCLUSION

The Electronic Product Code uniquely identifies physical objects. The type I version of the 96-bit EPC provides a single representation usable by nearly every manufacture.

The desire to reduce the cost of the electronic identifier, however, strongly encouraged the design of a smaller product code. Therefore we have considered the design and organization of reduced-sized Electronic Product Code – specifically a 64-bit EPC.

Because of this reduced size, it was difficult to accommodate the needs of every manufacturer within a single version. Some have many product types; others have many items of the same type, but most have few products and small quantities. A single representation is therefore inefficient.

We have proposed a set of 64-bit Electronic Product Code (EPC-64) types, which cover the needs of most industries, yet remain consistent with the larger 96-bit EPC versions and compatible with the traditional UCC.EAN standards.

6. REFERENCES

1. **Brock, D. L, "Intelligent Infrastructure – A Method for Networking Physical Objects,"**
   Presentation, MIT Smart World Conference, Apr 2000.
   http://auto-id.mit.edu/whatsnew/download/DB_Smart_World.pdf

2. **"The Networked Physical World – Proposal for Engineering the Next Generation of Computing,**
   Commerce and Automatic-Identification,"
   Auto-ID White Paper, WH-001, Dec 2000.
   http://auto-id.mit.edu/pdf/MIT-.AUTOID-WH-001.pdf

3. **Brock, D. L, "The Electronic Product Code – A Naming Scheme for Physical Objects,"**
   Auto-ID White Paper, WH-002, Jan 2001.
   http://auto-id.mit.edu/pdf/MIT-AUTOID-WH-002.pdf.

4. **Brock, D. L, "The Physical Markup Language – A Universal Language for Physical Objects,"**
   Auto-ID White Paper, WH-003, Feb 2001.
   http://auto-id.mit.edu/pdf/MIT-AUTOID-WH-003.pdf.

5. **The Uniform Code Council (UCC)**
   http://www.uc-council.org

6. **The European Article Numbering (EAN) International**
   http://www.ean.org

7. **Uniform Code Council (UCC), Uniform Product Code (U.P.C.)**
   Symbol Specification Manual, January 1986.
   http://www.uc-council.org/reflib/01302/d36-t.htm.

8. **Guidelines for Supply Chain Identification, Uniform Code**
   Council Publication, August 1999.