INTERNATIONAL STANDARD

ISO/IEC 24724

Second edition
2011-04-01

Information technology — Automatic identification and data capture techniques — GS1 DataBar bar code symbology specification

Technologies de l'information — Techniques automatiques d'identification et de capture des données — Spécifications de la symbologie des codes à barres GS1 DataBar

Reference number
ISO/IEC 24724:2011(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 24724 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 31, *Automatic identification and data capture techniques*.

This second edition cancels and replaces the first edition (ISO/IEC 24724:2006), which has been technically revised.

Introduction

GS1 DataBar was formerly known as “Reduced Space Symbology (RSS)” and is renamed to align with the name of the GS1 organization.

The GS1 DataBar family contains three types of linear symbologies to be used with the GS1 system. The first type has four variations (GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked and GS1 DataBar Stacked Omnidirectional). The stacked variations are two-row symbols. The second type comprises only one variation, namely GS1 DataBar Limited. The third type has two variations: a single row variation (GS1 DataBar Expanded) and a multi-row stacked variation (GS1 DataBar Expanded Stacked). The use of GS1 DataBar is intended to comply with the GS1 application guidelines as defined in the GS1 General Specifications.

GS1 DataBar Omnidirectional and GS1 DataBar Stacked Omnidirectional encode a 14-digit GS1 item identification (often referred to as a Global Trade Item Number, or GTIN) in a linear symbol that can be scanned omnidirectionally by suitably programmed point-of-sale scanners. GS1 DataBar Truncated and GS1 DataBar Stacked encode a 14-digit GS1 item identification in a linear symbol and are not suitable for omnidirectional scanning. GS1 DataBar Limited encodes a 14-digit GS1 item identification with a leading digit of zero or one in a linear symbol for use on small items that will not be scanned at the point-of-sale. GS1 DataBar Expanded encodes GS1 item identification plus supplementary application identifier element strings such as weight and “best before” date in a linear symbol that can be scanned omnidirectionally by suitably programmed point-of-sale scanners.

Any member of the GS1 DataBar family can be printed as a stand-alone linear symbol or as part of a GS1 Composite symbol with an accompanying two-dimensional component printed above the GS1 DataBar linear component.

GS1 DataBar symbols are intended for encoding identification numbers and data supplementary to the identification. The administration of the numbering system by GS1 ensures that identification codes assigned to particular items are unique worldwide and that they and the associated supplementary data are defined in a consistent way. The major benefit for the users of the GS1 system is the availability of uniquely defined identification codes and supplementary data formats for use in their trading transactions.

Information technology — Automatic identification and data capture techniques — GS1 DataBar bar code symbology specification

1 Scope

This International Standard defines the requirements for the GS1 DataBar symbology family. It specifies the characteristics of the GS1 DataBar symbology family, data character encodation, symbol formats, dimensions, print quality requirements, error detection, and decoding algorithms.

For GS1 Composite symbols, ISO/IEC 24723 defines the two-dimensional component.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 646, *Information technology — ISO 7-bit coded character set for information interchange*

ISO 4217, *Codes for the representation of currencies and funds*

ISO/IEC 15416, *Information technology — Automatic identification and data capture techniques — Bar code print quality test specification — Linear symbols*

ISO/IEC 15417, *Information technology — Automatic identification and data capture techniques — Code 128 bar code symbology specification*

ISO/IEC 15420, *Information technology — Automatic identification and data capture techniques — EAN/UPC bar code symbology specification*

ISO/IEC 19762-1, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 1: General terms relating to AIDC*

ISO/IEC 19762-2, *Information technology — Automatic identification and data capture (AIDC) techniques — Harmonized vocabulary — Part 2: Optically readable media (ORM)*

ISO/IEC 24723, *Information technology — Automatic identification and data capture techniques — GS1 Composite bar code symbology specification*

3 Terms, definitions, abbreviated terms and mathematical operators

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 19762-1, ISO/IEC 19762-2 and the following apply.

NOTE For terms which are defined below and in ISO/IEC 19762, the definitions given below apply.

3.1.1
2D component
two-dimensional portion of a GS1 Composite symbol, which encodes supplemental information about an item, such as its lot number or expiration date

3.1.2
AI element string
character string containing an application identifier followed by its associated data field

3.1.3
encodation methods
compaction schemes used by GS1 DataBar Expanded and 2D components to encode commonly used AI element strings in binary strings that are shorter than would be required using general data compaction for the symbology

3.1.4
indicator digit
leading digit of a GTIN-14 item identification number used to differentiate multiple levels of packaging or to indicate a variable measure item

3.1.5
linear component
linear portion of a GS1 Composite symbol, which encodes the primary identification of an item

3.1.6
linkage flag
indicator encoded in a GS1 DataBar or GS1-128 linear component to signal if a 2D component accompanies the linear component

3.1.7
segment
minimum decodable portion of a bar code symbol, consisting, in GS1 DataBar, of a symbol character and its adjacent finder pattern

3.1.8
GS1-128
subset, specified in GS1 General Specifications, of Code 128 as defined in ISO/IEC 15417

3.1.9
voting
decoding technique whereby decoded segment values are saved along with a count of the number of times they have been decoded

NOTE Voting is used for decoding GS1 DataBar by segments such as when used with omnidirectional scanning.

3.2 Abbreviated terms

AI	Application Identifier (see ISO/IEC 15418)
2D	two-dimensional

3.3 Mathematical operators and notational conventions

For the purposes of this document, the following mathematical operators apply.

div	integer division operator which discards the remainder
mod	integer remainder after integer division

The following ISO notational conventions are used.

0,2	A comma between numbers represents a decimal value (e.g. 0,2 equals 2/10) except when used in subscripts or as an (n,k) designation.
12 345	A space between digits indicates factors of a thousand.

4 Symbol description

4.1 Types of GS1 DataBar symbol

The GS1 DataBar symbology consists of the following three types:

First type of GS1 DataBar symbol that has the following four variations:

GS1 DataBar Omnidirectional

GS1 DataBar Truncated

GS1 DataBar Stacked

GS1 DataBar Stacked Omnidirectional

Second type of GS1 DataBar symbol that has the following one variation:

GS1 DataBar Limited

Third type of GS1 DataBar symbol that has the following two variations:

GS1 DataBar Expanded

GS1 DataBar Expanded Stacked

The first type of GS1 DataBar symbol contains four symbol characters in every symbol and has identical character encodation rules and structure for all symbols of this type.

The second type of GS1 DataBar symbol is structurally different than the first type, containing two symbol characters and uses different character encodation rules.

The third type has yet another distinct symbology structure and set of character encodation rules, and can contain a variable number of symbol characters.

GS1 DataBar Omnidirectional, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded and GS1 DataBar Expanded Stacked are designed to be read in segments by omnidirectional scanners.

NOTE Annex J contains a summary of characteristics of the three types of GS1 DataBar symbology types.

4.2 Symbology characteristics

The characteristics of the GS1 DataBar symbology are:

a) Encodable character set:

1) GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional and GS1 DataBar Limited: 0 through 9;

2) GS1 DataBar Expanded and GS1 DataBar Expanded Stacked: a subset of ISO/IEC 646, consisting of the upper and lowercase letters, digits, and 20 selected punctuation characters in addition to the special function character, FNC1.

b) Symbol character structure: different (n,k) symbol characters are used for each type of the symbology, where each symbol character is n modules in width and is composed of k bars and k spaces.

c) Code type: continuous, linear bar code symbology.

d) Maximum numeric data capacity (including implied application identifiers where appropriate, but not including any encoded FNC1 characters):

1) GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional and GS1 DataBar Limited: application identifier “01” plus a 14-digit numeric item identification;

2) GS1 DataBar Expanded and GS1 DataBar Expanded Stacked: 74 numeric or 41 alphabetic characters (see note).

NOTE The GS1 DataBar Expanded data capacity depends on the encodation method. The maximum is 74 digits for (01) + other AIs, the maximum is 70 digits for any AIs, and the maximum is 77 digits for (01) + (392x) + any AIs.

e) Error detection:

1) GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional: mod 79 checksum;

2) GS1 DataBar Limited: mod 89 checksum;

3) GS1 DataBar Expanded and GS1 DataBar Expanded Stacked: mod 211 checksum.

f) Character self-checking: yes.

g) Bidirectionally decodable: yes.

4.3 Summary of additional features

The following is a summary of additional GS1 DataBar symbology features:

a) Data compaction: Each member of the family has data compaction methods optimized for the data strings that they will encode. GS1 DataBar Expanded is optimized for specific combinations of application identifiers that are commonly used.

b) Component linkage: All GS1 DataBar symbols include a linkage flag. If the linkage flag is clear, i.e. equal to 0, then the GS1 DataBar symbol stands alone. If the linkage flag is set, i.e. equal to 1, then a 2D component is associated with the GS1 DataBar family linear component and its separator pattern.

c) GS1-128 emulation: Readers set to the GS1-128 emulation mode transmit the data encoded within the GS1 DataBar family symbol as if the data were encoded in one or more GS1-128 symbols.

4.4 Symbol structure

Each GS1 DataBar symbol contains outside guard patterns, symbol characters, and finder patterns. Every symbol includes a method for error detection.

The guard patterns for the first and third types of GS1 DataBar symbols consist of two one-module wide elements forming either a bar/space or a space/bar pair at each end of the symbol. GS1 DataBar Stacked and GS1 DataBar Expanded Stacked symbols have guard patterns at the ends of each row of the symbol. GS1 DataBar Limited symbols has a different guard pattern requirement that is designed to prevent misreads. See Annex I.1 regarding printing considerations for exterior guard pattern elements.

Every symbol has two or more symbol characters, each with an (n,k) structure. The symbol character values are combined mathematically to form the explicitly encoded data.

The finder pattern is a set of elements selected to be identifiable by the decoder so that the symbol can be recognized and the relative position of the elements can be determined. Each symbol contains one or more finder patterns. The finder patterns also function as the check character and/or segment identifiers.

All GS1 DataBar symbols include a linkage flag. If the flag is set, the GS1 DataBar linear component and its contiguous separator pattern shall be aligned with a 2D component in accordance with ISO/IEC 24723. Normally the GS1 DataBar linear component, its contiguous separator pattern, and the 2D component are printed at the same time, comprising a single GS1 Composite symbol. It is possible however, to preprint a GS1 DataBar linear component with the linkage flag set in anticipation of a subsequent process in which the 2D component is added. Under such circumstances the separator pattern shall be printed with the GS1 DataBar linear component in accordance with ISO/IEC 24723.

Bars and spaces may not be inverted for any form of GS1 DataBar symbol. That is, bars must be dark and spaces must be light. Scanners should not attempt to decode bar and space patterns as if they are inverted because inverted GS1 DataBar symbols can be misread.

5 Symbol requirements for GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked and GS1 DataBar Stacked Omnidirectional

5.1 Basic characteristics of GS1 DataBar Omnidirectional

GS1 DataBar Omnidirectional is a linear symbology capable of encoding 20 000 000 000 000 (2×10^{13}) values. These values are expressed as 14 digits. The first digit is a linkage flag. The following 13 digits are data characters. The 13 data characters plus an implied check digit form a GS1 14-digit item identification number including a leading indicator digit. Values 10 000 000 000 000 and above indicate that the linkage flag is set and therefore a 2D component is present, e.g. value 10 001 234 567 890 encodes item 00012345678905 with the linkage flag equal to 1.

GS1 DataBar Omnidirectional can be scanned and decoded in four segments and then reconstructed. This facilitates omnidirectional scanning. Figure 1 illustrates a stand-alone GS1 DataBar Omnidirectional symbol encoding the value 20012345678909 and having the linkage flag equal to 0.

Figure 1 — GS1 DataBar Omnidirectional symbol representing (01)20012345678909

NOTE The leading (01) is the implied application identifier and is not encoded in the symbol. The last digit, 9, is not directly encoded in the symbol, but is a calculated mod 10 check digit. See Annex A for the check digit calculation. Annex F Encoding examples contains a complete example of encoding a GS1 DataBar Omnidirectional symbol.

5.2 Symbol structure of GS1 DataBar Omnidirectional

A GS1 DataBar Omnidirectional symbol, as shown in Figure 2, consists of eight regions (from left to right) comprising 96 modules:

a) a one module space and one module bar left guard pattern

b) four spaces and four bars with 16 modules comprising symbol character 1, (n,k) = (16,4)

c) three spaces and two bars with 15 modules comprising the left finder pattern

d) four bars and four spaces with 15 modules comprising symbol character 2, (n,k) = (15,4) (right to left)

e) four bars and four spaces with 15 modules comprising symbol character 4, (n,k) = (15,4)

f) three bars and two spaces with 15 modules comprising the right finder pattern (right to left)

g) four spaces and four bars with 16 modules comprising symbol character 3, (n,k) = (16,4) (right to left)

h) a one module space and one module bar right guard pattern

NOTE The symbol character elements are ordered toward the adjacent finder.

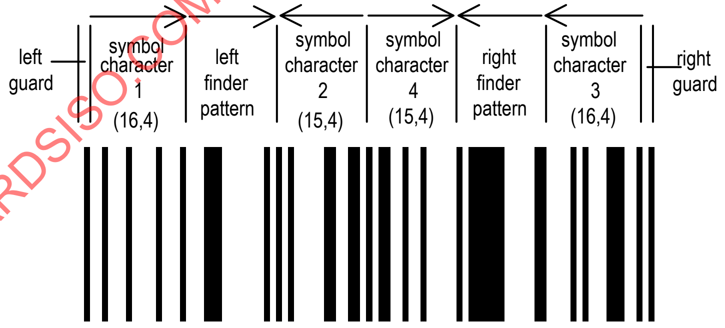
left guard
symbol character 1 (16,4)
left finder pattern
symbol character 2 (15,4)
symbol character 4 (15,4)
right finder pattern
symbol character 3 (16,4)
right guard

Figure 2 — GS1 DataBar Omnidirectional symbol representing (01)04412345678909

The total symbol contains 46 elements (bars and spaces) comprising 96 modules. Table E.1 in Annex E lists all 46 elements of a GS1 DataBar Omnidirectional symbol. A GS1 DataBar Omnidirectional symbol intended for omnidirectional scanning shall have a height greater than or equal to 33X (33 modules).

No quiet zones are required. The first and last elements may appear wider than one module without affecting the symbol if the adjacent background area is the same “color” (light on the left or dark on the right).

5.2.1 Symbol character structure

Each of the four symbol characters has an (n,k) structure. The value of n is 16 for the first and third (outside) symbol characters and 15 for the second and fourth (inside) symbol characters. The value of k is 4.

In Figure 2 the arrows show the ordering of element numbers within each character. The elements of the first and fourth symbol characters are ordered from left to right and the elements of the second and third characters are ordered from right to left, so that the symbol character elements are always ordered toward the adjacent finder.

Each symbol character contains two subsets of odd- and even-numbered elements. The terms odd and even refer to the ordinal number of the elements in each subset. For example the odd-numbered subset consists of the first, third, fifth and seventh elements in each symbol character starting with the element farthest from the adjacent finder pattern. In symbol characters one and two, the odd elements are spaces and the even elements are bars. In symbol characters three and four, the odd elements are bars and the even elements are spaces.

5.2.2 Symbol character value

For each symbol character value, an algorithm assigns a pattern of element widths to the odd and even subsets. The algorithm is given the number of elements, the number of modules, maximum element width, and whether the subset can have all elements wider than one module. Annex B gives a C-language implementation of the GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked and GS1 DataBar Stacked Omnidirectional symbol character element generation algorithm.

5.2.2.1 Outside symbol character values

For the outside symbol characters 1 and 3, the valid even element combinations shall have at least one single-module-wide element. The valid odd element subsets need not have a single-module-wide element. The even element restriction insures that the symbol characters have unique edge-to-similar-edge (bar plus space and space plus bar) module sums.

Table 1 shows the characteristics of the (16,4) subsets, listing the odd and even subset pairs in five groups. Both subsets have an even number of modules. The widest element widths are specified so that the number of modules in a pair of adjacent elements is never greater than nine. The total number of combinations of a (16,4) character is 2 841. The (16,4) symbol character value V_D is calculated by:

$$V_D = (V_{ODD} \times T_{EVEN}) + V_{EVEN} + G_{SUM}$$

where T_{EVEN} is the even subset total value, V_{ODD} is the odd subset value, V_{EVEN} is the even subset value, and G_{SUM} is the sum of the products of values for each previous group in Table 1. To encode a specific symbol character of V_D:

$$V_{ODD} = (V_D - G_{SUM}) \text{ div } T_{EVEN}$$

$$V_{EVEN} = (V_D - G_{SUM}) \text{ mod } T_{EVEN}$$

For example a (16,4) symbol character with the value of 2 315 is to be encoded. From Table 1, the value of the symbol character is in the range of Group 4, so G_{SUM} = 2 015 and T_{EVEN} = 70. Using the above equations:

$$V_{ODD} = (2\,315 - 2\,015) \text{ div } 70 = 300 \text{ div } 70 = 4$$

$$V_{EVEN} = (2\,315 - 2\,015) \text{ mod } 70 = 300 \text{ mod } 70 = 20$$

The symbol character value 2 315 is in Group 4 (see Table 1). The symbol character is comprised of an odd subset with 6 modules and a sequential value of V_{ODD} = 4 out of 10 (range 0 to 9) and an even subset with 10 modules and a sequential value of V_{EVEN} = 20 out of 70 (range 0 to 69). Using the routines in Annex B, the odd element widths are {1 2 2 1} and the even element widths are {1 5 1 3} giving the symbol character element widths of {1 1 2 5 2 1 1 3} as ordered towards the finder pattern (see Figure 2).

Table 1 — Outside symbol character (16,4) characteristics

Value range	Group	Sum of previous groups, G_{SUM}	Odd/even subset modules	Odd/even widest elements	Odd subset total values, T_{ODD}	Even subset total values, T_{EVEN}
0 to 160	1	0	12/4	8/1	161	1
161 to 960	2	161	10/6	6/3	80	10
961 to 2 014	3	961	8/8	4/5	31	34
2 015 to 2 714	4	2 015	6/10	3/6	10	70
2 715 to 2 840	5	2 715	4/12	1/8	1	126

5.2.2.2 Inside symbol character values

For the inside symbol characters 2 and 4, the valid odd element combinations shall have at least one single module wide element. The valid even element subsets need not have a single-module-wide element. The odd element restriction insures that the symbol characters have unique edge-to-similar-edge (bar plus space and space plus bar) module sums.

Table 2 shows the characteristics of the (15,4) subsets, listing the odd and even subset pairs in four groups. The odd subset has an odd number of modules and the even subset has an even number of modules. The widest element widths are specified so that the number of modules in a pair of adjacent elements is never greater than nine. The total number of combinations for a (15,4) character is 1 597. The range of allowed values of the odd subset is restricted so that the innermost element (odd element number 1) will not exceed 4 modules.

Table 2 — Inside symbol character (15,4) characteristics

Value range	Group	Sum of previous groups, G_{SUM}	Odd/even subset modules	Odd/even widest elements	Odd subset total values, T_{ODD}	Even subset total values, T_{EVEN}
0 to 335	1	0	5/10	2/7	4	84
336 to 1 035	2	336	7/8	4/5	20	35
1 036 to 1 515	3	1 036	9/6	6/3	48	10
1 516 to 1 596	4	1 516	11/4	8/1	81	1

The (15,4) symbol character value V_D is calculated by:

$$V_D = (V_{EVEN} \times T_{ODD}) + V_{ODD} + G_{SUM}$$

where T_{ODD} is the odd subset total value, V_{EVEN} is the even subset value, V_{ODD} is the odd subset value, and G_{SUM} is the sum of the products of values for each previous group. To encode a specific symbol character of value V_D:

$$V_{EVEN} = (V_D - G_{SUM}) \text{ div } T_{ODD}$$

$$V_{ODD} = (V_D - G_{SUM}) \text{ mod } T_{ODD}$$

Note that the significance of the even and odd subsets is reversed in these calculations compared to the (16,4) outside symbol characters.

5.2.3 Symbol value

The value of the symbol is formed by combining the values of the left symbol character pairs and the right symbol character pairs. The value of each symbol character pair is formed by combining the values of the outside and inside symbol characters. The symbol character pairs and their range of values are listed in Table 3.

Table 3 — Symbol character pair values

Outside symbol character		Inside symbol character		Symbol character pair	
(n,k)	values ($V_{OUTSIDE}$)	(n,k)	values (V_{INSIDE})	number of values	value range
(16,4)	2 841	(15,4)	1 597	4 537 077	0 to 4 537 076

The symbol character pair value V_{PAIR} is calculated by:

$$V_{PAIR} = (1\,597 \times C_{OUTSIDE}) + C_{INSIDE}$$

where $C_{OUTSIDE}$ and C_{INSIDE} are the symbol character values.

To encode the pair value V_{PAIR} into the outside and inside symbol characters $C_{OUTSIDE}$ and C_{INSIDE}:

$$C_{OUTSIDE} = V_{PAIR} \text{ div } V_{INSIDE}$$

$$C_{INSIDE} = V_{PAIR} \text{ mod } V_{INSIDE}$$

For example, if the symbol character pair value V_{PAIR} is 1 971 265, then $C_{OUTSIDE}$ and C_{INSIDE} are:

$$C_{OUTSIDE} = 1\,971\,265 \text{ div } 1\,597 = 1\,234$$

$$C_{INSIDE} = 1\,971\,265 \text{ mod } 1\,597 = 567$$

The symbol value is calculated by combining the values of the left and right symbol character pair values. The calculation is:

$$V_{SYMBOL} = (4\,537\,077 \times V_{LPAIR}) + V_{RPAIR}$$

where V_{SYMBOL} is the symbol value and V_{LPAIR} and V_{RPAIR} are the left and right symbol character pair values.

To encode the symbol value V_{SYMBOL} into the left and right symbol character pairs V_{LPAIR} and V_{RPAIR}:

$$V_{LPAIR} = V_{SYMBOL} \text{ div } 4\,537\,077$$

$$V_{RPAIR} = V_{SYMBOL} \text{ mod } 4\,537\,077$$

For example, if the symbol V_{SYMBOL} is 1 234 567 890, Then the value of the left pair V_{LPAIR} and the value of the right pair V_{RPAIR} are:

$$V_{LPAIR} = 1\,234\,567\,890 \text{ div } 4\,537\,077 = 272$$

$$V_{RPAIR} = 1\,234\,567\,890 \text{ mod } 4\,537\,077 = 482\,946$$

Combining the values of the symbol characters generates 20 585 067 703 929 values, however, only the first 20 000 000 000 000 values (0 to 19 999 999 999 999) are used. The high-order digit is the 2D component linkage flag: 0 for a stand-alone GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked, GS1 or GS1 DataBar Stacked Omnidirectional and 1 if a 2D component adjoins the GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked or GS1 DataBar Stacked Omnidirectional primary symbol. This flag is stripped from the remaining 13 digits to form the item identification. An implied mod-10 check digit is calculated and added to the end to form the GTIN-14 identification number. A leading application identifier prefix 01 is added to the transmitted data, immediately after the mandatory transmitted symbology identifier,]e0 or]C1.

5.2.4 Finder patterns

The symbol has two finder patterns that also encode the symbol checksum. Each finder pattern can encode nine values. The finder patterns are positioned between the first and second symbol characters and between the fourth and third symbol characters. Since a finder pattern is adjacent to all four symbol characters, the symbol can be scanned in four segments. Each segment will contain a symbol character and a finder pattern.

5.2.4.1 Finder pattern structure

The two finder patterns each consist of 5 elements comprising 15 modules. The left finder pattern starts and ends with a space and the right finder pattern starts and ends with a bar. Finder pattern elements are numbered from the outside to the inside of the symbol as shown in Figure 2.

The sum of the modules in the elements 2 and 3 is 10 to 12, while the sum of the modules in elements 4 and 5 is 2. The ratio of the wide element pair (2 and 3) to the total width of the four adjacent elements (2 through 5) is in the range of 10:12 to 12:14. This ratio is used for the first step in the recognition logic for the finder pattern. Table 4 lists the finder pattern element widths for the nine encoded values.

Table 4 — Finder pattern values and element widths

Finder Value	Element Widths (numbered from outside to inside)				
	1	2	3	4	5
0	3	8	2	1	1
1	3	5	5	1	1
2	3	3	7	1	1
3	3	1	9	1	1
4	2	7	4	1	1
5	2	5	6	1	1
6	2	3	8	1	1
7	1	5	7	1	1
8	1	3	9	1	1

Finder pairs 8,0 and 0,8 are not used as 0 and 8 can be transformed into a reverse of the other with a single 1-X edge error. The remaining 79 possible pairs encode a mod 79 checksum value.

5.2.4.2 Checksum calculation

The two finder pattern values, C_{LEFT} and C_{RIGHT}, each have nine possible values. Finder pattern value pairs 0,8 and 8,0 are not valid. This leaves a total of $(9 \times 9) - 2$ or 79 combinations. The checksum value is equal to the mod 79 residue of the weighted sum of the widths of the elements in the symbol characters.

The mod 79 checksum value is calculated by:

$$(W_{1,1}E_{1,1} + W_{1,2}E_{1,2} +\ldots+ W_{1,8}E_{1,8} + W_{2,1}E_{2,1} +\ldots+ W_{4,8}E_{4,8}) \bmod 79$$

where $W_{N,M}E_{N,M}$ is the product of the weight for symbol character N at ordinal element position M, from Table 5, and the module width of element M in symbol character N. The weights are successive powers of three mod 79.

Table 5 — Checksum calculation element weights

Data Character	Symbol character element ordinal positions							
	1	2	3	4	5	6	7	8
1	1	3	9	27	2	6	18	54
2	4	12	36	29	8	24	72	58
3	16	48	65	37	32	17	51	74
4	64	34	23	69	49	68	46	59

Encoding the two finder pattern values uses the following procedure:

```
temp = check value
if temp is greater than or equal to 8, then temp = temp + 1
if temp is greater than or equal to 72, then temp = temp + 1
$C_{LEFT}$ = temp div 9
$C_{RIGHT}$ = temp mod 9
```

See Annex F.1 for a complete example of checksum calculation and check character selection.

5.2.4.3 Decoding the finder pattern

The finder pattern is first identified by comparing the total width of four adjacent elements to the width of their leftmost or rightmost pair of elements. For the finder pattern, the ratio is within the range of 12:9,5 to 14:12,5. The left and right finder patterns are differentiated by their dark/light inversion.

The finder pattern and check for a valid symbol character to finder pattern pitch ratio should verify that a valid GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked or GS1 DataBar Stacked Omnidirectional symbol quarter segment has been scanned.

5.2.5 Reference decode algorithm

Bar code reading systems are designed to read imperfect symbols to the extent that practical algorithms permit. This section describes the reference decode algorithm used in the computation of the decodability value described in ISO/IEC 15416 for measuring symbol quality.

The algorithm contains the following steps to decode the symbol:

a) Find a segment by looking both left to right and right to left for a four-element sequence with the ratio:

for left to right:
$9{,}5:12 \le ((\text{element1} + \text{element2}) : (\text{element1} + \text{element2} + \text{element3} + \text{element4})) \le 12{,}5:14$

or for right to left:
$9{,}5:12 \le ((\text{element3} + \text{element4}) : (\text{element1} + \text{element2} + \text{element3} + \text{element4})) \le 12{,}5:14$

This ratio identifies the second through fifth elements of the finder.

Decode the finder pattern using the method in steps c) 1 to c) 3 to find the normalized edge-to-similar-edge values E1 and E2 from the pitch, p, the sum of the first four element widths of the finder. Verify that the values E1 and E2 correspond to a valid GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked or GS1 DataBar Stacked Omnidirectional finder pattern.

NOTE The element numbers are ordered from outside to inside, and thus element one of the left finder occurs on the left of the finder pattern and element one of the right finder pattern is on the right of the right finder pattern. See Figure 2.

b) Determine the direction and black-white inversion of the finder. Using the finder pattern and orientation, determine which (n,k) pattern, (16,4) or (15,4), is appropriate for the adjacent symbol character along with its leading element color, black or white.

c) For each adjacent symbol character with a (16,4) structure, decode it as follows:

1) Obtain the seven width measurements p, e_1, e_2, e_3, e_4, e_5, and e_6 (Figure 3).

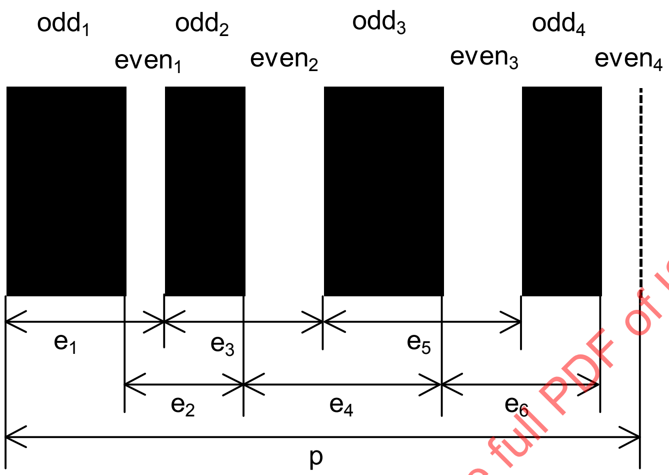

Figure 3 — Decode measurements

NOTE The diagram shows the first element as the left black element, but the symbol characters are also left-to-right mirrored and/or dark-light inverted.

2) Convert measurements e_1, e_2, e_3, e_4, e_5, and e_6 to normalized values E_1, E_2, E_3, E_4, E_5, and E_6 which will represent the integral module width (E_i) of these measurements. The following method is used for the i-th value.

If $1{,}5p/16 \leq e_i < 2{,}5p/16$, then $E_i = 2$
If $2{,}5p/16 \leq e_i < 3{,}5p/16$, then $E_i = 3$
If $3{,}5p/16 \leq e_i < 4{,}5p/16$, then $E_i = 4$
If $4{,}5p/16 \leq e_i < 5{,}5p/16$, then $E_i = 5$
If $5{,}5p/16 \leq e_i < 6{,}5p/16$, then $E_i = 6$
If $6{,}5p/16 \leq e_i < 7{,}5p/16$, then $E_i = 7$
If $7{,}5p/16 \leq e_i < 8{,}5p/16$, then $E_i = 8$
If $8{,}5p/16 \leq e_i < 9{,}5p/16$, then $E_i = 9$

Otherwise the character is in error.

3) Determine the normalized element widths from the E values. The last element is assigned the remaining modules rather that being calculated from the E values. The set of valid element widths is the only solution that has no element widths less than one module and has at least one even element that is one module wide. For example the Figure 3 values E_1 through E_6 are {4 3 4 5 5 4}. The possible derived element sets could be calculated as {4 0 3 1 4 1 3 0} (note the 0 width elements), {3 1 2 2 3 2 2 1}, {2 2 1 3 2 3 1 2} (note there are no single-module even-numbered elements), or {1 3 0 4 1 4 0 3} (note the 0 width elements and no single-module even-numbered elements). Only the eight element widths {3 1 2 2 3 2 2 1} satisfy the requirements and therefore are selected as the element widths. If no set of derived element widths is valid, then the character is in error. Annex G gives a C-language implementation of this element width decoding algorithm.

4) Determine the values of the odd and even subsets from the program in Annex B.

5) Calculate the symbol character value from the odd and even subset values.

6) Calculate and store the weighted element widths for the checksum calculation.

d) For each adjacent symbol character with a (15,4) structure, decode it as follows:

1) Calculate the seven width measurements p, e_1, e_2, e_3, e_4, e_5, and e_6 (as shown in Figure 3).

2) Convert measurements e_1, e_2, e_3, e_4, e_5, and e_6 to normalized values E_1, E_2, E_3, E_4, E_5, and E_6 which will represent the integral module width (E_i) of these measurements. The following method is used for the i-th value.

If $1,5p/15 \leq e_i < 2,5p/15$, then $E_i = 2$
If $2,5p/15 \leq e_i < 3,5p/15$, then $E_i = 3$
If $3,5p/15 \leq e_i < 4,5p/15$, then $E_i = 4$
If $4,5p/15 \leq e_i < 5,5p/15$, then $E_i = 5$
If $5,5p/15 \leq e_i < 6,5p/15$, then $E_i = 6$
If $6,5p/15 \leq e_i < 7,5p/15$, then $E_i = 7$
If $7,5p/15 \leq e_i < 8,5p/15$, then $E_i = 8$
If $8,5p/15 \leq e_i < 9,5p/15$, then $E_i = 9$

Otherwise the character is in error.

3) Calculate the (15,4) symbol character value using the steps c) 3 through c) 6 above.

e) Decode the value of the finder pattern using the normalized element width method as follows:

1) Find p, the sum of elements e1, e2, e3, and e4 of the finder pattern.

2) Convert measurements e_1, e_2, e_3, and e_4 to normalized values E_1, E_2, E_3, and E_4,which will represent the integral module width (E_i) of these measurements. The following method is used for the i-th value.

If $1,5p/14 \leq e_i < 2,5p/14$, then $E_i = 2$
If $2,5p/14 \leq e_i < 3,5p/14$, then $E_i = 3$
If $3,5p/14 \leq e_i < 4,5p/14$, then $E_i = 4$
If $4,5p/14 \leq e_i < 5,5p/14$, then $E_i = 5$
If $5,5p/14 \leq e_i < 6,5p/14$, then $E_i = 6$
If $6,5p/14 \leq e_i < 7,5p/14$, then $E_i = 7$
If $7,5p/14 \leq e_i < 8,5p/14$, then $E_i = 8$
If $8,5p/14 \leq e_i < 9,5p/14$, then $E_i = 9$

3) Look up the value of the finder pattern for E1, E2, E3, and E4 in Table 4.

f) When all four symbol characters and two finder patterns have been decoded, verify that the two finder pattern values belong to the set of 79 valid pairs. Verify that the mod 79 checksum value calculated from the two finder patterns matches the sum of the weighted element widths (see 5.2.4.2) in the symbol characters mod 79.

g) Calculate the linkage flag and item identification number from the four symbol characters.

h) In addition, perform such other secondary checks on acceleration, absolute timing, dimensions, etc., as are deemed prudent and appropriate considering the specific reading device and intended application environment.

In the design of practical scanners for GS1 DataBar, security measures such as redundant capture of data are required. Annex H contains additional symbol decoding considerations which should be followed to minimize misreads.

5.3 Variations for specific applications

5.3.1 GS1 DataBar Truncated

GS1 DataBar Truncated (shown in Figure 4) is structured and encoded in the same way as the standard GS1 DataBar Omnidirectional format, except its height is reduced to a 13X minimum. It may be used for small items, instead of GS1 DataBar Limited, when an indicator digit greater than one is required. It may also be used when the four-column 2D component is desired in order to minimize the height of a GS1 Composite symbol.

GS1 DataBar Truncated is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional point-of-sale scanners.

Figure 4 — GS1 DataBar Truncated symbol representing (01)00012345678905

The overall size of this format is 96X wide by a minimum of 13X high.

5.3.2 Two-row variations

A truncated variation and an omnidirectional variation are defined. GS1 DataBar Stacked is a GS1 DataBar Omnidirectional symbol that has the two halves of the symbol reduced in height and stacked into two rows. The top row consists of the left half of the symbol with a 1X bar and 1X space guard pattern added to the right of the row. The bottom row starts on the left with a 1X bar and 1X space guard pattern followed by the right half of the symbol. GS1 DataBar Stacked Omnidirectional is similar, but with rows which are full height, and includes a three module tall separator pattern.

5.3.2.1 GS1 DataBar Stacked

GS1 DataBar Stacked (shown in Figure 5) is a truncated two-row variation. For comparison, the Figure 5 symbol encodes the same data as Figure 4.

Figure 5 — GS1 DataBar Stacked symbol representing (01)00012345678905

The top row is 5X high and the bottom row is 7X high with a 1X (minimum) high separator pattern between the two rows. The overall size of this format is 50X wide by 13X high.

The separator pattern starts with a one-module-wide space on the left. Wherever the top row and bottom row modules vertically adjacent to a separator pattern module have the same color, the separator module shall be the opposite color to the adjacent row modules. This results in a separator space between two vertically adjacent bar segments and a separator bar between two vertically adjacent space segments.

Wherever the top and bottom row modules vertically adjacent to a separator module have different colors, the separator module shall be an opposite color to the separator module to the left. This results in a pattern of alternating one-module wide bars and spaces in a region with complementary top and bottom row colors. As an exception, the first four and last four modules of the separator row are always light, overriding the complementary and alternating pattern. (See Figure 5)

GS1 DataBar Stacked may be used for small items instead of GS1 DataBar Limited when the available space is too narrow for GS1 DataBar Limited. Moreover, the narrower width of GS1 DataBar Stacked might allow for a larger X dimension and potentially higher print quality. However, GS1 DataBar Limited or GS1 DataBar Truncated should be used in preference to the stacked format whenever space permits without reducing X dimension, as they are easier to scan with a wand or linear scanner.

GS1 DataBar Stacked is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional point-of-sale scanners.

5.3.2.2 GS1 DataBar Stacked Omnidirectional

GS1 DataBar Stacked Omnidirectional (shown in Figure 6) is a full height two-row variation. A 3X (minimum) high separator pattern shall separate the symbol rows and consists of three 1X (minimum) high rows.

Figure 6 — GS1 DataBar Stacked Omnidirectional symbol representing (01)00034567890125

The upper row of the separator pattern is the complement of the bars and spaces in the top symbol row, except for the ends and 13 modules under the finder pattern elements 1, 2 and 3. These 13 modules are light under the adjacent finder bars and alternating dark, light, dark, etc. under the adjacent finder spaces.

The middle row consists of alternating light and dark modules except for the ends.

The lower row is the complement of the bottom symbol row, except for the ends and 13 modules above the finder pattern elements one, two and three (right to left). These 13 modules are light over the adjacent finder bars and alternating dark, light, dark, etc. (left to right) over the adjacent finder space. The single dark module that occurs in the 13 modules over finder value 3 is shifted one module to the right so that it is over the start of the three module-wide finder bar.

The first four and last four modules of the separator rows are always light, overriding the complementary alternating pattern.

Each row is 33X high minimum with a 3X high separator pattern between the two rows. The overall size of this format is 50X wide by 69X high minimum.

GS1 DataBar Stacked Omnidirectional can be used instead of GS1 DataBar Omnidirectional for omnidirectional scanning applications where the different aspect ratio is needed.

6 Symbol requirements for GS1 DataBar Limited

6.1 Basic characteristics

GS1 DataBar Limited is a linear symbology capable of encoding 4 000 000 000 000 (4×10^{12}) numbers (see 6.2.3). The entire GS1 number set with indicator digits 1 and 0 can be encoded in addition to a flag to provide linkage to a 2D component.

GS1 DataBar Limited is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional point-of-sale scanners. Figure 7 illustrates a GS1 DataBar Limited symbol.

Figure 7 — GS1 DataBar Limited symbol representing (01)15012345678907

6.2 Symbol structure

A GS1 DataBar Limited symbol shown in Figure 8 consists of five regions (from left to right) comprising 79 modules:

a) a one module space and one module bar left guard pattern;

b) seven spaces and seven bars with 26 modules comprising the left symbol character, (n,k) = (26,7);

c) seven spaces and seven bars with 18 modules comprising the check character, (n,k) = (18,7);

d) seven spaces and seven bars with 26 modules comprising the right symbol character, (n,k) = (26,7);

e) a three element guard pattern consisting of a one module space followed by a one module bar followed by a five module space.

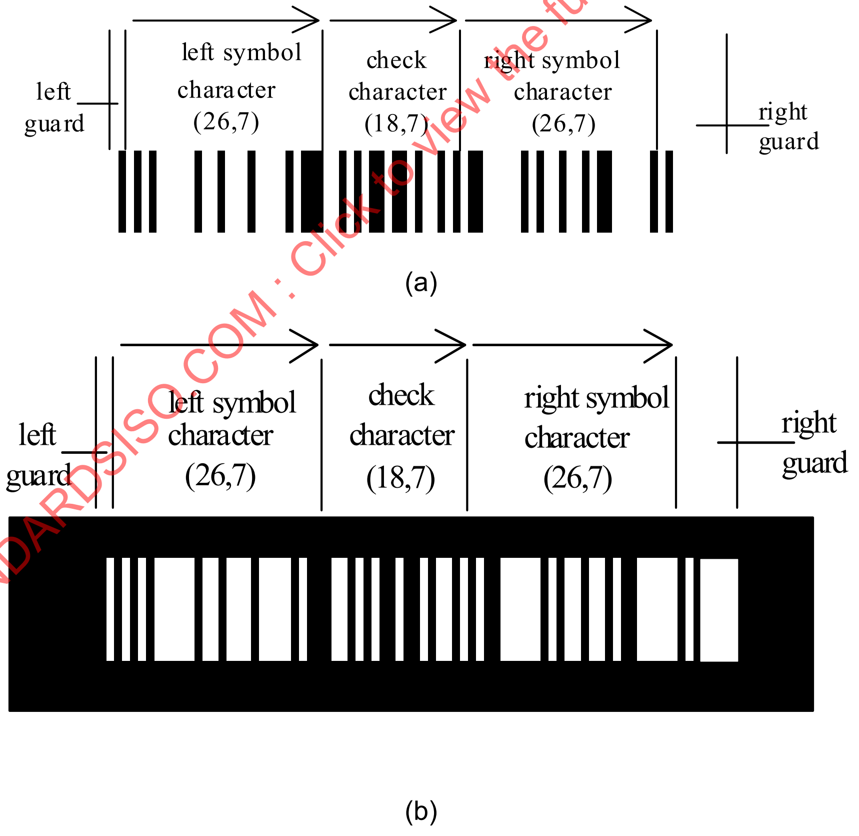
left guard
left symbol character (26,7)
check character (18,7)
right symbol character (26,7)
right guard
(a)
left guard
left symbol character (26,7)
check character (18,7)
right symbol character (26,7)
right guard
(b)

Figure 8 — (a) GS1 DataBar Limited symbol representing (01)00312345678906
(b) the same symbol on a dark background. Notice the trailing space in the right guard pattern

The total symbol contains 47 elements comprising 79 modules. Table E.2 in Annex E lists all the elements in a GS1 DataBar Limited symbol. The minimum height shall be 10X.

No quiet zones are required, however while each light module on both ends of the Limited symbol may look like a quiet zone, each differs from a quiet zone in that the reference decode algorithm must check for these guard bar patterns in order to avoid misreading a UPC-A symbol as a GS1 DataBar Limited symbol. See Annex H. The leading and trailing space elements may blend into the background of the symbol if that background is the same color as the spaces in the symbol.

6.2.1 Symbol character structure

The two symbol characters have an (n,k) structure. The value of n is 26 and the value of k is 7.

The elements of the two symbol characters are ordered from left to right as shown in Figure 8.

Each symbol character contains two subsets of odd- and even-numbered elements. The terms odd and even refer to the ordinal number of the elements in each subset. The odd elements are spaces and the even elements are bars. For example the odd-numbered subset consists of the first, third, fifth, seventh, ninth, eleventh, and thirteenth elements in each symbol character starting with the leftmost element. The subsets of seven elements may contain from a minimum of seven to a maximum of 19 modules. Both the subsets contain an odd number of modules. The sum of the number of modules in the odd- and even-numbered subsets of a symbol character is equal to 26.

The odd and even element subsets are assigned sequential values by an algorithm that encodes values to patterns of element widths. The algorithm is given the number of elements, the number of modules, maximum element width, and whether the subset can have all elements wider than one module.

6.2.2 Symbol character value

For each symbol value, an algorithm assigns a pattern of element widths to the odd and even subsets. The algorithm is given the number of elements, the number of modules, maximum element width, and whether the subset can have all elements wider than one module. Annex B gives a C-language implementation of the GS1 DataBar Limited symbol character element generation algorithm.

The valid even element subsets have at least one single-module element whereas the odd element subsets can have all elements wider than one module. The even element restriction insures that the symbol characters have unique edge-to-similar-edge (bar plus space and space plus bar) module sums.

Table 6 shows the characteristics of the (26,7) subsets, listing the odd and even subset pairs in seven groups. Both subsets have an odd number of modules. The widest element widths are specified so that the number of modules in a pair of adjacent elements is never greater than nine. The total number of combinations for a symbol character is 2 013 571.

Table 6 — Symbol character (26,7) characteristics

Value range	Group	Sum of previous groups, G_{SUM}	Odd/even subset modules	Odd/even widest elements	Odd subset total values, T_{ODD}	Even subset total values, T_{EVEN}
0 to 183 063	1	0	17/9	6/3	6 538	28
183 064 to 820 063	2	183 064	13/13	5/4	875	728
820 064 to 1 000 775	3	820 064	9/17	3/6	28	6 454
1 000 776 to 1 491 020	4	1 000 776	15/11	5/4	2 415	203
1 491 021 to 1 979 844	5	1 491 021	11/15	4/5	203	2 408
1 979 845 to 1 996 938	6	1 979 845	19/7	8/1	17 094	1
1 996 939 to 2 013 570	7	1 996 939	7/19	1/8	1	16 632

The symbol character value V_D is calculated by:

$$V_D = (V_{ODD} \times T_{EVEN}) + V_{EVEN} + G_{SUM}$$

where T_{EVEN} is the even subset total value, V_{ODD} is the odd subset value, V_{EVEN} is the even subset value, and G_{SUM} is the sum of the products of values for each previous group. To encode a specific symbol character of value V_D:

$$V_{ODD} = (V_D - G_{SUM}) \text{ div } T_{EVEN}$$

$$V_{EVEN} = (V_D - G_{SUM}) \bmod T_{EVEN}$$

For example a symbol character with the value of 917 879 is to be encoded. From Table 6, the value of the symbol character is in range of Group 3, so G_{SUM} = 820 064 and T_{EVEN} = 6 454. Using the above equations:

$$V_{ODD} = (917\,879 - 820\,064) \text{ div } 6\,454 = 97\,815 \text{ div } 6\,454 = 15$$

$$V_{EVEN} = (917\,879 - 820\,064) \bmod 6\,454 = 97\,815 \bmod 6\,454 = 1\,005$$

Using the Annex B algorithm, the symbol character from Group 3 (see Table 6) has an odd subset with 9 modules and a sequential value of 15 out of 28 (range 0 to 27) and an even subset with 17 modules and sequential value of 1 005 out of 6 454 (range 0 to 6 453). The odd element widths are {1 2 1 1 1 1 2} and the even element widths are {1 2 3 5 1 2 3} giving the symbol character element widths of {1 1 2 2 1 3 1 5 1 1 1 2 2 3} as ordered from left to right.

6.2.3 Symbol value

The symbol value is calculated by combining the left and right symbol character values. The calculation is:

$$V_{SYMBOL} = (2\,013\,571 \times V_{DLEFT}) + V_{DRIGHT}$$

where V_{SYMBOL} is the symbol value and V_{DLEFT} and V_{DRIGHT} are the left and right symbol character values.

To encode the symbol value V_{SYMBOL} into the left and right symbol characters V_{DLEFT} and V_{DRIGHT}:

$$V_{DLEFT} = V_{SYMBOL} \text{ div } 2\,013\,571$$

$$V_{DRIGHT} = V_{SYMBOL} \bmod 2\,013\,571$$

Combining the values of the symbol characters generates 4 054 468 172 041 values, however only 4 000 000 000 000 values are used. These are the values 0 to 1 999 999 999 999 and 2 015 133 531 096 to 4 015 133 531 095. The two ranges are specified in such a way that the existence of a 2D component can be determined by the number of modules in each subset of the left symbol character without decoding the right symbol character. (The stand-alone GS1 DataBar Limited symbols have left character values of 0 to 993 260, while within GS1 Composite symbols, GS1 DataBar Limited symbols have left character values of 1 000 776 to 1 994 036.)

The second range of values, 2 015 133 531 096 and above, indicate that the linkage flag is set and a 2D component accompanies the GS1 DataBar Limited symbol. The primary data value is then calculated by subtracting 2 015 133 531 096 from the GS1 DataBar Limited value. The remainder has values 0 to 1 999 999 999 999 that is identical to the first range of values and forms the primary item identification.

The values 0 to 1 999 999 999 999 represent the first 13 digits of the GTIN-14 primary item identification. The indicator digit has two possible values of zero and one. An implied mod 10 check digit is calculated and added to the end of the transmitted data to form the GTIN-14 identification number. A leading application identifier prefix 01 is added to the transmitted data, immediately after the mandatory transmitted symbology identifier,]e0 or]C1.

6.2.4 Check character

The symbol has a check character that also serves as the symbol finder pattern. The check character is positioned between the two symbol characters.

6.2.4.1 Check character structure

The (18,7) check character encodes 89 values, 0 to 88. Each character consists of 9 modules forming the 7 spaces and 9 modules forming the 7 bars. The check character patterns are selected to exclude patterns that are the same as shifted and/or left-to-right mirrored 14 element patterns appearing near the middle of a symbol. Annex C lists the check character element widths for the 89 encoded values.

6.2.4.2 Check character calculation

The value of the check character is the mod 89 residue of the weighted sum of the widths of the elements in the symbol characters.

The mod 89 check value is calculated by:

$$(W_{1,1}E_{1,1} + W_{1,2}E_{1,2} + \ldots + W_{1,14}E_{1,14} + W_{2,1}E_{2,1} + \ldots + W_{2,14}E_{2,14}) \bmod 89$$

where $W_{N,M}E_{N,M}$ is the product of the weight for symbol character N at ordinal element position M, from Table 7, and the module width of element M in symbol character N. The weights are successive powers of three mod 89.

Table 7 — Check character calculation element weights

Data Character	Element Weights (numbered from the left)													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	3	9	27	81	65	17	51	64	14	42	37	22	66
2	20	60	2	6	18	54	73	41	34	13	39	28	84	74

Annex F.2 contains an example of encoding a GS1 DataBar Limited symbol.

6.2.5 Finder pattern

The finder pattern is the ratio of the pitches of the symbol characters to the check character. It is identified by the nominal ratio of 26:18:26 in the pitch of the 14 elements of the check character versus the 14 elements of the two adjacent symbol characters. In addition, the symbol is identified by the check character having a valid check character space/bar pattern. Reversed and/or offset check character patterns that may appear in valid symbols have been removed from the set of check characters.

6.2.6 Reference decode algorithm

Bar code reading systems are designed to read imperfect symbols to the extent that practical algorithms permit. This section describes the reference decode algorithm used in the computation of the decodability value described in ISO/IEC 15416 for measurement of symbol quality.

The algorithm contains the following steps to decode each bar coded character:

a) Find the symbol by looking for three fourteen-element sequences with the sequence width ratio of (26 ± 1,5) : 18 : (26 ± 1,5) (plus an allowance for acceleration if required for the scanning device).

b) Verify that the center sequence is a valid finder pattern. Using steps c.1 and c.2 below (with the exception that the p divisor is 18, not 26, in step c.2), determine the element widths. Look up the element widths in the table in Annex C.

c) Decode the two symbol characters as follows:

1) Obtain the thirteen width measurements p, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, and e_{12} (Figure 9).

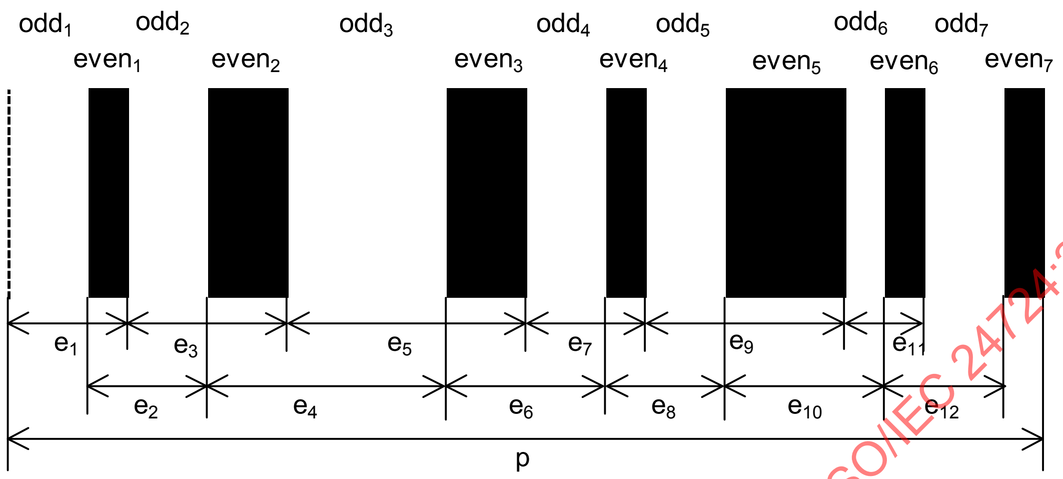

Figure 9 — Decode measurements

2) Convert measurements e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, and e_{12} to normalized values E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9, E_{10}, E_{11}, and E_{12} which will represent the integral module width (E_i) of these measurements. The following method is used for the i-th value.

If $1{,}5p/26 \le e_i < 2{,}5p/26$, then $E_i = 2$
If $2{,}5p/26 \le e_i < 3{,}5p/26$, then $E_i = 3$
If $3{,}5p/26 \le e_i < 4{,}5p/26$, then $E_i = 4$
If $4{,}5p/26 \le e_i < 5{,}5p/26$, then $E_i = 5$
If $5{,}5p/26 \le e_i < 6{,}5p/26$, then $E_i = 6$
If $6{,}5p/26 \le e_i < 7{,}5p/26$, then $E_i = 7$
If $7{,}5p/26 \le e_i < 8{,}5p/26$, then $E_i = 8$
If $8{,}5p/26 \le e_i < 9{,}5p/26$, then $E_i = 9$

Otherwise the character is in error.

3) Determine the normalized element widths from the E values. The last element is assigned the remaining modules rather than being calculated from the E values. The set of valid element widths is the only solution that has no element widths less than one module and has at least one even element that is one module wide. For example the Figure 9 values E_1 through E_{12} are {3 3 4 6 6 4 3 3 5 4 2 3}. The possible derived element sets are {1 2 1 3 3 3 1 2 1 4 0 2 1 2}, {2 1 2 2 4 2 2 1 2 3 1 1 2 1}, or {3 0 3 1 5 1 3 0 3 2 2 0 3 0}. Only the fourteen element widths {2 1 2 2 4 2 2 1 2 3 1 1 2 1} satisfy the valid element width requirements and therefore are selected as the element widths. If no set of derived element widths is valid, then the character is in error. Annex G gives a C-language implementation of this element width decoding algorithm.

4) Determine the values of the odd and even subsets from the program in Annex B.

d) When the two symbol characters and the finder pattern have been decoded, verify that the mod 89 check character value from the finder pattern matches the sum of the weighted element widths in the symbol characters mod 89.

e) Calculate the linkage flag and item identification number from the two symbol characters.

f) In addition, perform such other secondary checks on acceleration, absolute timing, dimensions, etc., as are deemed prudent and appropriate considering the specific reading device and intended application environment.

g) Confirm that all of the following are true:

   1) the distance from the beginning of the symbol (i.e. the left edge of the guard space) to the right edge of the first bar (i.e. the second element of the guard pattern) is greater than (3/52)p of the first symbol character (i.e. >1,5 modules)

   2) the distance from the leading edge of the guard bar to the leading edge of the first bar of the left symbol character is equal to ((N+1)/26)p +/- 0,5 modules, where N equals the normalized E value of the first space as determined in clause 6.2.6 step c) 3) (i.e. the guard bar is nominally one module wide)

   3) if there are at least 10 elements preceding and including the first bar (numbered from the farthest bar to the left = 10 to the first bar = 1) elements one through eight do not decode as two valid even parity UPC-A characters (according to ISO/IEC 15420) with nominal X dimension within 25% of (1/26)p (i.e. 0,25 of the nominal module size of the left DataBar Limited Symbol Character).

   If any of 1, 2, 3 above are not true, reject the GS1 DataBar Limited decode on the current scan.

h) confirm that the distance from the left edge of the right guard pattern to the right edge of the guard bar is equal to (1/13)p of the right symbol character (+/- 0,5 modules) and confirm that the distance from the left edge of the trailing guard bar to the end of the symbol (i.e. the end of the trailing space) is greater than (5/26)p (i.e. 5 modules).

IMPORTANT — Scanners shall perform steps equivalent to step g) and h) above in order to prevent misread of certain UPC-A symbols as GS1 DataBar Limited. See Annex H.5 for explanation.

In the design of practical scanners for GS1 DataBar, security measures such as redundant capture of data are required. Annex H contains additional symbol decoding considerations which should be followed to minimize misreads.

7 Symbol requirements for GS1 DataBar Expanded and GS1 DataBar Expanded Stacked

7.1 Basic characteristics of GS1 DataBar Expanded

GS1 DataBar Expanded is a variable length linear symbology capable of encoding up to 74 numeric or 41 alphabetic characters of AI element string data internally represented as a binary number. GS1 DataBar Expanded can be used to encode primary and supplementary data intended for use in retail point-of-sale and other applications where the scanners and application software have been appropriately programmed.

GS1 DataBar Expanded can be scanned and decoded in up to 22 segments and then reconstructed. This facilitates omnidirectional scanning. Figure 10 illustrates a GS1 DataBar Expanded symbol.

Figure 10 — GS1 DataBar Expanded symbol representing (01)98898765432106(3202)012345(15)991231

7.2 Symbol structure

7.2.1 Overall symbol structure

The first symbol character in every symbol is the check character that encodes the checksum and symbol length. The example symbol in Figure 11 contains one check character and five symbol characters.

GS1 DataBar Expanded symbols are constructed as a sequence of triplets, each consisting of a finder pattern between two symbol characters. If there are an odd number of symbol characters a finder pattern follows the last symbol character. The odd-numbered characters have element order from left to right whereas the even-numbered characters have element order from right to left. Symbol characters 1, 2, 5, 6, 9, 10, etc. have a space for element one (the element farthest from the adjacent finder pattern) while symbol characters 3, 4, 7, 8, 11, 12, etc. have a bar for element one. The left and right guard patterns always start and end the symbol, or start and end each row in a stacked symbol. The six symbol character GS1 DataBar Expanded symbol shown in Figure 11 consists of eleven regions (from left to right):

a) a one module space and one module bar left guard pattern

b) four spaces and four bars with 17 modules comprising the check character, (n,k) = (17,4)

c) three spaces and two bars with 15 modules comprising finder pattern A1

d) four bars and four spaces with 17 modules comprising symbol character 1, (n,k) = (17,4) (right to left)

e) four bars and four spaces with 17 modules comprising symbol character 2, (n,k) = (17,4)

f) three bars and two spaces with 15 modules comprising finder pattern B2

g) four spaces and four bars with 17 modules comprising the symbol character 3, (n,k) = (17,4) (right to left)

h) four spaces and four bars with 17 modules comprising symbol character 4, (n,k) = (17,4)

i) three spaces and two bars with 15 modules comprising finder pattern B1

j) four bars and four spaces with 17 modules comprising the symbol character 5, (n,k) = (17,4) (right to left)

k) a one module bar and one module space right guard pattern

NOTE The symbol character elements are ordered in sequence toward the adjacent finder pattern.

The symbol in Figure 11 has 67 elements comprising 151 modules. Table E.3 in Annex E lists all the 67 elements of the GS1 DataBar Expanded symbol in Figure 11. The minimum height shall be 34 modules.

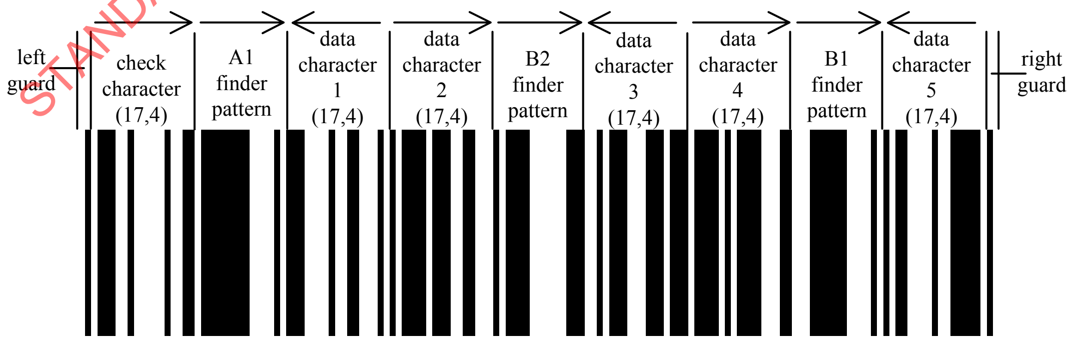

Figure 11 — GS1 DataBar Expanded symbol representing (01)90012345678908(3103)001750

No quiet zones are required. The first and last elements may appear wider than one module if the adjacent background area is the same “color” (light or dark) as the outside guard pattern element.

7.2.2 Symbol character structure

Each data or check character has an (n,k) structure. The value of n is 17 and the value of k is 4.

The elements of the first, third, and other odd-numbered symbol character are ordered from left to right. The elements of the second, fourth, and other even-numbered symbol characters are ordered from right to left, so that the symbol character elements are ordered in sequence toward the adjacent finder pattern as indicated by the arrows in Figure 11.

Each symbol character contains two subsets of odd- and even-numbered elements. The terms odd and even refer to the ordinal number of the elements in each subset. For example the odd-numbered subset consists of the first, third, fifth and seventh elements in each symbol character starting with the element farthest from the adjacent finder pattern. The odd elements are farther from the finder pattern adjacent to the character, while the eighth (inside, even) element is adjacent to the finder pattern.

7.2.3 Symbol character value

For each symbol value, an algorithm assigns a pattern of element widths to the odd and even subsets. The algorithm is given the number of elements, the number of modules, maximum element width, and whether the subset can have all elements wider than one module. Annex B gives a C-language implementation of the GS1 DataBar Expanded symbol character element generation algorithm.

The valid odd element subsets shall have at least one single-module element. The valid even element subsets need not have a single-module-wide element. The odd element restriction insures that the symbol characters have unique edge-to-similar-edge (bar plus space and space plus bar) module sums.

In addition to the above restriction, the first (furthest from the finder pattern) odd element is always less than five modules wide. This restriction prevents a false finder pattern from appearing between adjacent symbol characters.

Table 8 shows the characteristics of the (17,4) subsets, listing the odd and even subset pairs in five groups. The odd subset has an even number of modules and the even subset has an odd number of modules. The widest element widths are specified so that the number of modules in a pair of adjacent elements is never greater than nine. The total number of combinations for a symbol character is 4 192.

Table 8 — Symbol character (17,4) characteristics

Value range	Group	Sum of previous groups G_{SUM}	Odd/even subset modules	Odd/even widest elements	Odd subset total values, T_{ODD}	Even subset total values, T_{EVEN}
0 to 347	1	0	12/5	7/2	87	4
348 to 1 387	2	348	10/7	5/4	52	20
1 388 to 2 947	3	1 388	8/9	4/5	30	52
2 948 to 3 987	4	2 948	6/11	3/6	10	104
3 988 to 4 191	5	3 988	4/13	1/8	1	204

The symbol character value V_S is calculated by:

$$V_S = (V_{ODD} \times T_{EVEN}) + V_{EVEN} + G_{SUM}$$

where T_{EVEN} is the even subset total value, V_{ODD} is the odd subset value, V_{EVEN} is the even subset value, and G_{SUM} is the sum of the products of values for each previous group. To encode a specific symbol character of value V_S:

$$V_{ODD} = (V_S - G_{SUM}) \text{ div } T_{EVEN}$$

$$V_{EVEN} = (V_S - G_{SUM}) \text{ mod } T_{EVEN}$$

For example a symbol character with the value of 3 544 is to be encoded. From Table 8, the value of the symbol character is in the range of Group 4, so G_{SUM} = 2 948 and T_{EVEN} = 104. Using the above equations:

$$V_{ODD} = (3\,544 - 2\,948) \text{ div } 104 = 596 \text{ div } 104 = 5$$

$$V_{EVEN} = (3\,544 - 2\,948) \text{ mod } 104 = 596 \text{ mod } 104 = 76$$

Using the Annex B algorithm, the symbol character from Group 4 (see Table 8) has an odd subset with 6 modules and a sequential value of 5 out of 10 (range 0 to 9) and an even subset with 11 modules and sequential value of 76 out of 104 (range 0 to 103). The odd element widths are {1 3 1 1} and the even element widths are {4 1 4 2} giving the symbol character element widths of {1 4 3 1 1 4 1 2}.

7.2.4 Symbol binary value

The symbol character values 0 to 4 095 each represent 12 bits of a binary number encoding the symbol value. The bit values for each symbol character are concatenated to form the encoded bit string. The first symbol character (second symbol character) contains the highest order bits.

The symbol sizes and their binary string length are listed in Table 9.

Table 9 — Binary capacity of each symbol size

Number of symbol characters	Number of symbol characters encoding data	Bits encoded
4	3	36
5	4	48
6	5	60
7	6	72
8	7	84
9	8	96
10	9	108
11	10	120
12	11	132
13	12	144
14	13	156
15	14	168
16	15	180
17	16	192
18	17	204
19	18	216
20	19	228
21	20	240
22	21	252

7.2.5 Data encodation

The user data to be encoded into a GS1 DataBar Expanded symbol consists of application identifiers and data fields that comply with the data standard of the GS1 General Specifications, formatted exactly as they would be for encodation into a GS1-128 symbol. The GS1-128 rules for concatenation of AI element strings, such as the use of a FNC1 to separate a variable length element string from the element string following it, shall be followed when encoding a GS1 DataBar Expanded symbol.

The GS1 DataBar Expanded binary string is divided into up to five binary fields. All symbols require the first two fields and one or more of the other three fields. The fields are:

a) 2D component linkage flag (see 7.2.5.1)

b) encodation method (see 7.2.5.2)

c) variable length symbol bit field (see 7.2.5.3)

d) compressed data (see 7.2.5.4)

e) general-purpose data compaction (see 7.2.5.5)

These binary fields are concatenated in sequence and encoded in the binary data string of the symbol. The encodation method is always encoded after the 2D component linkage flag. Fixed length encodation methods exactly fill the specified binary data string of the symbol with compressed data. Variable length encodation methods end with an optional general-purpose data compaction field followed by pad bits to fill any unused bits in the binary data string of the appropriately sized symbol.

In the text of the following subclauses, bit fields will be indicated by their binary values enclosed in double quotation marks.

7.2.5.1 2D component linkage flag field

This is a single bit that indicates whether the GS1 DataBar Expanded symbol is printed as part of a GS1 Composite symbol. Its value is zero for a stand-alone GS1 DataBar Expanded symbol or one to indicate that it is the linear component of a GS1 Composite symbol.

7.2.5.2 Encodation method field

The encodation method field consists of one or more bits. It is encoded directly following the 2D component linkage flag. It defines whether the symbol is a general-purpose symbol or begins with an application-oriented compressed data field (such as for efficient representation of an item identification AI element string). The encodation method field is defined in Table 10.

Table 10 — Encodation methods and characteristics

Encodation method field	Number of symbol characters	Compressed field Length	General-purpose compaction field	AI element strings
1	5 to 22	44	yes	(01) and other AIs
00	4 to 22	None	yes	any AIs
0100	6 fixed length	55	no	(01) and (3103)
0101	6 fixed length	55	no	(01) and (3202)/(3203)
01100	6 to 22	42	yes	(01) and (392x)
01101	7 to 22	52	yes	(01) and (393x)
0111000	8 fixed length	76	no	(01), (310x), and (11)
0111001	8 fixed length	76	no	(01), (320x), and (11)
0111010	8 fixed length	76	no	(01), (310x), and (13)
0111011	8 fixed length	76	no	(01), (320x), and (13)
0111100	8 fixed length	76	no	(01), (310x), and (15)
0111101	8 fixed length	76	no	(01), (320x), and (15)
0111110	8 fixed length	76	No	(01), (310x), and (17)
0111111	8 fixed length	76	No	(01), (320x), and (17)

The encodation method field of “1” is for encoding AI 01 primary data with supplementary AIs.

The encodation method field of “00” is for items that do not use AI 01 primary identification. It defines a variable length symbol of four or more symbol characters. The symbol has a general-purpose data compaction field, but no compressed data field.

The encodation method fields of “0100” and “0101” encode primary identification and weight for variable weight items. The encodation method fields of “0111000” through “0111111” encode primary identification, weight, and any one of the four AI date fields.

The encodation method fields of “01100” and “01101” encode primary identification and price.

7.2.5.3 Variable length symbol bit field

This field is present only in variable length symbols with an encodation method field of “1”, “00”, “01100”, and “01101”, and follows the encodation method field. The field contains two bits. The first bit is zero if there is an even number of symbol characters in the symbol or one if the number of symbol characters is odd. The second bit is the size group bit. It is zero if the number of symbol characters in the symbol is less than or equal to 14 and one if the number of symbol characters are greater than 14. These two bits along with the set of finder patterns provide a double check on the number of symbol characters defined in the check character (see 7.2.6).

7.2.5.4 Compressed data field

The binary data in this field is interpreted according to the specific encodation method. All symbols except those with an encodation method field of “00” contain a compressed data field.

7.2.5.4.1 Encodation method field “1” – general item identification data

This encodation method may be used if the AI element string 01 occurs at the start of the data message to be encoded. The 2-digit AI 01 and the check digit are stripped from the item identification AI element string. The remaining 13 digits are encoded in a 44-bit compressed data field in five groups of one, three, three, three, and three digits encoded into four, ten, ten, ten, and ten bits respectively. Each group is encoded as a straight decimal to binary conversion. Additional supplementary AI data is encoded in the general-purpose data compaction field immediately following the 44-bit compressed data field.

A decoder will reconstruct the compressed data field by converting the 44 bits to 13 digits in the five groups of four, ten, ten, ten, and ten bits respectively. It will add the two digits of 01 as a prefix and add the calculated mod 10 check digit to the end of the AI element string. The remainder of the bit string is decoded using general-purpose data compaction.

As an example of Method “1”, (01)**000123456789**05(10)ABC123 could be encoded, where only the bold digits are actually encoded in the compressed data field. The AI 10 and batch number ABC123 are encoded in the general-purpose data compaction field directly following the 44-bit compressed data field. The decoder will transmit:]e0010001234567890510ABC123.

Method “1” defines a variable length symbol of five or more symbol characters including a general-purpose data compaction field.

7.2.5.4.2 Encodation method field “0100” – variable weight item (0,001 kilogram increments)

This encodation method may be used if the data message to be encoded consists of only the two AI element strings, AI 01 followed by AI 3103. The AI 01 item identification element string must have an indicator digit of 9. The AI 3103 variable weight element string must specify a weight no more than 32,767 kilograms. The two AI element strings are compressed into 40 bits and 15 bits respectively for a total field length of 55 bits. Method “0100” encodes a fixed length symbol of six symbol characters without a general-purpose data compaction field.

To encode the two AI element strings, the leading three digits of 019 and the trailing check digit are stripped from the AI 01 element string. The 12 digits remaining are encoded in a 40-bit compressed data field in four groups of three digits each, encoded into four groups of ten bits. The leading 4-digit AI of 3103 is stripped from the AI 3103 element string. The remaining 6 digits in the range of 000000 to 032767 are encoded in 15 bits and added to the compressed data field.

A decoder will decode the first 40 bits into 12 digits in four groups of ten bits converted to three digits each. It will prefix the 12 digits with three digits of 019. It will also add the calculated mod 10 check digit to the end of the first AI element string. It will convert the remaining 15 bits to base 10, padding the resulting digits with leading zeroes to form six digits. It will add the AI prefix of 3103 to the decoded six-digit weight to form the second AI element string in the data message.

As an example of Method “0100”, (01)9**001234567890**8(3103)**001750** could be encoded, where only the bold digits are actually encoded in the compressed data field. The decoder will transmit:]e0019001234567890831030001750.

7.2.5.4.3 Encodation method field “0101” – variable weight item (0,01 or 0,001 pound increments)

This encodation method may be used if the data message to be encoded consists of only the two AI element strings, AI 01 followed by AI 3202 or AI 3203. The AI 01 item identification element string must have an indicator digit of 9. The AI 3202 variable weight element string must specify a weight no more than 99,99 pounds. The AI 3203 variable weight element string must specify a weight no more than 22,767 pounds. The two AI element strings are compressed into 40 bits and 15 bits respectively for a total field length of 55 bits. Method “0101” defines a fixed length symbol of six symbol characters without a general-purpose data compaction field.

To encode the two AI element strings, the leading three digits of 019 and the trailing check digit are stripped from the AI 01 element string. The 12 digits remaining are encoded in a 40-bit compressed data field in four groups of three digits each, encoded into four groups of ten bits. The leading 4-digit AI is stripped from the weight element string. For AI 3202, the value of the remaining 6 digits, which shall be in the range of 0 to 9 999, is encoded in 15 bits and added to the compressed data field. For AI 3203, the value of the remaining 6 digits shall be in the range of 0 to 22 767. This value plus 10 000 is encoded in 15 bits and added to the compressed data field.

A decoder will decode the first 40 bits into 12 digits in four groups of ten bits converted to three digits each. It will prefix the 12 digits with three digits of 019. It will also add the calculated mod 10 check digit to the end of the first AI element string. It will convert the remaining 15 bits to a value. If the value is less than 10 000, it will convert the value to six digits prefixed by the AI 3202 to form the AI element string in the data message. Otherwise it will subtract 10 000 from the value, convert the remainder to six digits by padding with zeros on the left if necessary, and add the AI prefix of 3203 to the six-digit value, to form the second AI element string in the data message.

As an example of Method “0101”, (01)9**001234567890**8(3202)00**0156** could be encoded, where only the bold digits are actually encoded in the compressed data field. The decoder will transmit “]e001900123456789083202000156”.

7.2.5.4.4 Encodation method fields “0111000” through “0111111” – variable weight item plus date

This encodation method may be used if the data message to be encoded consists of the two or three AI element strings for AI 01, AI 310x or 320x (x ranging from 0 to 9), and optionally AI 11 or 13 or 15 or 17. The AI 01 item identification element string must have an indicator digit of 9. The variable weight AI element string may have any value from 0 to 99 999. Methods “0111000” through “0111111” have a fixed length symbol of eight symbol characters without a general-purpose data compaction field.

The eight methods encoding variable weight product and date are:

Method "0111000"	AI 01 + AI 310x + AI 11	metric weight and production date
Method "0111001"	AI 01 + AI 320x + AI 11	English weight and production date
Method "0111010"	AI 01 + AI 310x + AI 13	metric weight and packaging date
Method "0111011"	AI 01 + AI 320x + AI 13	English weight and packaging date
Method "0111100"	AI 01 + AI 310x + AI 15	metric weight and “best before” date
Method "0111101"	AI 01 + AI 320x + AI 15	English weight and “best before” date
Method "0111110"	AI 01 + AI 310x + AI 17	metric weight and expiration date
Method "0111111"	AI 01 + AI 320x + AI 17	English weight and expiration date

The three AI element strings are compressed into 76 bits. The item identification is compressed into 40 bits, the weight is compressed into 20 bits, and the date is compressed into 16 bits.

To encode the three AI element strings, the leading three digits of 019 and the trailing check digit are stripped from the AI 01 element string. The 12 digits remaining are encoded in a 40-bit compressed data field in four groups of three digits each, encoded into four groups of ten bits. The leading 3-digit partial AI of 310 or 320 is stripped from the weight element string. The second digit (a zero) is removed from the remaining 7-digit AI element string data to form six digits. These 6 digits, which consist of the last digit of the AI and the last five digits of the weight, are encoded in 20 bits and added to the compressed data field. The 16-bit compressed date is formed by stripping the two-digit AI 11, 13, 15, or 17 and converting the remaining “YYMMDD” six digits to a compressed string with values 0 to 38 399:

$$(YY \times 384) + ((MM-1) \times 32) + (DD).$$

The value of 38 400 in the 16-bit sub-field is used to indicate that no date field is encoded. This option is used to encode item identification and weight when the weight value is not supported by Methods “0100” or “0101”.

A decoder will decode the first 40 bits into 12 digits in four groups of ten bits converted to three digits each. It will prefix the 12 digits with three digits of 019. It will also add the calculated mod 10 check digit to the end of the first AI element string. It will convert the next 20 bits to base 10 to form six digits. It will insert a zero after the first digit to form seven digits. It will add the AI prefix of 310 or 320 to the seven-digit string to form the second AI element string in the data message. It will convert the final 16 bits to a decimal value. If the value of the last 16 bits is 38 400, no further data is decoded.

Otherwise a date was encoded. The 6-digit date shall be extracted according to the above equation and an application identifier (11, 13, 15, or 17) added as a prefix by the decoder according to the method. The eight digits are added to the decoded message as the third AI element string.

As an example of Method "0111000", (01)9**001234567890**8(310**3**)0**12233**(15)**991231** could be encoded, where only the bold digits are actually encoded in the compressed data field. The decoder will transmit "]e0019001234567890831030122331599 1231".

7.2.5.4.5 Encodation method field "01100" – variable measure item and price

This encodation method may be used if the data message starts with the two element strings AI 01 followed by AI 392x. The AI 01 item identification element string must have an indicator digit of 9. The AI 392x price may only have from zero to three digits to the right of the decimal point (x = 0 to 3). The AI 01 element string is compressed into 40 bits and the AI 392x decimal point digit, x, is encoded in two bits. The price digits are then encoded in general-purpose Numeric encodation. Any additional element strings are encoded using general-purpose encodation.

To encode the two element strings, the leading three digits of 019 and the trailing check digit are stripped from the AI 01 element string. The 12 digits remaining are encoded in a 40-bit compressed data field in four groups of three digits each, encoded into four groups of ten bits. The first three digits, 392, of the price AI are stripped from the AI 392x element string. The next digit, x, is the decimal point position and is encoded directly in the next two bits for the allowable values 0 to 3. The data after the decimal point digit is encoded in general-purpose encodation starting in the Numeric encodation scheme.

A decoder will decode the first 40 bits into 12 digits in four groups of ten bits converted to three digits each. It will prefix the 12 digits with three digits of 019. It will also add the calculated mod 10 check digit to the end of the first element string. It will add the first three digits of the AI, 392, followed by the decimal point digit which is decoded from the next two bits. The remainder of the message will be decoded using general-purpose encodation.

As an example of Method "01100", (01)9**001234567890**8(392**2**)795 could be encoded, where only the bold digits are actually encoded in the compressed data field. The decoder will transmit:]e001900123456789083922795.

7.2.5.4.6 Encodation method field "01101" – variable measure item and price with ISO 4217 Currency Code

This encodation method may be used if the data message starts with the two element strings AI 01 followed by AI 393x. The AI 01 item identification element string must have an indicator digit of 9. The AI 393x price may only have from zero to three digits to the right of the decimal point (x = 0 to 3). The AI 01 element string is compressed into 40 bits and the AI 393x decimal point, x, is encoded in two bits. The 3-digit ISO 4217 Currency Code is encoded in ten bits. The price digits are then encoded in general-purpose Numeric encodation. Any additional element strings are encoded using general-purpose encodation.

To encode the two element strings, the leading three digits of 019 and the trailing check digit are stripped from the AI 01 element string. The 12 digits remaining are encoded in a 40-bit compressed data field in four groups of three digits each, encoded into four groups of ten bits. The first three digits, 393, of the price AI are stripped from the AI 393x element string. The next digit, x, is the decimal point position and is encoded directly in the next two bits for the allowable values 0 to 3. The next three digits, the ISO 4217 Currency Code, are encoded directly in the next ten bits. The data after the three-digit ISO 4217 Currency Code is encoded in general-purpose encodation starting in the Numeric encodation scheme.

A decoder will decode the first 40 bits into 12 digits in four groups of ten bits converted to three digits each. It will prefix the 12 digits with three digits of 019. It will also add the calculated mod 10 check digit to the end of the first element string. It will add the first three digits of the AI, 393, followed by the decimal point digit which is decoded from the next two bits. The next ten bits are directly decoded as the three digits of the ISO 4217 Currency Code. The remainder of the message will be decoded using general-purpose encodation.

As an example of Method “01101”, (01)9**001234567890**8(393**2**)**040**1234 could be encoded, where only the bold digits are actually encoded in the compressed data field. The decoder will transmit:]e0019001234567890839320401234.

7.2.5.5 General-purpose data compaction field

The general-purpose data compaction field encodes AI element strings into a binary string, using a combination of three encodation schemes:

a) Numeric encodation is used for numbers and FNC1 characters, requiring 3,5 bits per encoded character;

b) Alphanumeric encodation is used for a mix of numbers and uppercase letters, requiring 5 bits per digit or FNC1 character, and 6 bits per letter;

c) ISO/IEC 646 encodation is used for a mix of uppercase and lowercase letters, numbers, and most punctuation characters, requiring 5 bits per digit, 7 bits per letter, and 8 bits per punctuation character.

General-purpose data compaction is used to encode the AI element strings other than those specifically encoded using one of the compressed data encodation methods. The general-purpose data compaction field is the last field in a variable length symbol. Once all of the data has been processed using general-purpose data compaction scheme(s), the general-purpose encoding process concludes with the padding procedure described in 7.2.5.5.4.

7.2.5.5.1 Numeric encodation

Numeric encodation is the default encodation scheme that is in effect at the start of the general-purpose data encodation field. It encodes two digits or a digit and FNC1 character in either order in seven bits. The seven-bit value is determined by:

$$\text{value} = (11 \times D_1) + D_2 + 8$$

where D_1 and D_2 are the first and second digit values or FNC1 value (FNC1 is assigned a value of 10). The value ranges from 8 to 127 corresponding to the seven-bit values of “0001000” to “1111111” (two FNC1 characters cannot be encoded in seven bits). A binary sequence of four zeros “0000” at the start of the field or following the end of a previous Numeric encodation seven-bit value signals a latch, or change, to Alphanumeric encoding (see Table 11).

Table 11 — Numeric encodation

Character(s)	Encoded binary data
digit-digit, digit-FNC1, and FNC1-digit pairs	0001000 to 1111111
Alphanumeric latch	0000

A Numeric encodation sequence continues to encode pairs of data characters until one of the following conditions becomes true:

a) If at least two characters remain and Numeric encodation cannot be applied, encode an Alphanumeric latch in the data compaction field.

b) If one character remains, which is not a digit, encode an Alphanumeric latch in the data compaction field.

c) If one character remains, which is a digit, first calculate the symbol size needed to encode the current binary string, and then the number of unused bits is equal to the number needed to bring the total number of bits up to the next even multiple of 12.

   1) If seven or more unused bits remain, encode the digit and a FNC1 pad in the next seven bits. This trailing FNC1 will be recognized as a pad and will not be transmitted by the reader.

2) If four to six bits remain, add one to the digit value and encode the result in the next four bits.

3) Otherwise the next larger symbol size will be used, encoding the digit and a FNC1 pad in the next seven bits. This trailing FNC1 will be recognized as a pad and will not be transmitted by the reader.

Any remaining bits are encoded according to the padding procedure of 7.2.5.5.4.

d) No characters remain: any remaining bits are encoded according to the padding procedure of 7.2.5.5.4.

Whenever an Alphanumeric latch is encoded, the encodation scheme is changed to reflect that latch. If the next data character requires ISO/IEC 646 encodation, the Alphanumeric encodation scheme will immediately encode an ISO/IEC 646 latch following the Alphanumeric latch.

During decoding the following special checks must be made when Numeric compaction is in effect at the end of the symbol:

a) If the last seven bits of Numeric encodation immediately before the pad sequence encode a digit followed by a FNC1, the FNC1 is ignored.

b) If Numeric encodation is in effect when only four to six bits remain in the symbol, then the value of the 4-bit string at the start of the remaining bits is converted into its decimal value.

1) If the value is zero, then the data message is complete.

2) Otherwise, it is decoded as a final digit in the data message equal to the decimal value minus one.

7.2.5.5.2 Alphanumeric encodation

Alphanumeric encodation encodes the digits and FNC1/Numeric latch, the uppercase letters, five punctuation characters, and two latch characters. The encoded bit stream does not have a fixed bit length per character. The bits are allocated according to the bit length of each encoded character. Each character is encoded in from three to six bits as shown in Table 12.

Table 12 — Alphanumeric encodation

Character(s)	ASCII value(s)	Encoded Value	Encoded binary data
0 to 9	48 to 57	ASCII value minus 43 (5-bit)	00101 to 01110
FNC1/Numeric latch		15 (5-bit)	01111
A to Z	65 to 90	ASCII value minus 33 (6-bit)	100000 to 111001
* (asterisk)	42	58 (6-bit)	111010
, (comma)	44	59 (6-bit)	111011
- (minus or hyphen)	45	60 (6-bit)	111100
. (period or full stop)	46	61 (6-bit)	111101
/ (slash or solidus)	47	62 (6-bit)	111110
Numeric latch		0 (3-bit)	000
ISO/IEC 646 latch		4 (5-bit)	00100

The data is encoded by appending the variable length binary data for each character to the general-purpose data compaction field with the following exceptions:

a) If the next character is a FNC1, encode it in alphanumeric encodation.

b) If the next character can only be encoded using ISO/IEC 646 encodation, encode an ISO/IEC 646 latch in the data compaction field.

c) If the next six characters can be encoded using Numeric encodation, encode a Numeric latch in the data compaction field.

d) If the next four or more characters can be encoded using Numeric encodation and they terminate the data string, encode a Numeric latch in the data compaction field.

Whenever a latch is encoded, the encodation scheme is changed to reflect that latch.

The encoded bit field is decoded by first examining the first one or three bits in the field following the previously decoded character or initially at the start of the field.

a) If the first bit is “1”, decode the next character as a 6-bit character.

b) If the first three bits are “000”, it is a numeric latch.

c) Otherwise, decode the next character as a 5-bit character.

7.2.5.5.3 ISO/IEC 646 encodation

This mode encodes the digits, the uppercase and lowercase letters, and 21 punctuation characters of ISO/IEC 646, as well as FNC1 and two latch characters required by the symbology. The encoded bit stream does not have a fixed bit length per character. The bits are allocated according to the bit length of each encoded character. Each character is encoded in from three to eight bits as shown in Table 13.

Table 13 — ISO/IEC 646 encodation

Character(s)	ASCII value(s)	Encoded Value	Encoded binary data
0 to 9	48 to 57	ASCII value minus 43 (5-bit)	00101 to 01110
FNC1/Numeric latch		15 (5-bit)	01111
A to Z	65 to 90	ASCII value minus 1 (7-bit)	1000000 to 1011001
a to z	97 to 122	ASCII value minus 7 (7-bit)	1011010 to 1110011
! (exclamation mark)	33	232 (8-bit)	11101000
" (quotation mark)	34	233 (8-bit)	11101001
% (percent sign)	37	234 (8-bit)	11101010
& (ampersand)	38	235 (8-bit)	11101011
' (apostrophe)	39	236 (8-bit)	11101100
((left parenthesis)	40	237 (8-bit)	11101101
) (right parenthesis)	41	238 (8-bit)	11101110
* (asterisk)	42	239 (8-bit)	11101111
+ (plus sign)	43	240 (8-bit)	11110000
, (comma)	44	241 (8-bit)	11110001
- (minus or hyphen)	45	242 (8-bit)	11110010
. (period or full stop)	46	243 (8-bit)	11110011
/ (slash or solidus)	47	244 (8-bit)	11110100
: (colon)	58	245 (8-bit)	11110101
; (semicolon)	59	246 (8-bit)	11110110
< (less-than sign)	60	247 (8-bit)	11110111
= (equals sign)	61	248 (8-bit)	11111000
> (greater-than sign)	62	249 (8-bit)	11111001
? (question mark)	63	250 (8-bit)	11111010
_ (underline or low line)	95	251 (8-bit)	11111011
Space	32	252 (8-bit)	11111100
Numeric latch		0 (3-bit)	000
Alphanumeric latch		4 (5-bit)	00100

The data is encoded by appending the variable length binary data for each character to the data compaction field with the following exceptions:

a) If the next character is a FNC1, encode it in ISO/IEC 646 encodation.

b) If the next four characters can be encoded in Numeric Compaction and no character which can only be encoded using ISO/IEC 646 encodation occurs in the next ten characters, encode a Numeric latch in the data compaction field.

c) If the next five characters can be encoded in Alphanumeric Compaction and no character which can only be encoded using ISO/IEC 646 encodation occurs in the next ten characters, encode an Alphanumeric latch in the data compaction field.

With either exception, if the data terminates in less than ten characters, then the ten-character test completes early at the end of the data. If a latch is encoded, the encodation scheme is changed to reflect that latch.

The encoded bit field is decoded by first examining the first three or five bits following the previously decoded character or initially at the start of the field:

a) If the first three bits are “000”, it is a numeric latch.

b) Otherwise, get the decimal value of the first five bits. If the value is:

   1) 15 or less, decode the next character as a 5-bit character;

   2) 16 to 28, decode the next character as a 7-bit character;

   3) 29 or more, decode the next character as a 8-bit character.

7.2.5.5.4 Pad bits for the general-purpose data compaction field

The number of symbol characters in the symbol should be the minimum required for encoding the data in that symbol. However, there may be unused bits in the symbol after the data is encoded in the data compaction field. These bits shall be filled with a pad bit sequence until the data capacity of the symbol is filled.

The padding bit string is created by repeated five-bit pad sequences of “00100”, which is both the ISO/IEC 646 latch in Alphanumeric encodation and the Alphanumeric latch in ISO/IEC 646 encodation, so that the encodation modes and their latches alternate without encoding more data. The last pad sequence may be truncated on the right if there are not enough bits left in the symbol.

If Numeric encodation ends the data encodation, a four-bit alphanumeric latch of “0000” is required before the “00100” alternating latch pad sequence. For example, if the encoding ends in Numeric encodation and there are seven remaining bits, they shall be encoded as “0000001”, which is the alphanumeric latch “0000” followed by the first three bits “001” of the ISO/IEC 646 latch of “00100”. The first four-bit latch itself may be shortened if fewer than four unused bits remain to be padded.

7.2.6 Check character

The first symbol character in a GS1 DataBar Expanded symbol is the check character. It encodes both the symbol length and a checksum of the weighted symbol character element widths. Only the first 4 009 values (0 to 4 008) of the check character are used.

The number of symbol characters, S, in the symbol (4 to 22) and the checksum value are encoded in the check character as:

$$\text{check character value} = 211 \times (S - 4) + \text{checksum value}$$

The checksum value is equal to the mod 211 residue of the weighted sum of the widths of the elements in the symbol characters. The weights in Table 14 are consecutive powers of 3 mod 211.

Table 14 — Element weights used in the mod 211 checksum calculation

Symbol Character Relation to Finder	Symbol Character Element Weights							
	1	2	3	4	5	6	7	8
A1 left*	-	-	-	-	-	-	-	-
A1 right	1	3	9	27	81	32	96	77
A2 left	20	60	180	118	143	7	21	63
A2 right	189	145	13	39	117	140	209	205
B1 left	193	157	49	147	19	57	171	91
B1 right	62	186	136	197	169	85	44	132
B2 left	185	133	188	142	4	12	36	108
B2 right	113	128	173	97	80	29	87	50
C1 left	150	28	84	41	123	158	52	156
C1 right	46	138	203	187	139	206	196	166
C2 left	76	17	51	153	37	111	122	155
C2 right	43	129	176	106	107	110	119	146
D1 left	16	48	144	10	30	90	59	177
D1 right	109	116	137	200	178	112	125	164
D2 left	70	210	208	202	184	130	179	115
D2 right	134	191	151	31	93	68	204	190
E1 left	148	22	66	198	172	94	71	2
E1 right	6	18	54	162	64	192	154	40
E2 left	120	149	25	75	14	42	126	167
E2 right	79	26	78	23	69	207	199	175
F1 left	103	98	83	38	114	131	182	124
F1 right	161	61	183	127	170	88	53	159
F2 left	55	165	73	8	24	72	5	15
F2 right	45	135	194	160	58	174	100	89

* The symbol character to the left of the A1 finder is the check character, which has no element weighting.

The weighted mod 211 checksum value is calculated by:

$$(W_{1,1}E_{1,1} + W_{1,2}E_{1,2} + \ldots + W_{1,8}E_{1,8} + \ldots + W_{x,8}E_{x,8}) \bmod 211$$

where $W_{N,M}E_{N,M}$ is the product of the weight for symbol character N at ordinal element position M, from Table 14, and the module width of element M in symbol character N for all symbol characters in the symbol. Note that N is a symbol character designation (for example "C1 right" which is for the symbol character to the right of finder C1) from Table 14 and is not consecutive but determined by the sequence in Table 16. The subscript x is the designation of the last symbol character in the symbol.

Annex F.3 contains an example of encoding a GS1 DataBar Expanded symbol.

7.2.7 Finder pattern

The symbol has twelve unique symbol finder patterns. These patterns are positioned between pairs of symbol characters. Since a finder pattern is adjacent to all the symbol characters, the symbol can be scanned in segments consisting of a symbol character and adjacent finder pattern.

If there is an odd number of symbol characters in the symbol, the last finder pattern and the right guard pattern will terminate the symbol. In this case, the last finder pattern will be adjacent to and to the right of the last symbol character.

The twelve patterns are based on six basic patterns A through F. Each basic pattern has two versions, the first with element one being a space on the left and the second with element one being a bar on the right. The black-white inversion allows the two patterns to be differentiated. The 12 finder patterns are A1, A2, B1, B2, C1, C2, D1, D2, E1, E2, F1, and F2. The '1' versions have a space on the left for element one. The '2' versions are mirrored left-to-right and inverted black-white. Table 15 lists the element widths for the six basic patterns.

Each finder pattern consists of five elements comprising 15 modules. The sum of the modules in the elements 2 and 3 in the '1' versions is 10 to 12, while the sum of the modules in elements 4 and 5 in the '1' versions is 2. The ratio of the width of the wide element pair two and three to the total width of the four elements two through five, within the range of 10:12 to 12:14, is the basis of the first step in the recognition logic for the finder pattern. Likewise, for the version '2' finders, the total width of the four elements one through four are compared to the width of the wide element pair three and four.

Table 15 — Finder pattern element widths

Base Pattern	Element Widths (1 is a space)					Base Pattern	Element Widths (1 is a bar)				
	1	2	3	4	5		1	2	3	4	5
A1	1	8	4	1	1	A2	1	1	4	8	1
B1	3	6	4	1	1	B2	1	1	4	6	3
C1	3	4	6	1	1	C2	1	1	6	4	3
D1	3	2	8	1	1	D2	1	1	8	2	3
E1	2	6	5	1	1	E2	1	1	5	6	2
F1	2	2	9	1	1	F2	1	1	9	2	2

The finder patterns are used in the symbol in 10 unique sets assigned according to the symbol length (see Table 16). The sets are divided into two groups. Within each group each set has at least one unique subset of finder patterns to differentiate it from the other sets in the group. This selection of sets will protect against an erroneous symbol length derived from a misdecoded check character causing a misread.

Table 16 — Finder pattern sequences

No. of Segments	Finder Pattern Sequence										
	1	2	3	4	5	6	7	8	9	10	11
Group 1											
4	A1	A2									
5 or 6	A1	B2	B1								
7 or 8	A1	C2	B1	D2							
9 or 10	A1	E2	B1	D2	C1						
11 or 12	A1	E2	B1	D2	D1	F2					
13 or 14	A1	E2	B1	D2	E1	F2	F1				
Group 2											
15 or 16	A1	A2	B1	B2	C1	C2	D1	D2			
17 or 18	A1	A2	B1	B2	C1	C2	D1	E2	E1		
19 or 20	A1	A2	B1	B2	C1	C2	D1	E2	F1	F2	
21 or 22	A1	A2	B1	B2	C1	D2	D1	E2	E1	F2	F1

7.2.8 GS1 DataBar Expanded Stacked

GS1 DataBar Expanded may be stacked in two to eleven rows. Figure 12 illustrates a two row symbol. GS1 DataBar Expanded Stacked is used when the symbol area or print mechanism is not wide enough to accommodate the full single row symbol. Each row is 34X high with a 3X separator pattern between the rows. For comparison the Figure 12 symbol encodes the same data as Figure 10.

Figure 12 — GS1 DataBar Expanded Stacked symbol representing (01)98898765432106(3202)012345(15)991231

The number of symbol characters in all but the last row shall be even. There shall be a minimum of four symbol characters in the first row of a GS1 DataBar Expanded Stacked symbol when it is the linear component of a GS1 Composite symbol. The rows shall be ordered in a top to bottom sequence. The last row shall contain a minimum of two symbol characters with extra padding, if needed. If the symbol is part of a GS1 Composite symbol, the 2D component shall be printed above the top row.

Each of these rows shall have a finder pattern between each symbol character pair plus the two-element left and right guard patterns. The last row may have an odd number of symbol characters, in which case the last finder pattern will be next to the right guard pattern.

The first row and following odd-numbered rows shall start with a space, while the second row and following even-numbered rows shall start with a bar with the possible exception of the bottom row, see Figure 13. If there is an even number of segment pairs per row (2, 4, etc. pairs, which is 4, 8, etc. segments) then even-numbered rows shall be printed in reverse element order, i.e. as a mirror image. This reversal is necessary so that these even-numbered rows start with a bar. If there is an odd number of segment pairs per row (1, 3, 5, etc. pairs, which is 2, 6, 10, etc. segments) then the even-numbered rows will naturally start with a bar. Table 17 lists the reversed rows.

Table 17 — Reversed rows in GS1 DataBar Expanded Stacked symbols

Row Number	Width of stacked symbol in segments									
	2	4	6	8	10	12	14	16	18	20
1	F	F	F	F	F	F	F	F	F	F
2	F	R	F	R	F	R	F	R	F	R
3	F	F	F	F	F					
4	F	R	F							
5	F	F								
6	F	R								
7	F									
8	F									
9	F									
10	F									
11	F									
F is a forward row, R is reversed, unachievable entries are left blank.										

A 3X (minimum) high separator pattern shall separate the symbol rows and consists of three 1X (minimum) high rows.

The upper row of the separator pattern is the complement of the bars and spaces in the adjacent symbol row above, except for the ends and 13 modules under the finder pattern elements 1, 2 and 3 of a version 1 finder and elements 3, 4, and 5 of a version 2 finder. These 13 modules are light under the adjacent finder bars and alternating dark, light, dark, etc. under the adjacent finder spaces.

The middle row consists of alternating light and dark modules except for the ends.

The lower row has a similar structure to the upper row. It is the complement of the bars and spaces in the adjacent symbol row below, except for the ends and 13 modules over the finder pattern elements 1, 2 and 3 of a version 1 finder pattern and elements 3, 4, and 5 of a version 2 finder pattern. These 13 modules are light under the adjacent finder bars and alternating dark, light, dark, etc. under the adjacent finder spaces.

The first four and last four modules of the separator rows are always light, overriding the complementary alternating pattern.

The GS1 DataBar Expanded Stacked symbol shown in Figure 12 has eight symbol characters. The symbol is stacked into two rows, each with four symbol characters. The top row consists of the first four symbol characters with a 1X bar and 1X space guard pattern added to the right of the row. The bottom row is printed as a mirror image. It starts on the left with the mirror image of the last four symbol characters followed by an added 1X bar and 1X space guard pattern. The overall size of the example symbol shown in Figure 12 is 102X wide by 71X high.

There is one exception to the row reversal requirement. If the last row should be mirrored left-to-right, but is a partial width row containing an odd number of finder patterns, then the row is offset one module to the right by adding a white module to the left of the row. This is necessary because the partial row with an odd number of finders has symmetrical guard bar patterns which will have the same pattern after reversal. Figure 13 illustrates a GS1 DataBar Expanded Stacked symbol which requires an offset last row.

Figure 13 — GS1 DataBar Expanded Stacked symbol representing (01)95012345678903(3103)000123

7.2.9 Reference decode algorithm

Bar code reading systems are designed to read imperfect symbols to the extent that practical algorithms permit. This section describes the reference decode algorithm used in the computation of the decodability value described in ISO/IEC 15416 for measurement of symbol quality.

The algorithm contains the following steps to decode the symbol:

a) Find a segment by looking left to right and right to left for a four-element sequence with the ratio:

for left to right:

$$9{,}5:12 \leq ((\text{element1} + \text{element2}) : (\text{element1} + \text{element2} + \text{element3} + \text{element4})) \leq 12{,}5:14$$

or for right to left:

$9{,}5:12 \le ((\text{element3} + \text{element4}) : (\text{element1} + \text{element2} + \text{element3} + \text{element4})) \le 12{,}5:14$

This ratio identifies the second through fifth elements of the finder.

Decode the finder pattern using the method in step b) to find the normalized edge-to-similar-edge values E1 and E2 from the pitch, p, the sum of the first four element widths of the finder. Verify that the values E1 and E2 correspond to a valid GS1 DataBar Expanded finder pattern.

b) Verify that the ratio of the pitch of each adjacent symbol character to the pitch of the finder has a value of $(17 \pm 1{,}5) : 15$, then decode each symbol character as follows:

1) Calculate the seven width measurements p, e_1, e_2, e_3, e_4, e_5, and e_6 (Figure 14).

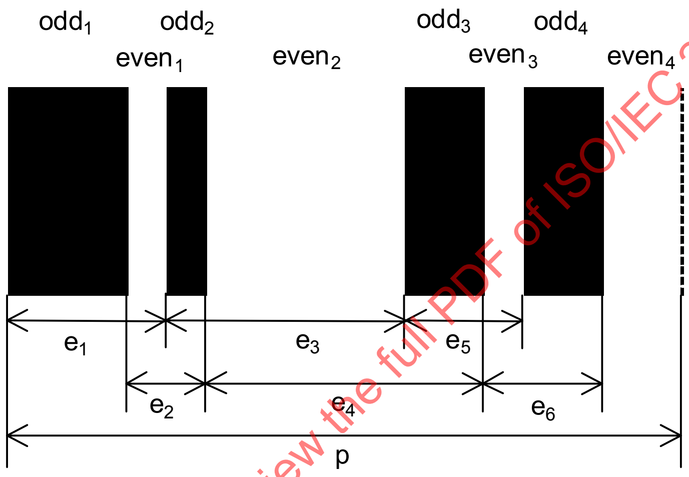

Figure 14 — Decode measurements

NOTE The diagram shows the first element as the left black element, but the symbol characters are also left-to-right mirrored and/or black-white inverted.

2) Convert measurements e_1, e_2, e_3, e_4, e_5, and e_6 to normalized values E_1, E_2, E_3, E_4, E_5, and E_6 which will represent the integral module width (E_i) of these measurements. The following method is used for the i-th value.

If $1{,}5p/17 \le e_i < 2{,}5p/17$, then $E_i = 2$
If $2{,}5p/17 \le e_i < 3{,}5p/17$, then $E_i = 3$
If $3{,}5p/17 \le e_i < 4{,}5p/17$, then $E_i = 4$
If $4{,}5p/17 \le e_i < 5{,}5p/17$, then $E_i = 5$
If $5{,}5p/17 \le e_i < 6{,}5p/17$, then $E_i = 6$
If $6{,}5p/17 \le e_i < 7{,}5p/17$, then $E_i = 7$
If $7{,}5p/17 \le e_i < 8{,}5p/17$, then $E_i = 8$
If $8{,}5p/17 \le e_i < 9{,}5p/17$, then $E_i = 9$

Otherwise the character is in error.

3) Determine the normalized element widths from the E values. The last element is assigned the remaining modules rather that being calculated from the E values. The set of valid element widths is the only solution that has no element widths less than one module and has at least one odd element that is one module wide. For example the Figure 14 values E_1 through E_6 are {4 2 6 7 3 3}. The possible derived element sets are {2 2 0 6 1 2 1 3}, {3 1 1 5 2 1 2 2}, or {4 0 2 4 3 0 3 1}. Only the eight element widths {3 1 1 5 2 1 2 2} satisfy the requirements and therefore are selected as the element widths. If no set of derived element widths is valid, then the character is in error. Annex G gives a C-language implementation of this element width decoding algorithm.

4) Determine the values of the odd and even subsets from the program in Annex B.

5) Calculate the symbol character value from the odd and even subset values.

6) Decode the value of the finder pattern using the normalized element width method above and look up the pattern in Table 15.

7) Calculate and store the weighted element widths for the checksum calculation.

c) When all the symbol characters and finder patterns specified in the check character have been decoded, verify that the mod 211 checksum calculated from the check character matches the sum of the weighted element widths of the symbol characters mod 211.

d) Translate the symbol characters into a binary string and translate the binary string to AI element strings.

e) In addition, perform such other secondary checks on acceleration, absolute timing, dimensions, etc., as are deemed prudent and appropriate considering the specific reading device and intended application environment.

In the design of practical scanners for GS1 DataBar Expanded and GS1 DataBar Expanded Stacked, security measures such as redundant capture of data are required. Annex H contains additional symbol decoding considerations which should be followed to minimize misreads.

8 Symbol quality

8.1 Linear symbology parameters

ISO/IEC 15416 defines a standardized methodology for measuring and grading bar code symbols. The GS1 DataBar family of symbols shall be evaluated according to that standard. The reference decode algorithms defined for the GS1 DataBar symbology types in this specification shall be used for the assessment of the ‘decode’ and ‘decodability’ parameters under ISO/IEC 15416. All elements in the separator patterns should be visually distinguishable. For the purpose of assessing symbol quality the separator patterns are not graded. For guidance on printing, see Annex J.

8.2 Additional pass/fail criteria

ISO/IEC 15416 allows for additional pass/fail criteria to be stipulated by a symbology specification. For the first and third types of GS1 DataBar symbols the additional criteria are that on each test scan, both of the interior guard pattern elements must be present and each must be no wider than 3Z. Any individual scan profile that does not meet this requirement shall receive a grade of 0.

This revision defines new guard bar/space elements in GS1 DataBar Limited that are different from the first edition of this International Standard. The guard bar/space elements are now checked in the new reference decode algorithm and shall conform to 6.2.6.g and 6.2.6.h. In addition, the trailing guard pattern shall meet the following requirement or else receive a grade of 0 for that scan:

The sum of the widths of the interior space and the bar of the trailing guard pattern shall exceed 11/52 of the sum of the widths of the fourteen elements comprising the right symbol character (i.e. 5.5 modules).

8.3 Stacked symbols

Each row shall be evaluated in accordance with ISO/IEC 15416 as though it were a separate symbol. Scan lines shall pass through the inspection band of the central 80% of the height of each row, as specified in ISO/IEC 15416, in order to minimise the effects of cross-talk from adjacent rows. The minimum number of scans per row should be the lower of ten, or the row height divided by the measuring aperture diameter. The overall symbol grade shall be the lowest overall grade obtained for any row.

9 Transmitted data

The GS1 DataBar family of symbologies is designed and intended to be used with symbology identifiers as specified in ISO/IEC 15424. Applications using GS1 DataBar should have symbology identifiers enabled in their readers. The GS1 system requires the use of symbology identifiers. Applications that do not use symbology identifiers either will not recognize the application identifiers in a GS1 DataBar symbol or conversely will misinterpret the data from other symbols as application identifier data. GS1 DataBar symbols are transmitted using a symbology identifier prefix of “]e0”. If a 2D component accompanies the GS1 DataBar symbol, the AI element string data immediately follows the data in the linear component.

A GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional or GS1 DataBar Limited symbol would be transmitted as “]e001**1001234567890**2” where the bold data 1001234567890 for this example is explicitly encoded in the symbol. The symbology identifier prefix, application identifier “01” for item identification, and calculated GS1 mod 10 check digit of “2” are added to the transmitted data string.

GS1 DataBar Expanded and GS1 DataBar Expanded Stacked encodes the application identifiers, so only the symbology identifier Prefix is added as a prefix to the encoded data. GS1 DataBar Expanded and GS1 DataBar Expanded Stacked can encode a FNC1 character which is transmitted as a <GS> (ASCII 29) unless it is the last character in a symbol in which case it is not transmitted.

For GS1 Composite symbols, the AI element strings in the 2D component shall be transmitted directly following the data in the linear component. Normally, a reader must decode both components if the linkage flag in the linear component is set. However, a reader shall also support a mode where only the linear component is decoded and transmitted regardless of the state of the linkage flag. This mode supports applications that only require the primary item identification.

The reader shall support an option for GS1-128 emulation. This mode emulates the GS1-128 symbology for data transmission in accordance with the GS1 General Specifications. The symbology identifier shall be]C1. GS1 DataBar Expanded symbols which exceed 48 data characters shall be transmitted as two messages. Each message shall have a symbology identifier prefix of]C1 and neither shall exceed 48 data characters. The messages shall be split between two element strings, see Annex D.

10 Human readable interpretation

When printed, the human readable data shall appear below the symbol in a legible font. The GS1 application specification provides additional requirements for human readable interpretation.

11 Minimum and Maximum width of a module (X)

The minimum and maximum X dimension range should be specified in an application standard.

The X dimension shall be constant throughout a given symbol. When linked to a 2D component, the GS1 DataBar symbol shall have the same X dimension as the 2D component.

12 Application-defined parameters

The parameters of data content, X-dimension, minimum symbol height, minimum symbol quality grade, symbol type, symbol placement, and any other required application parameters are defined by GS1 for each application.

Annex A
(normative)

Check digit calculation

GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, and GS1 DataBar Limited encode the GS1 GTIN-14 numbering structure (which could encode a 14-digit item identification) shown in Table A.1. GS1 DataBar Expanded using application identifier “01” also encodes the GS1 GTIN-14 numbering structure. GS1 DataBar Expanded and GS1 DataBar Expanded Stacked may also encode the SSCC-18 numbering structure by the use of application identifier “00”.

Table A.1 — Check digit calculations of GS1 numbering structures

	Digit positions																	
GS1 GTIN -14					N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}
SSCC-18	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	N_{16}	N_{17}	N_{18}
	Multiply value of each position by																	
	X3	X1	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	x1	x3	
	Total of weighted products = ***Sum***																	
	Subtract ***Sum*** from its value rounded up to the next higher integer multiple of 10, if ***Sum*** is not an integer multiple of 10 = ***Check digit*** ==→																	N

The GS1 check digit is always shown in the human readable interpretation and transmitted by the decoder even though it is not explicitly encoded in most GS1 DataBar symbols.

Annex B
(normative)

C-language element width encoder and decoder

For each (n,k) subset, symbol values are assigned to patterns of element widths. The C encoding routine getRSSwidths calculates the element widths given the subset value. The C decoding routine getRSSvalue calculates the value given the subset element widths. Consecutive values are assigned element subset widths in an ordered sequence.

NOTE The C routine shown still has the original identifiers within it which use the name "RSS". The C routine has not been changed to avoid technical changes to publicly available software.

The sequence of element widths starts with assigning the lower-numbered elements in the subset a width of one module, or the narrowest width that gives a valid subset. (The first pattern, with value 0, will have all but the last element assigned a width of one module unless the last element would exceed the maximum element restriction.) Subsequent values are assigned to the next valid pattern that maintains the narrowest possible lower-numbered elements. For example, the 6 module subset contains the following values 0 through 9 and their pattern of module widths:

value	pattern
0	1 1 1 3
1	1 1 2 2
2	1 1 3 1
3	1 2 1 2
4	1 2 2 1
5	1 3 1 1
6	2 1 1 2
7	2 1 2 1
8	2 2 1 1
9	3 1 1 1

Patterns with element widths wider than the maximum width (maxWidth) are skipped. If patterns with no one-module wide elements are excluded (noNarrow = 0), then these patterns will also be skipped.

```
/**********************************************************************
* getRSSwidths
* routine to generate widths for GS1 DataBar elements for a given value.
*
* Calling arguments:
* val = required value
* n = number of modules
* elements = elements in set (GS1 DataBar Omnidirectional, GS1 DataBar Truncated,
* GS1 * DataBar* Stacked, GS1 DataBar Stacked Omnidirectional & Expanded = 4; GS1 DataBar
* Limited = 7)
* maxWidth = maximum module width of an element
* noNarrow = 0 will skip patterns without a one module wide element
*
* Return:
* static int widths[] = element widths
**********************************************************************/
void getRSSwidths(int val, int n, int elements, int maxWidth, int noNarrow)
{
int bar;
int elmWidth;
int i;
int mxwElement;
int subVal, lessVal;
int narrowMask = 0;
    for (bar = 0; bar < elements-1; bar++)
    {
        for (elmWidth = 1, narrowMask |= (1<<bar);
               ;
                elmWidth++, narrowMask &= ~(1<<bar))
        {
            /* get all combinations */
            subVal = combins(n-elmWidth-1, elements-bar-2);
            /* less combinations with no single-module element */
            if ((!noNarrow) && (narrowMask == 0) &&
                     (n-elmWidth-(elements-bar-1) >= elements-bar-1))
            {
                subVal -= combins(n-elmWidth-(elements-bar), elements-bar-2);
            }
            /* less combinations with elements > maxVal */
            if (elements-bar-1 > 1)
            {
                lessVal = 0;
                for (mxwElement = n-elmWidth-(elements-bar-2);
                        mxwElement > maxWidth;
                        mxwElement--)
                {
                    lessVal += combins(n-elmWidth-mxwElement-1, elements-bar-3);
                }
                subVal -= lessVal * (elements-1-bar);
            }
            else if (n-elmWidth > maxWidth)
            {
                subVal--;
            }
            val -= subVal;
            if (val < 0) break;
        }
        val += subVal;
        n -= elmWidth;
        widths[bar] = elmWidth;
    }
    widths[bar] = n;
    return;
}
```

```
/***************************************************************************
* getRSSvalue
* routine to calculate the subset value given element widths.
*
* Calling arguments:
* widths[] = the given elemt widths
* elements = elements in set (GS1 DataBar Omnidirectional, GS1 DataBar Truncated,
* GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional & Expanded = 4;
*                                                        GS1 DataBar Limited = 7)
* maxWidth = maximum module width of an element
* noNarrow = 0 will skip patterns without a one module wide element
*
* Return:
* the subset value
**************************************************************************/
int getRSSvalue(int widths[], int elements, int maxWidth, int noNarrow)
{
int val = 0;
int n;
int bar;
int elmWidth;
int i;
int mxwElement;
int subVal, lessVal;
int narrowMask = 0;
    for (n = i = 0; i < elements; i++)
    {
        n += widths[i];
    }
    for (bar = 0; bar < elements-1; bar++)
    {
        for (elmWidth = 1, narrowMask |= (1<<bar);
                 elmWidth < widths[bar];
                 elmWidth++, narrowMask &= ~(1<<bar)) {
            /* get all nk combinations */
            subVal = combins(n-elmWidth-1, elements-bar-2);
            /* less combinations with no narrow */
            if ((!noNarrow) && (narrowMask == 0) &&
                        (n-elmWidth-(elements-bar-1)>= elements-bar-1))
            {
                subVal -= combins(n-elmWidth-(elements-bar),
                                elements-bar-2);
            }
            /* less combinations with elements > maxVal */
            if (elements-bar-1 > 1)
            {
                lessVal = 0;
                for (mxwElement = n-elmWidth-(elements-bar-2);
                            mxwElement > maxWidth; mxwElement--)
                {
                    lessVal += combins(n-elmWidth-mxwElement-1,
                                    elements-bar-3);
                }
                subVal -= lessVal * (elements-1-bar);
            }
            else if (n-elmWidth > maxWidth)
            {
                subVal--;

            val += subVal;
        }
        n -= elmWidth;
    }
    return(val);
}
```

```
/**********************************************************************
* combins(n,r): returns the number of Combinations of r selected from n:
*   Combinations = n! / ((n - r)! * r!)
**********************************************************************/
int combins(int n, int r) {
int i, j;
int maxDenom, minDenom;
int val;
    if (n-r > r) {
        minDenom = r;
        maxDenom = n-r;
    }
    else {
        minDenom = n-r;
        maxDenom = r;
    }
    val = 1;
    j = 1;
    for (i = n; i > maxDenom; i--) {
        val *= i;
        if (j <= minDenom) {
            val /= j;
            j++;
        }
    }
    for (; j <= minDenom; j++) {
        val /= j;
    }
    return(val);
}
```

Annex C
(normative)

GS1 DataBar Limited check character element widths

Value	Sequence [1]	Element Widths (from leftmost space to rightmost bar)													
		S1	B1	S2	B2	S3	B3	S4	B4	S5	B5	S6	B6	S7	B7
0	0	1	1	1	1	1	1	1	1	1	1	3	3	1	1
1	1	1	1	1	1	1	1	1	1	1	2	3	2	1	1
2	2	1	1	1	1	1	1	1	1	1	3	3	1	1	1
3	3	1	1	1	1	1	1	1	2	1	1	3	2	1	1
4	4	1	1	1	1	1	1	1	2	1	2	3	1	1	1
5	5	1	1	1	1	1	1	1	3	1	1	3	1	1	1
6	6	1	1	1	1	1	2	1	1	1	1	3	2	1	1
7	7	1	1	1	1	1	2	1	1	1	2	3	1	1	1
8	8	1	1	1	1	1	2	1	2	1	1	3	1	1	1
9	9	1	1	1	1	1	3	1	1	1	1	3	1	1	1
10	10	1	1	1	2	1	1	1	1	1	1	3	2	1	1
11	11	1	1	1	2	1	1	1	1	1	2	3	1	1	1
12	12	1	1	1	2	1	1	1	2	1	1	3	1	1	1
13	13	1	1	1	2	1	2	1	1	1	1	3	1	1	1
14	14	1	1	1	3	1	1	1	1	1	1	3	1	1	1
15	15	1	2	1	1	1	1	1	1	1	1	3	2	1	1
16	16	1	2	1	1	1	1	1	1	1	2	3	1	1	1
17	17	1	2	1	1	1	1	1	2	1	1	3	1	1	1
18	18	1	2	1	1	1	2	1	1	1	1	3	1	1	1
19	19	1	2	1	2	1	1	1	1	1	1	3	1	1	1
20	20	1	3	1	1	1	1	1	1	1	1	3	1	1	1
21	21	1	1	1	1	1	1	1	1	2	1	2	3	1	1
22	22	1	1	1	1	1	1	1	1	2	2	2	2	1	1
23	23	1	1	1	1	1	1	1	1	2	3	2	1	1	1
24	24	1	1	1	1	1	1	1	2	2	1	2	2	1	1
25	25	1	1	1	1	1	1	1	2	2	2	2	1	1	1
26	26	1	1	1	1	1	1	1	3	2	1	2	1	1	1
27	27	1	1	1	1	1	2	1	1	2	1	2	2	1	1
28	28	1	1	1	1	1	2	1	1	2	2	2	1	1	1
29	29	1	1	1	1	1	2	1	2	2	1	2	1	1	1

Value	Sequence [1]	Element Widths (from leftmost space to rightmost bar)													
		S1	B1	S2	B2	S3	B3	S4	B4	S5	B5	S6	B6	S7	B7
30	30	1	1	1	1	1	3	1	1	2	1	2	1	1	1
31	31	1	1	1	2	1	1	1	1	2	1	2	2	1	1
32	32	1	1	1	2	1	1	1	1	2	2	2	1	1	1
33	33	1	1	1	2	1	1	1	2	2	1	2	1	1	1
34	34	1	1	1	2	1	2	1	1	2	1	2	1	1	1
35	35	1	1	1	3	1	1	1	1	2	1	2	1	1	1
36	36	1	2	1	1	1	1	1	1	2	1	2	2	1	1
37	37	1	2	1	1	1	1	1	1	2	2	2	1	1	1
38	38	1	2	1	1	1	1	1	2	2	1	2	1	1	1
39	39	1	2	1	1	1	2	1	1	2	1	2	1	1	1
40	40	1	2	1	2	1	1	1	1	2	1	2	1	1	1
41	41	1	3	1	1	1	1	1	1	2	1	2	1	1	1
42	42	1	1	1	1	1	1	1	1	3	1	1	3	1	1
43	43	1	1	1	1	1	1	1	1	3	2	1	2	1	1
44	45	1	1	1	1	1	1	1	2	3	1	1	2	1	1
45	52	1	1	1	2	1	1	1	1	3	1	1	2	1	1
46	57	1	2	1	1	1	1	1	1	3	1	1	2	1	1
47	63	1	1	1	1	1	1	2	1	1	1	2	3	1	1
48	64	1	1	1	1	1	1	2	1	1	2	2	2	1	1
49	65	1	1	1	1	1	1	2	1	1	3	2	1	1	1
50	66	1	1	1	1	1	1	2	2	1	1	2	2	1	1
51	73	1	1	1	2	1	1	2	1	1	1	2	2	1	1
52	74	1	1	1	2	1	1	2	1	1	2	2	1	1	1
53	75	1	1	1	2	1	1	2	2	1	1	2	1	1	1
54	76	1	1	1	2	1	2	2	1	1	1	2	1	1	1
55	77	1	1	1	3	1	1	2	1	1	1	2	1	1	1
56	78	1	2	1	1	1	1	2	1	1	1	2	2	1	1
57	79	1	2	1	1	1	1	2	1	1	2	2	1	1	1
58	82	1	2	1	2	1	1	2	1	1	1	2	1	1	1
59	126	1	1	1	1	2	1	1	1	1	1	2	3	1	1
60	127	1	1	1	1	2	1	1	1	1	2	2	2	1	1
61	128	1	1	1	1	2	1	1	1	1	3	2	1	1	1
62	129	1	1	1	1	2	1	1	2	1	1	2	2	1	1
63	130	1	1	1	1	2	1	1	2	1	2	2	1	1	1
64	132	1	1	1	1	2	2	1	1	1	1	2	2	1	1
65	141	1	2	1	1	2	1	1	1	1	1	2	2	1	1

Value	Sequence [1]	Element Widths (from leftmost space to rightmost bar)													
		S1	B1	S2	B2	S3	B3	S4	B4	S5	B5	S6	B6	S7	B7
66	142	1	2	1	1	2	1	1	1	1	2	2	1	1	1
67	143	1	2	1	1	2	1	1	2	1	1	2	1	1	1
68	144	1	2	1	1	2	2	1	1	1	1	2	1	1	1
69	145	1	2	1	2	2	1	1	1	1	1	2	1	1	1
70	146	1	3	1	1	2	1	1	1	1	1	2	1	1	1
71	210	1	1	2	1	1	1	1	1	1	1	2	3	1	1
72	211	1	1	2	1	1	1	1	1	1	2	2	2	1	1
73	212	1	1	2	1	1	1	1	1	1	3	2	1	1	1
74	213	1	1	2	1	1	1	1	2	1	1	2	2	1	1
75	214	1	1	2	1	1	1	1	2	1	2	2	1	1	1
76	215	1	1	2	1	1	1	1	3	1	1	2	1	1	1
77	216	1	1	2	1	1	2	1	1	1	1	2	2	1	1
78	217	1	1	2	1	1	2	1	1	1	2	2	1	1	1
79	220	1	1	2	2	1	1	1	1	1	1	2	2	1	1
80	316	2	1	1	1	1	1	1	1	1	2	2	2	1	1
81	317	2	1	1	1	1	1	1	1	1	3	2	1	1	1
82	318	2	1	1	1	1	1	1	2	1	1	2	2	1	1
83	319	2	1	1	1	1	1	1	2	1	2	2	1	1	1
84	320	2	1	1	1	1	1	1	3	1	1	2	1	1	1
85	322	2	1	1	1	1	2	1	1	1	2	2	1	1	1
86	323	2	1	1	1	1	2	1	2	1	1	2	1	1	1
87	326	2	1	1	2	1	1	1	1	1	2	2	1	1	1
88	337	2	1	1	1	1	1	1	1	2	2	1	2	1	1

NOTE 1 Sequence is the sequence number in the 21 × 21 = 441 (numbered 0 to 440) pattern sequence generated by "getRSSwidths" in Annex B for all 8 module combinations for the first 6 spaces and the first 6 bars. The sequence is calculated as ("space pattern value" × 21) + "bar pattern value". A list of the sequence numbers could be used instead of a list of the bar and space widths to define the check character set.

The calling arguments for getRSSwidths to generate the bars and spaces is:

val	=	"space pattern value" or "bar pattern value" for space or bar (equal to 0 to 20),
n	=	8,
elements	=	6,
maxWidth	=	3,
noNarrow	=	1.

Annex D
(normative)

Splitting long GS1 DataBar Expanded or GS1 DataBar Expanded Stacked symbols for GS1 emulation mode

GS1 DataBar Expanded and GS1 DataBar Expanded Stacked symbols which exceed 48 data characters must be split in two in GS1 emulation mode before transmission. The split shall be made at the start of the element string that would cause the data transmission to exceed the 48 character limit. The element strings are determined either by a FNC1 separator or at the end of a fixed length AI element string listed in Table D.1. The reader shall perform the following steps to find the element string that would exceed 48 characters:

a) If the data in the GS1 DataBar Expanded or GS1 DataBar Expanded Stacked symbol exceeds 48 characters, start at the beginning of the symbol.

b) Check the next two digits of the data in the linear component (the first two digits of the AI).

c) If the two digits are found in Table D.1, skip the number of characters indicated in the entry for the AI in Table D.1.

d) Otherwise the AI is not in the table so it is not a predefined fixed-length AI. Search forward in the data for the first occurrence of a FNC1 or until the end of the symbol, whichever occurs first.

e) If the number of characters exceeds the 48 character limit, split the data at the beginning of the last element string processed. Otherwise go to step b).

If the first message would end in a FNC1, the FNC1 is not transmitted.

Table D.1 — Application identifiers with predefined length

First two digits of the application identifier	Number of characters (Application identifier and data field)	First two digits of the application identifier	Number of characters (Application identifier and data field)
00	20	18	8
01	16	19	8
02	16	20	4
03	16	23	2n+4 [1]
04	18	31	10
11	8	32	10
12	8	33	10
13	8	34	10
14	8	35	10
15	8	36	10
16	8	41	16
17	8		

NOTE 1 Where n is the third digit of the application identifier (directly following the 23).

Only the application identifiers with the first two digits matching those listed in Table D.1 are considered as “fixed length”. Other application identifiers may be defined with element string data that is naturally only one length, but these element strings shall still be considered “variable length” because they are not in the predefined length table. No additions will be made to this table in the future.

Annex E
(informative)
Symbol elements

Table E.1, Table E.2, and Table E.3 described all the elements in sequence in the various types of GS1 DataBar symbols.

Table E.1 — GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Stacked and GS1 DataBar Stacked Omnidirectional element description

Element	Type	Description
1	space	left guard pattern outside element (one module wide)
2	bar	left guard pattern inside element (one module wide)
3	space	symbol character 1, odd element no. 1
4	bar	symbol character 1, even element no. 1
5	space	symbol character 1, odd element no. 2
6	bar	symbol character 1, even element no. 2
7	space	symbol character 1, odd element no. 3
8	bar	symbol character 1, even element no. 3
9	space	symbol character 1, odd element no. 4
10	bar	symbol character 1, even element no. 4
11	space	left check character element no. 1
12	bar	left check character element no. 2
13	space	left check character element no. 3
14	bar	left check character element no. 4 (one module wide)
15	space	left check character element no. 5 (one module wide)
16	bar	symbol character 2, even element no. 4
17	space	symbol character 2, odd element no. 4
18	bar	symbol character 2, even element no. 3
19	space	symbol character 2, odd element no. 3
20	bar	symbol character 2, even element no. 2
21	space	symbol character 2, odd element no. 2
22	bar	symbol character 2, even element no. 1
23	space	symbol character 2, odd element no. 1
24	bar	symbol character 4, odd element no. 1
25	space	symbol character 4, even element no. 1
26	bar	symbol character 4, odd element no. 2
27	space	symbol character 4, even element no. 2
28	bar	symbol character 4, odd element no. 3
29	space	symbol character 4, even element no. 3
30	bar	symbol character 4, odd element no. 4
31	space	symbol character 4, even element no. 4
32	bar	right check character element no. 5 (one module wide)
33	space	right check character element no. 4 (one module wide)
34	bar	right check character element no. 3
35	space	right check character element no. 2
36	bar	right check character element no. 1
37	space	symbol character 3, even element no. 4
38	bar	symbol character 3, odd element no. 4
39	space	symbol character 3, even element no. 3
40	bar	symbol character 3, odd element no. 3
41	space	symbol character 3, even element no. 2
42	bar	symbol character 3, odd element no. 2
43	space	symbol character 3, even element no. 1
44	bar	symbol character 3, odd element no. 1
45	space	right guard pattern inside element (one module wide)
46	bar	right guard pattern outside element (one module wide)

Table E.2 — GS1 DataBar Limited element description

Element	Type	Description
1	space	left guard pattern outside element (one module wide)
2	bar	left guard pattern inside element (one module wide)
3	space	left symbol character, symbol character element no 1, odd element no. 1
4	bar	left symbol character, symbol character element no 2, even element no. 1
5	space	left symbol character, symbol character element no 3, odd element no. 2
6	bar	left symbol character, symbol character element no 4, even element no. 2
7	space	left symbol character, symbol character element no 5, odd element no. 3
8	bar	left symbol character, symbol character element no 6, even element no. 3
9	space	left symbol character, symbol character element no 7, odd element no. 4
10	bar	left symbol character, symbol character element no 8, even element no. 4
11	space	left symbol character, symbol character element no 9, odd element no. 5
12	bar	left symbol character, symbol character element no 10, even element no. 5
13	space	left symbol character, symbol character element no 11, odd element no. 6
14	bar	left symbol character, symbol character element no 12, even element no. 6
15	space	left symbol character, symbol character element no 13, odd element no. 7
16	bar	left symbol character, symbol character element no 14, even element no. 7
17	space	check character element no. 1
18	bar	check character element no. 2
19	space	check character element no. 3
20	bar	check character element no. 4
21	space	check character element no. 5
22	bar	check character element no. 6
23	space	check character element no. 7
24	bar	check character element no. 8
25	space	check character element no. 9
26	bar	check character element no. 10
27	space	check character element no. 11
28	bar	check character element no. 12
29	space	check character element no. 13 (one module wide)
30	bar	check character element no. 14 (one module wide)
31	space	right symbol character, symbol character element no 1, odd element no. 1
32	bar	right symbol character, symbol character element no 2, even element no. 1
33	space	right symbol character, symbol character element no 3, odd element no. 2
34	bar	right symbol character, symbol character element no 4, even element no. 2
35	space	right symbol character, symbol character element no 5, odd element no. 3
36	bar	right symbol character, symbol character element no 6, even element no. 3
37	space	right symbol character, symbol character element no 7, odd element no. 4
38	bar	right symbol character, symbol character element no 8, even element no. 4
39	space	right symbol character, symbol character element no 9, odd element no. 5
40	bar	right symbol character, symbol character element no 10, even element no. 5
41	space	right symbol character, symbol character element no 11, odd element no. 6
42	bar	right symbol character, symbol character element no 12, even element no. 6
43	space	right symbol character, symbol character element no 13, odd element no. 7
44	bar	right symbol character, symbol character element no 14, even element no. 7
45	space	right guard pattern inside element (one module wide)
46	bar	right guard pattern bar element (one module wide or at least 5 modules wide)
47	Space	right guard pattern outside element (5 modules)

Table E.3 — GS1 DataBar Expanded (six segment) element description

Element	Type	Description
1	space	left guard pattern outside element (one module wide)
2	bar	left guard pattern inside element (one module wide)
3	space	symbol character 1, odd element no. 1 (restricted to 4 modules or fewer)
4	bar	symbol character 1, even element no. 1
5	space	symbol character 1, odd element no. 2
6	bar	symbol character 1, even element no. 2
7	space	symbol character 1, odd element no. 3
8	bar	symbol character 1, even element no. 3
9	space	symbol character 1, odd element no. 4
10	bar	symbol character 1, even element no. 4
11	space	finder pattern A1 element no. 1
12	bar	finder pattern A1 element no. 2
13	space	finder pattern A1 element no. 3
14	bar	finder pattern A1 element no. 4 (one module wide)
15	space	finder pattern A1 element no. 5 (one module wide)
16	bar	symbol character 2, even element no. 4
17	space	symbol character 2, odd element no. 4
18	bar	symbol character 2, even element no. 3
19	space	symbol character 2, odd element no. 3
20	bar	symbol character 2, even element no. 2
21	space	symbol character 2, odd element no. 2
22	bar	symbol character 2, even element no. 1
23	space	symbol character 2, odd element no. 1 (restricted to 4 modules or fewer)
24	bar	symbol character 3, odd element no. 1 (restricted to 4 modules or fewer)
25	space	symbol character 3, even element no. 1
26	bar	symbol character 3, odd element no. 2
27	space	symbol character 3, even element no. 2
28	bar	symbol character 3, odd element no. 3
29	space	symbol character 3, even element no. 3
30	bar	symbol character 3, odd element no. 4
31	space	symbol character 3, even element no. 4
32	bar	finder pattern B2 element no. 5 (one module wide)
33	space	finder pattern B2 element no. 4 (one module wide)
34	bar	finder pattern B2 element no. 3
35	space	finder pattern B2 element no. 2
36	bar	finder pattern B2 element no. 1
37	space	symbol character 4, even element no. 4
38	bar	symbol character 4, odd element no. 4
39	space	symbol character 4, even element no. 3
40	bar	symbol character 4, odd element no. 3
41	space	symbol character 4, even element no. 2
42	bar	symbol character 4, odd element no. 2
43	space	symbol character 4, even element no. 1
44	bar	symbol character 4, odd element no. 1 (restricted to 4 modules or fewer)
45	space	symbol character 5, odd element no. 1 (restricted to 4 modules or fewer)
46	bar	symbol character 5, even element no. 1
47	space	symbol character 5, odd element no. 2
48	bar	symbol character 5, even element no. 2

Element	Type	Description
49	space	symbol character 5, odd element no. 3
50	bar	symbol character 5, even element no. 3
51	space	symbol character 5, odd element no. 4
52	bar	symbol character 5, even element no. 4
53	space	finder pattern B1 element no. 1
54	bar	finder pattern B1 element no. 2
55	space	finder pattern B1 element no. 3
56	bar	finder pattern B1 element no. 4 (one module wide)
57	space	finder pattern B1 element no. 5 (one module wide)
58	bar	symbol character 6, even element no. 4
59	space	symbol character 6, odd element no. 4
60	bar	symbol character 6, even element no. 3
61	space	symbol character 6, odd element no. 3
62	bar	symbol character 6, even element no. 2
63	space	symbol character 6, odd element no. 2
64	bar	symbol character 6, even element no. 1
65	space	symbol character 6, odd element no. 1 (restricted to 4 modules or fewer)
66	bar	right guard pattern inside element (one module wide)
67	space	right guard pattern outside element (one module wide)

Annex F
(informative)

Encoding examples

F.1 GS1 DataBar Omnidirectional and GS1 DataBar Truncated

The GS1 DataBar Omnidirectional linear component in the GS1 Composite symbol of Figure F.1 below encodes a linkage flag of 1 (indicating the presence of a 2D component) and an item number of 24012345678905.

Figure F.1 — GS1 DataBar Omnidirectional Composite symbol example

The element widths of the GS1 DataBar Omnidirectional linear component are calculated by the following steps:

a) The symbol value is linkage flag 10000000000000 + item no. 2401234567890 = 12401234567890 (the check digit 5 is dropped).

b) The left and right symbol character pair values are:

left = 12 401 234 567 890 div 4 537 077 = 2 733 309.

right = 12 401 234 567 890 mod 4 537 077 = 1 170 097.

c) The four symbol characters are:

data1 = left div 1 597 = 2 733 309 div 1 597 = 1 711,

data2 = left mod 1 597 = 2 733 309 mod 1 597 = 842,

data3 = right div 1 597 = 1 170 097 div 1 597 = 732,

data4 = right mod 1 597 = 1 170 097 mod 1 597 = 1 093.

d) The odd and even subset values for the four symbol characters are:

data1 is (16,4) and the value 1 711 is in Group 3 with 8/8 odd/even modules. Using the equations for V_{ODD} and V_{EVEN} from the value of data1,

V_{ODD1} = (data1 – 961) div 34 = (1 711 – 961) div 34 = 750 div 34 = 22,

V_{EVEN1} = (data1 – 961) mod 34 = (1 711 – 961) mod 34 = 750 mod 34 = 2,

data2 is (15,4) and the value 842 is in Group 2 with 7/8 odd/even modules. Using the equations for V_{EVEN} and V_{ODD} from the value of data2,

V_{EVEN2} = (data2 – 336) div 20 = (842 – 336) div 20 = 506 div 20 = 25,

V_{ODD2} = (data2 – 336) mod 20 = (842 – 336) mod 20 = 506 mod 20 = 6, data3 is (16,4) and the value 732 is in Group 2 with 10/6 odd/even modules. Using the equations for V_{ODD} and V_{EVEN} from the value of data3,

V_{ODD3} = (data3 – 161) div 10 = (732 – 161) div 10 = 571 div 10 = 57,

V_{EVEN3} = (data3 – 161) mod 10 = (732 – 161) mod 10 = 571 mod 10 = 1,

data4 is (15,4) and the value 1 093 is in Group 3 with 9/6 odd/even modules. Using the equations for V_{EVEN} and V_{ODD} from the value of data4,

V_{EVEN4} = (data4 – 1 036) div 48 = (1 093 – 1 036) div 48 = 57 div 48 = 1,

V_{ODD4} = (data4 – 1 036) mod 48 = (1 093 – 1 036) mod 48 = 57 mod 48 = 9.

e) The GS1 DataBar subset width algorithm in Annex B generates the following widths from the subset values:

odd 1 (value 22) = 3 1 1 3

even 1 (value 2) = 1 1 3 3

therefore data 1 widths = 3 1 1 1 1 3 3 3

odd 2 (value 6) = 1 2 3 1

even 2 (value 25) = 3 1 1 3

therefore data 2 widths = 1 3 2 1 3 1 1 3 (left-to-right mirrored)

odd 3 (value 57) = 3 3 3 1

even 3 (value 1) = 1 1 2 2

therefore data 3 widths = 3 1 3 1 3 2 1 2 (left-to-right mirrored)

odd 4 (value 9) = 1 2 4 2

even 4 (value 1) = 1 1 2 2

therefore data 4 widths = 1 1 2 1 4 2 2 2.

f) Calculate the checksum:

data 1: $3\times1 + 1\times3 + 1\times9 + 1\times27 + 1\times2 + 3\times6 + 3\times18 + 3\times54$ = 278

data 2: $1\times4 + 3\times12 + 2\times36 + 1\times29 + 3\times8 + 1\times24 + 1\times72 + 3\times58$ = 435

data 3: $3\times16 + 1\times48 + 3\times65 + 1\times37 + 3\times32 + 2\times17 + 1\times51 + 2\times74$ = 657

data 4: $1\times64 + 1\times34 + 2\times23 + 1\times69 + 4\times49 + 2\times68 + 2\times46 + 2\times59$ = 755

2 125

Therefore the checksum = 2 125 mod 79 = 71.

g) Calculate the two check characters from the checksum:

71 is greater than or equal to 8 so the intermediate finder pair value is 71 + 1 = 72

72 is greater than or equal to 72 so the finder pair value is 72 + 1 = 73

left check is 73 div 9 = 8

right check is 73 mod 9 = 1

left check 8 = elements 1 3 9 1 1

right check 1 = elements 3 5 5 1 1 (left-to-right mirrored).

h) The element widths of the symbol are (left guard, symbol character 1, left check, symbol character 2 (mirrored), symbol character 4, right check (mirrored), symbol character 3 (mirrored), and right guard):

1 1, 3 1 1 1 1 3 3 3, 1 3 9 1 1, 3 1 1 3 1 2 3 1, 1 1 2 1 4 2 2 2, 1 1 5 5 3, 2 1 2 3 1 3 1 3, 1 1.

F.2 GS1 DataBar Limited

The GS1 DataBar Limited symbol in Figure F.2 below encodes an item number of 00098765432105.

Figure F.2 — GS1 DataBar Limited symbol example

The element widths of the GS1 DataBar Limited symbol are calculated by the following steps:

a) The symbol value is the item no. 00098765432105 = 9876543210 (the check digit 5 is dropped).

b) The left and right symbol character values are:

left = 9 876 543 210 div 2 013 571 = 4 904.

right = 9 876 543 210 mod 2 013 571 = 1 991 026.

c) The odd and even subset values for the two symbol characters are:

left symbol character with the value 4 904 is in class 1 with 17/9 odd/even modules, therefore,

left odd = (left data – 0) div 28 = 4 904 div 28 = 175,

left even = (left data – 0) mod 28 = 4 904 mod 28 = 4,

right symbol character with the value 1 991 026 is in class 6 with 19/7 odd/even modules, therefore,

right odd = (right data – 1 979 845) div 1 = 11 181 div 1 = 11 181,

right even = (right data – 1 979 845) mod 1 = 11 181 mod 1 = 0.

d) The GS1 DataBar subset width algorithm in Annex B generates the following widths from the subset values:

left odd (value 175) = 1 1 2 2 2 4 5

left even (value 4) = 1 1 1 1 2 2 1

therefore left data widths = 1 1 1 1 2 1 2 1 2 2 4 2 5 1

right odd (value 11 181) = 3 3 1 3 5 2 2

right even (value 0) = 1 1 1 1 1 1 1

therefore right data widths = 3 1 3 1 1 1 3 1 5 1 2 1 2 1.

e) Calculate the checksum:

left = $1\times1 + 1\times3 + 1\times9 + 1\times27 + 2\times81 + 1\times65 + 2\times17 + 1\times51 + 2\times64 + 2\times14 + 4\times42 + 2\times37 + 5\times22 + 1\times66 = 926$

right = $3\times20 + 1\times60 + 3\times2 + 1\times6 + 1\times18 + 1\times54 + 3\times73 + 1\times41 + 5\times34 + 1\times13 + 2\times39 + 1\times28 + 2\times84 + 1\times74 = 995$

Therefore the checksum = (926 + 995) mod 89 = 52.

f) The check character element widths in Annex B for value 52 = 1 1 1 2 1 1 2 1 1 2 2 1 1 1.

g) The element widths of the symbol are (left guard, left symbol character, check character, right symbol character, and right guard):

1 1, 1 1 1 1 2 1 2 1 2 2 4 2 5 1, 1 1 1 2 1 1 2 1 1 2 2 1 1 1, 3 1 3 1 1 1 3 1 5 1 2 1 2 1, 1 1 5.

F.3 GS1 DataBar Expanded

The GS1 DataBar Expanded symbol in Figure F.3 below encodes the AI element string (10)12A. This data in this example contains no primary item identification but is for purely instructional purposes.

Figure F.3 — GS1 DataBar Expanded symbol example

The element widths of the GS1 DataBar Expanded symbol widths are calculated by the following steps:

a) The symbol encodes 1012A.

b) The bit fields are:

linkage flag bit	=	0 (no 2D component)
encodation method	=	00 (no specific AI is compressed, but any AI's to be encodd literally)
variable length bits	=	00 (even segments, group 1)
data bits:	=	0010011 (numeric encodation for the pair of digits '10')
	=	0010101 (numeric encodation for the pair of digits '12')
		0000 (alphanumeric latch)
	=	100000 (alphanumeric encodation for the character 'A')
	=	0010000 (for the pad)

Therefore the data string is 000000010011001010100001000000010000.

NOTE This example follows the method described in clause 7.2.5.

c) The data grouped in 12-bit values are 000000010011, 001010100001, and 000000010000. The three symbol characters values are:

data 1 = 000000010011 = 19

data 2 = 001010100001 = 673

data 3 = 000000010000 = 16.

d) The odd and even subset values for the three symbol characters are:

data 1 with the value 19 is in class 1 with 12/5 odd/even modules, therefore,

odd 1 = (data 1 – 0) div 4 = 19 div 4 = 4,

even 1 = (data 1 – 0) mod 4 = 19 mod 4 = 3,

data 2 with the value 673 is in class 2 with 10/7 odd/even modules, therefore,