ICS 35.0
A 24

中华人民共和国国家标准

GB/T 21335—2008

RSS 条码

Reduced Space Symbology（RSS）bar code

2008-01-09 发布　　2008-08-01 实施

中华人民共和国国家质量监督检验检疫总局
中国国家标准化管理委员会　发布

目　次

前　言

本标准与国际标准 ISO/IEC 24724《信息技术　自动识别与数据采集技术　缩小空间条码(RSS)码制规范》在技术内容上保持一致。

本标准的附录 A、附录 B、附录 C、附录 D 为规范性附录，附录 E、附录 F、附录 G、附录 H、附录 I、附录 J 为资料性附录。

本标准由全国物流信息管理标准化技术委员会提出并归口。

本标准由中国物品编码中心负责起草，北京网路畅想科技发展有限公司参加起草。

本标准主要起草人：张成海、赵辰、吴宏、熊立勇、吴娟、孔洪亮、刘伟、张铎。

RSS 条码

1 范围

本标准规定了 RSS 条码符号的结构、数据符编码、尺寸、印制质量要求、校验方法和译码算法。

本标准适用于采用 RSS 条码符号的贸易项目及贸易项目附加信息的标识。

2 规范性引用文件

下列文件中的条款通过本标准的引用而成为本标准的条款。凡是注日期的引用文件，其随后所有的修改单(不包括勘误的内容)或修订版均不适用于本标准，然而，鼓励根据本标准达成协议的各方研究是否可使用这些文件的最新版本。凡是不注日期的引用文件，其最新版本适用于本标准。

GB/T 1988 信息技术 信息交换用七位编码字符集(GB 1988—1989，eqv ISO/IEC 646:1991)

GB 12904 商品条码(GB 12904—2003，ISO/IEC 15420:2000，NEQ)

GB/T 12905 条码术语

GB/T 14258 信息技术 自动识别与数据采集技术 条码符号印制质量的检验(GB/T 14258—2003，ISO/IEC 15416:2000，MOD)

GB/T 15425 EAN·UCC 系统 128 条码(GB/T 15425—2002，EAN·UCC，NEQ)

GB/T 16986 EAN·UCC 系统应用标识符(GB/T 16986—2003，ISO/IEC 15418:1999，NEQ)

ISO/IEC 15424 信息技术 自动识别与数据采集技术 数据载体标识符(包括码制标识符)

ISO/IEC 24723 信息技术 自动识别与数据采集技术 EAN·UCC 复合码规范

GS1 通用规范

3 术语、定义和数学运算符

3.1 术语和定义

GB/T 12905 确立的以及下列术语和定义适用于本标准。

3.1.1

复合码 composite bar code

由一维条码和二维条码组合成的条码符号。

3.1.2

一维部分 linear component

EAN·UCC 复合码中用于对贸易项目标识的一维条码部分。

3.1.3

二维部分 2D component

EAN·UCC 复合码中用于对贸易项目的附加信息(如批号、有效期等)标识的二维条码部分。

3.1.4

全球贸易项目代码 global trade item number (GTIN)

用于全世界范围内贸易项目的唯一标志关键字，包括 14 位、13 位、12 位和 8 位数字的代码(GTIN-14、GTIN-13、GTIN-12 和 GTIN-8)。GTIN-14 以 EAN/UCC-14 标准结构编码形成；GTIN-13、GTIN-12 和 GTIN-8 分别以商品标志代码(见 GB 12904)EAN/UCC-13、EAN/UCC-12 和 EAN/UCC-8 结构编码形成。在计算机字段中全球贸易项目代码必须表示为 14 位数字的标准字段(关键字)，GTIN-13、GTIN-12 和 GTIN-8 需在前面适当补 0 形成 14 位数字的标准字段。

3.1.5

指示符 indicator digit

14 位全球贸易项目代码的第一位数字，用于区分相同贸易项目不同组合的包装或指明变量贸易项目。

3.1.6

连接标志 linkage flag

在作为一维部分的 RSS 或 UCC/EAN-128 条码(见 GB/T 15425)中，表示是否连接二维部分的指示符。

3.1.7

段 segment

条码符号的最小可译码单元。在 RSS 条码符号中，一个段由一个符号字符和与它相邻的定位符组成。

3.1.8

表决 voting

一种译码技术，被译码段的值连同被译码的次数的计数一起存储，最终选出被译码次数最多的被译码段的值。表决是用于以段为单位进行译码的译码技术，在全向扫描中，表决用于通过段对 RSS 进行译码。

3.2 数学运算符

本标准使用下列数学运算符：

div 取整数商运算符 整数除法舍弃余数的运算

mod 模运算符 整数除法只取余数的运算

4 RSS 条码描述

4.1 RSS 条码的类型

RSS 系列条码符号有 RSS-14、限定式 RSS 和扩展式 RSS 三种类型。

其中：RSS-14 包括标准 RSS-14(简称 RSS-14)、截短式 RSS-14、层排式 RSS-14、全向层排式 RSS-14；扩展式 RSS 包括单行扩展式 RSS(简称扩展式 RSS)和层排扩展式 RSS。

标准 RSS-14、全向层排式 RSS-14、扩展式 RSS、层排扩展式 RSS 可用于全向扫描器按段进行识读。

RSS 系列条码符号的特点总汇参见附录 J。

4.2 RSS 条码的特点

RSS 系列条码符号的特点包括：

a) 可编码字符集

1) RSS-14 和限定式 RSS：数字 0～9。

2) 扩展式 RSS：信息交换用 7 位编码字符集(见 GB/T 1988)的一个子集，其中包括全部英文大、小写字母，数字和选出的 21 个标点符号(含空格符号)。另外还有一个特殊功能字符 FNC1。

b) 符号字符结构

每种类型的符号采用不同的(n,k)结构，每个符号字符是 n 个模块宽，由 k 个条和 k 个空组成。

c) 符号类型

连续型一维条码符号。

d) 最大数据容量

1) RSS-14 和限定式 RSS：应用标识符(见 GB/T 16986)"01"加 14 位数字的项目标识代码。

2) 扩展式 RSS：74 个数字或 41 个字母字符。

注：最大数据容量包括适合的隐含的应用标识符，但不包括 FNC1 字符。扩展式 RSS 的数据容量取决于编码方法，对于 AI(01)+其他 AI 数据串，最大数据容量为 74 个数字；对于所有其他 AI 数据串，最大数据容量为 70 个数字；对于 AI(01)+(392$_x$)+所有其他 AI 数据串，最大数据容量为 77 个数字。

e) 错误校验

1) RSS-14：模 79 校验值。

2) 限定式 RSS：模 89 校验值。

3) 扩展式 RSS：模 211 校验值。

f) 具有字符自校验功能。

g) 可双向译码。

4.3 附加特征

RSS 条码符号系列有以下附加特征：

a) 数据压缩：RSS 系列条码符号的每种符号都具有对数据串进行优化的数据压缩方法。扩展式 RSS 选用常用应用标志符 AI 优化的特定组合。

b) 各部分的连接：所有的 RSS 符号都包含连接标志。如果连接标志是"0"，表示 RSS 符号是独立的。如果连接标志是"1"，表示 RSS 符号连接有一个二维部分及相应的分隔符。

c) UCC/EAN-128 条码模拟：识读器设定在 UCC/EAN-128 模拟方式时，传输 RSS 系列符号的编码数据如同传输一个或多个 UCC/EAN-128 符号的编码数据。

4.4 符号结构

每个 RSS 符号包括外侧保护符、数据符和定位符。每个符号具有一种错误校验方法。

符号两端的保护符各由一个条/空对或一个空/条对的两个单模块单元组成。层排式 RSS-14 和层排扩展式 RSS 符号在符号每行的两端都有保护符。附录 I.1 给出了外侧保护符单元的印刷注意事项。

每个符号都有两个或多个数据符，每个数据符都采用(n,k)结构。数据符值通过数学运算形成确切的编码数据。

定位符是选择出来的一组条空组合，供识读器识别和确认符号，确定单元的相对位置。每个符号包含一个或多个定位符，定位符还具有校验符和段标识符的作用。

所有 RSS 符号都包含一个连接标志。如果连接标志是"1"，则作为一维部分的 RSS 及与其相邻的复合码分隔符应该按照 ISO/IEC 24723 的要求与二维部分对齐。

5 RSS-14 条码符号的要求

5.1 RSS-14 的基本特点

RSS-14 条码符号能够对 20000000000000(2×10^{13})个数值进行编码。每个数值由 14 位数字组成，第一位是连接标志，当数值大于或等于 10000000000000 时，则连接标志为"1"。后 13 位数字加上一个隐含的校验码就构成了包含包装指示符的 14 位全球贸易项目代码。例如，数值 10001234567890 对应的贸易项目代码为 00012345678905。

RSS-14 符号可分 4 个段被扫描和译码，然后进行重组，这适于全向扫描。图 1 为一个独立的标准 RSS-14 符号。

图 1 表示数据(01)20012345678909 的 RSS-14 符号

注：图 1 的数据中，开头的(01)是隐含的应用标志符，不在符号中编码；最后一位数字 9 是一个按模 10 计算的校验码，不在符号中编码。校验码的计算见附录 A。附录 F.1 为一个 RSS-14 符号编码的完整例子。

5.2 RSS-14 的符号结构

RSS-14 符号分为 8 个区域，共 96 个模块。8 个区域从左到右依次为：左侧保护符、数据符 1、左侧

定位符、数据符2、数据符4、右侧定位符、数据符3和右侧保护符。结构如图2所示。8个区域的条、空及模块组成如下：

a) 左侧保护符：由1个模块宽的空和1个模块宽的条组成；

b) 数据符1：由4个空和4个条共16个模块组成，$(n,k)=(16,4)$；

c) 左侧定位符：由3个空和2个条共15个模块组成；

d) 数据符2：由4个条和4个空共15个模块组成，$(n,k)=(15,4)$；

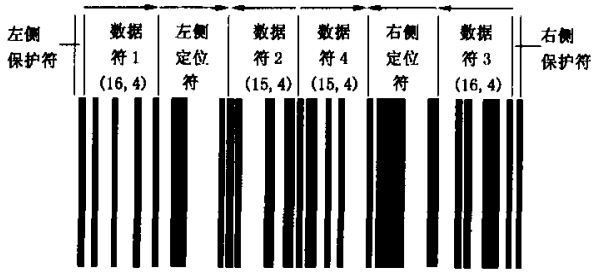
e) 数据符4：由4个条和4个空共15个模块组成，$(n,k)=(15,4)$；

f) 右侧定位符：由3个条和2个空共15个模块组成；

g) 数据符3：由4个空和4个条共16个模块组成，$(n,k)=(16,4)$；

h) 右侧保护符：由1个模块宽的空和1个模块宽的条组成。

数据符1与数据符2为左侧数据符对；数据符3与数据符4为右侧数据符对。



注：图中箭头表示每个字符单元排序的方向。

图2 表示数据(01)04412345678909的RSS-14条码符号

整个符号包括46个单元，共96个模块，参见附录E.1。在全向扫描识读环境中，RSS-14条码符号的符号高度应大于或等于33个模块宽度(X尺寸)。

注：对于(n,k)条码，模块宽度即X尺寸(X)。

RSS-14条码符号不需要空白区。如果最左边的空或最右边的条的颜色与相邻的背景区域颜色相同，那么第1个或最后1个单元可能会出现宽于1个模块的情况，但不会影响符号的识读。

5.2.1 数据符结构

每个数据符的模块组配为(n,k)结构，其中n为模块数，k是组成数据符的条空对的数目。数据符1和数据符3(又称外侧数据符)的n值是16，k值是4；数据符2和数据符4(又称内侧数据符)的n值是15，k值是4。

在图2中，箭头表明每个字符的单元排序的方向。数据符1和数据符4的单元从左到右排序，数据符2和数据符3的单元从右到左排序。因此，数据符的单元是朝着相邻的定位符排序的。

每个数据符的单元集合包含奇和偶两个子集，这里的奇和偶指的是每个子集中单元序号的奇、偶。例如，奇子集由第1、第3、第5和第7单元组成，在数据符1和数据符2中，奇数单元是空，偶数单元是条。在数据符3和数据符4中，奇数单元是条，偶数单元是空。

5.2.2 数据符值

对每个数据符值，由一个算法给出奇子集和偶子集中各单元宽度的值(以模块为单位)组成的序列，需要为这个算法提供单元数量、模块数量、单元最大宽度以及子集中所有的单元是否都比1个模块宽等信息。附录B给出了用C语言程序实现的生成RSS-14数据符单元的算法。

5.2.2.1 外侧数据符值

外侧数据符的有效的偶子集应至少有1个单个模块宽的单元，有效的奇子集则不必要具有1个单个模块宽的单元。对偶数单元的上述限制保证了数据符的各个边缘到相似边缘距离(条加空与空加条)有唯一的模块和。

表1描述了(16,4)子集的特征，列出了5组奇子集和偶子集对。2个子集都具有偶数个模块。最宽的单元的宽度被规定下来，保证在1对相邻单元中的模块数之和不会超过9。在有限制要求的情况下，构成1个(16,4)字符的条空组合的模式共有2841种，能表示2841个值。

表1 外侧数据符(16,4)的特征

数据符值的范围	组	前面各组所能表示的数据符值总数(G_{SUM})	奇/偶子集模块数	奇数/偶数最宽单元	奇子集值的总数(T_{ODD})	偶子集值的总数(T_{EVEN})
0～160	1	0	12/4	8/1	161	1
161～960	2	161	10/6	6/3	80	10
961～2014	3	961	8/8	4/5	31	34
2015～2714	4	2015	6/10	3/6	10	70
2715～2840	5	2715	4/12	1/8	1	126

(16,4)数据符值V_D与奇子集的值及偶子集的值的关系见下式：

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

式中：

T_{EVEN}——偶子集值的总数；

V_{ODD}——奇子集的值；

V_{EVEN}——偶子集的值；

G_{SUM}——前面各组所能表示的字符值的总数。

通过下面两式把外侧数据符的值V_D编码为V_{ODD}和V_{EVEN}：

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

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

例如：对一个值为2315的(16,4)数据符进行编码。从表1可知，数据符的值在第4组内，因此$G_{SUM}=2015$，$T_{EVEN}=70$。应用上面的公式：

$$V_{ODD}=(2315-2015)\text{div } 70=300 \text{ div } 70=4$$

$$V_{EVEN}=(2315-2015)\text{mod } 70=300 \text{ mod } 70=20$$

数据符值2315在第4组中(见表1)，该数据符有：一个6个模块的奇子集，子集值4是10个连续值(0～9)中的一个；一个10个模块的偶子集，子集值20是70个连续值(0～69)中的一个。使用附录B中的程序可以得到：该数据符各奇数单元的宽度是{1 2 2 1}，各偶数单元的宽度是{1 5 1 3}，整个字符各单元的宽度为{1 1 2 5 2 1 1 3}，单元宽度序列朝向相邻定位符排序。

5.2.2.2 内侧数据符值

内侧数据符有效的奇子集应至少有一个单个模块宽的单元，有效的偶子集则不必具有一个单个模块宽的单元。对奇数单元的上述限制保证数据符的各个边缘到相似边缘距离有唯一的模块和。

表2描述了(15,4)子集的特征，列出了4组奇子集和偶子集对。奇子集具有奇数个模块，偶子集具有偶数个模块。最宽的单元的宽度被规定下来，保证在一对相邻单元中的模块数之和不会超过9。在有限制要求的情况下，构成一个(15,4)字符的条空组合的模式共有1597种，能表示1597个值。奇子集允许值的范围被限定下来，保证奇数单元序号为1的单元宽度不超过4个模块。

表 2　内侧数据符(15,4)的特征

数据符值的范围	组	前面各组所能表示的数据符值总数(G_{SUM})	奇/偶子集模块数	奇数/偶数最宽单元	奇子集值的总数(T_{ODD})	偶子集值的总数(T_{EVEN})
0～335	1	0	5/10	2/7	4	84
336～1035	2	336	7/8	4/5	20	35
1036～1515	3	1036	9/6	6/3	48	10
1516～1596	4	1516	11/4	8/1	81	1

(15,4)数据符值 V_D 与奇子集的值及偶子集的值的关系见下式：

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

式中：

T_{ODD}——奇子集值的总数；

V_{EVEN}——偶子集的值；

V_{ODD}——奇子集的值；

G_{SUM}——前面各组所能表示的字符值的总数。

通过下面两式把内侧数据符的值 V_D 编码为 V_{EVEN} 和 V_{ODD}：

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

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

注：与(16,4)外侧数据符相比，这些计算式中偶子集和奇子集的位置是相反的。

5.2.3　符号的值

符号的值由左侧数据符对的值和右侧数据符对的值组合构成。各数据符对的值由相应外侧数据符的值和内侧数据符的值组合构成。数据符对的值的范围列在表 3 中。

表 3　数据符对的值

外侧数据符			内侧数据符			数据符对	
(n,k)	值的总数($V_{OUTSIDE}$)	值的范围	(n,k)	值的总数(V_{INSIDE})	值的范围	值的数目	值的范围
(16,4)	2841	0～2840	(15,4)	1597	0～1596	4537077	0～4537076

数据符对的值 V_{PAIR} 与内、外侧数据符值的关系见下式：

$$V_{PAIR}=(1597\times C_{OUTSIDE})+C_{INSIDE}$$

式中：

C_{INSIDE} 与 $C_{OUTSIDE}$——内、外侧数据符值。

通过下面两式把数据符对的值 V_{PAIR} 编码为 $C_{OUTSIDE}$ 与 C_{INSIDE}：

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

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

例如：如果数据符对的值 V_{PAIR} 为 1971265，那么 $C_{OUTSIDE}$ 与 C_{INSIDE} 是：

$$C_{OUTSIDE}=1971265\text{ div }1597=1234$$

$$C_{INSIDE}=1971265\text{ mod }1597=567$$

符号的值与左侧和右侧数据符对的值的关系见下式：

$$V_{SYMBOL}=(4537077\times V_{LPAIR})+V_{RPAIR}$$

式中：

V_{SYMBOL}——符号的值；

V_{LPAIR} 和 V_{RPAIR}——左侧和右侧数据符对的值。

通过下面两式把符号的值 V_{SYMBOL} 编码为 V_{LPAIR} 与 V_{RPAIR}：

$V_{LPAIR}=V_{SYMBOL}$ div 4537077

$V_{RPAIR}=V_{SYMBOL}$ mod 4537077

例如，如果符号的值 V_{SYMBOL} 是 1234567890，那么左侧数据符对的值 V_{LPAIR} 和右侧数据符对的值 V_{RPAIR} 是：

$V_{LPAIR}=1234567890$ div $4537077=272$

$V_{RPAIR}=1234567890$ mod $4537077=482946$

数据符值组合产生 20585067703929 个值。其中，只有前面的 20000000000000 个值（0～19999999999999）被使用。最高位数字是二维部分连接标志：0 用于独立的 RSS-14，1 用于二维部分与主符号 RSS-14 相结合的情况中。将其余 13 位数字与标志位分离，形成项目标志。隐含的模 10 校验码被计算出来（见附录 A），加到 13 位数字的末端，形成 14 位的全球贸易项目代码。应用标志符 01 被添加到传输数据中，位于必须传输的码制标志符（见 ISO/IEC 15424）]e0 或]C1 之后。

5.2.4 定位符

RSS 条码符号中有 2 个定位符，定位符还可以对符号的校验和的值进行编码。每个定位符可以对 9 个值进行编码。左侧定位符位于数据符 1 和数据符 2 之间，右侧定位符位于数据符 4 和数据符 3 之间。由于定位符与 4 个数据符相邻，符号可以分为 4 个段进行扫描，每个段包括 1 个数据符和 1 个定位符。

5.2.4.1 定位符的结构

每一个定位符都由 5 个单元、共 15 个模块组成。左侧定位符的起始和结束都是空单元，右侧定位符的起始和结束都是条单元。定位符的单元如图 2 所示由符号外侧到内侧的方向进行排序。

定位符的单元 2 和单元 3 的模块之和是 10～12，而单元 4 和单元 5 中的模块之和为 2。宽单元对（单元 2 和单元 3）的宽度与相连的 4 个单元（单元 2～单元 5）宽度和的比值在 10∶12～12∶14 范围内。这个比值用于对定位符的识别。表 4 列出了 9 个编码值的定位符单元宽度。

表 4 定位符的值和单元宽度

定位符的值	单元宽度（模块数）				
	单元 1	单元 2	单元 3	单元 4	单元 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
注：单元从符号外侧向内侧排序。					

左、右侧定位符的配对中“8,0”和“0,8”两种是不使用的。因为在出现一个模块宽的边缘错误的情况下，值为 0 和 8 的定位符中的一个会被错误译码，得出另一个的反转形式的值。其余 79 个可能的组合将对模 79 校验和的值进行编码。

5.2.4.2 校验和的计算

左、右侧定位符的值 C_{LEFT} 和 C_{RIGHT} 每个都有 9 种可能的值。定位符值的配对 0,8 和 8,0 是不可用的，则共有 $9\times9-2$ 即 79 个组合。校验和的值等于数据符单元宽度加权的和模 79 运算的结果，按下式

计算：

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

式中：

$W_{N,M}$——数据符 N 中序号为 M 的单元的权(见表 5)；

$E_{N,M}$——数据符 N 中单元 M 的宽度模块数；

$W_{N,M}E_{N,M}$——两者的乘积。

表 5 中权的值是 3 的连续次幂模 79 运算的结果，用公式 $W_{N,M}=3^{M+8N-9}\bmod 79$ 计算。

表 5 校验和计算的数据符单元的权

数据符序号(N)	数据符单元序号(M)							
	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

采用下面的方法对两个定位符进行编码：

令 temp＝校验和的值。

如果 temp 大于或等于 8，则将 temp＋1 作为本步得出的 temp；否则将 temp 作为本步得出的 temp。

如果上一步得出的 temp 大于或等于 72，则将 temp＋1 作为本步得出的 temp；否则将 temp 作为本步得出的 temp。

上一步得出的 temp 用于下面的运算。

$C_{\text{LEFT}}=\text{temp div } 9$

$C_{\text{RIGHT}}=\text{temp mod } 9$

校验和的计算与校验符选择的完整示例见附录 F.1。

5.2.4.3 定位符的译码

通过将 4 个相连单元的宽度的总和与这 4 个单元左侧单元对或右侧单元对的宽度进行比较来识别定位符。定位符上述宽度的比在 12∶9.5～14∶12.5 范围之内。左侧定位符和右侧定位符可通过它们各自的条/空交替模式来区别。

定位符及对有效数据符与定位符间距比的检查，将确认有效的 RSS-14 符号的四分之一的 1 个段已经被扫描。

5.2.5 参考译码算法

条码识读系统设计成可在现行译码算法允许的范围内识读有缺陷的条码符号。本部分叙述了 GB/T 14258 中描述的用于检测符号质量的可译码度值计算中使用的参考译码算法。

算法包括下列译码步骤：

a) 通过从左到右和从右到左寻找一段 4 个单元的序列并计算其中相应单元宽度的比，找到符号：

1) 从左到右：

9.5∶12 ≤(单元 1＋单元 2)∶(单元 1＋单元 2＋单元 3＋单元 4)≤ 12.5∶14

2) 从右到左：

9.5∶12 ≤(单元 3＋单元 4)∶(单元 1＋单元 2＋单元 3＋单元 4)≤ 12.5∶14

注：上面的单元 1、单元 2、单元 3、单元 4 的序号是一段 4 个单元的序列中的单元序号，顺着扫描方向排序，它们与定位符中单元的序号是不同的。

通过上述比率的确定识别出左侧定位符的第 2 个单元～第 5 个单元。采用同样的方法可

识别出右侧定位符的第 2 个单元～第 5 个单元，但要将上面 1)中的“从左到右”改为“从右到左”；将上面 2)中的“从右到左”改为“从左到右”。

使用步骤 c)中 1)～3)的方法对定位符进行译码，利用定位符前 4 个单元的宽度和(p)，找到标称的相似边之间距离的数值 E_1 和 E_2，因为定位符前 4 个单元的宽度和(p)的模块总数为 14，此时需将步骤 c) 2)中与 p 相除的数由 16 改为 14。验证数值 E_1 和 E_2 是否符合有效的 RSS-14 定位符。

b) 确定定位符的方向和条—空交替模式。利用定位符及方向，确定某种起始单元颜色(条或空)的相邻数据符是哪种(n,k)结构，即是(16,4)还是(15,4)。

c) 具有(16,4)结构的数据符，译码如下：

1) 获得 7 个宽度的测量值 p、e_1、e_2、e_3、e_4、e_5 和 e_6(图 3)。

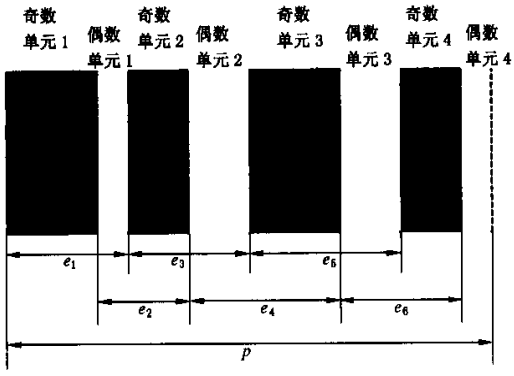


图 3 译码测量

注：上图表示左起条单元为第一个单元，但数据符也可以是上图从左到右镜像或条空转换的形式。

2) 将测量值 e_1、e_2、e_3、e_4、e_5 和 e_6 转换为表示整数模块宽度(E_i)的标称值 E_1、E_2、E_3、E_4、E_5 和 E_6。下面的方法用于 E_i(i=1,2,…,5,6)的确定：

如果 $1.5p/16 \leqslant e_i < 2.5p/16$，那么 $E_i=2$；

如果 $2.5p/16 \leqslant e_i < 3.5p/16$，那么 $E_i=3$；

如果 $3.5p/16 \leqslant e_i < 4.5p/16$，那么 $E_i=4$；

如果 $4.5p/16 \leqslant e_i < 5.5p/16$，那么 $E_i=5$；

如果 $5.5p/16 \leqslant e_i < 6.5p/16$，那么 $E_i=6$；

如果 $6.5p/16 \leqslant e_i < 7.5p/16$，那么 $E_i=7$；

如果 $7.5p/16 \leqslant e_i < 8.5p/16$，那么 $E_i=8$；

如果 $8.5p/16 \leqslant e_i < 9.5p/16$，那么 $E_i=9$。

否则字符出错。

3) 从 E 值确定字符各单元的标称宽度。n(n=16)个模块中剩余的模块分配给最后 1 个单元，得出该单元的宽度，而不是从 E 值中计算出来。有效的单元宽度集合是没有单元宽度小于 1 个模块、并且至少有 1 个偶数单元是 1 个模块宽。例如：图 3 中 E_1～E_6 的值是{4 3 4 5 5 4}，可能的单元宽度的集合是{4 0 3 1 4 1 3 0}(注意不应有 0 宽度单元)、{3 1 2 2 3 2 2 1}或{2 2 1 3 2 3 1 2}(注意没有单个模块的偶数序号单元)，其中只有 8 个单元宽度为{3 1 2 2 3 2 2 1}的集合满足要求，因此被选作字符各单元的宽度。如果导出的单元宽度的集合都是无效的，那么字符出错。附录 G 给出了这种单元宽度译码算法的 C 语言程序。

4) 采用附录 B 中的程序确定奇子集和偶子集的值。

5) 从奇子集和偶子集的值计算数据符的值。

6) 计算并存储单元宽度加权和，用于校验和的计算。

d) 具有(15,4)结构的数据符，译码如下：

1) 获得 7 个宽度的测量值 p、e_1、e_2、e_3、e_4、e_5 和 e_6(图 3)。

2) 将测量值 e_1，e_2，e_3，e_4，e_5 和 e_6 转换为表示整数模块宽度(E_i)的标称值 E_1、E_2、E_3、E_4、E_5 和 E_6。下面的方法用于 E_i(i=1,2,…,5,6)的确定：

如果 $1.5p/15 \leqslant e_i < 2.5p/15$，那么 $E_i=2$；

如果 $2.5p/15 \leqslant e_i < 3.5p/15$，那么 $E_i=3$；

如果 $3.5p/15 \leqslant e_i < 4.5p/15$，那么 $E_i=4$；

如果 $4.5p/15 \leqslant e_i < 5.5p/15$，那么 $E_i=5$；

如果 $5.5p/15 \leqslant e_i < 6.5p/15$，那么 $E_i=6$；

如果 $6.5p/15 \leqslant e_i < 7.5p/15$，那么 $E_i=7$；

如果 $7.5p/15 \leqslant e_i < 8.5p/15$，那么 $E_i=8$；

如果 $8.5p/15 \leqslant e_i < 9.5p/15$，那么 $E_i=9$。

否则，字符出错。

3) 使用上面的步骤 c)中 3)～6)的方法计算(15,4)数据符的值。

e) 使用上面的确定标称单元宽度的方法对定位符进行译码，在表 4 中查找图形，得出定位符的值。

f) 当 4 个数据符和 2 个定位符全部被译码之后，验证 2 个定位符的值是否属于 79 个有效对的集合。验证从 2 个定位符中计算出来的模 79 的校验和的值与数据符单元宽度加权和的模 79 运算的结果是否一致。

g) 从 4 个数据符计算连接标志和项目标识代码。

h) 此外，为稳妥起见，考虑到具体的识读设备和设想中的应用环境，需对扫描加速度、绝对计时以及尺寸等进行其他的附加检查。

在设计实际的识读 RSS-14 的扫描器时，可参见附录 H 中为使误读最小化而附加的符号译码考虑。

5.3 特定应用中的 RSS-14 形式

5.3.1 截短式 RSS-14

截短式 RSS-14(见图 4)的结构和编码与标准 RSS-14 一样，唯一不同的是它的高度可减小到最小值 13X。当需要包装指示符数字大于 1 时，小项目可以用截短式 RSS-14，而不用限定式 RSS。当需要 4 列二维部分且要求复合码的高度最小时，也可以使用截短式 RSS-14。

截短式 RSS-14 可以采用光笔、手持激光扫描器、线性和二维图像式扫描器识读。它不能被全向式扫描器有效识读。

截短式 RSS-14 的整体尺寸是 96X 宽、13X 高(最小)。

图 4 表示数据(01)00012345678905 的截短式 RSS-14 符号

5.3.2 RSS-14 的两行形式

RSS-14 的两行形式是从标准 RSS-14 符号的中央即数据符 2 与数据符 4 的接合处把符号分成两半，再把左、右两半部分排成上、下两行的 RSS-14 符号。上面行由符号左半部分和右侧附加 1X 条和 1X 空组成的保护符构成；下面行由符号右半部分和左侧附加 1X 条和 1X 空组成的保护符构成。

两行形式的 RSS-14 条码包括层排式 RSS-14 和全向层排式 RSS-14。

5.3.2.1 层排式 RSS-14

层排式 RSS-14(图 5)是截短式 RSS-14 的两行形式。图 5 中的条码符号对图 4 中符号表示的数据进行编码,以便于比较。

图 5 表示数据(01)00012345678905 的层排式 RSS-14 符号

上面行是 5X 高,下面行是 7X 高,在两行之间是 1X(最小)高的层分隔符。层排式 RSS-14 符号的整体尺寸是 50X 宽、13X 高。

层分隔符的前 4 个和最后 4 个模块总是空。若位于层分隔符的上面行和下面行的垂直相对的模块具有相同的颜色,则此处层分隔符的模块颜色与上、下行中的模块颜色相反,即在上面行、下面行两个垂直相对的条的部分之间形成一个层分隔符的空,两个垂直相对的空的部分之间形成一个层分隔符的条。

若位于层分隔符的上面行和下面行的垂直相对的模块颜色不同,则此处层分隔符的每一个模块的颜色与其左侧的模块颜色相反。这样,层分隔符可在上面行和下面行模块颜色互补的一段区域出现单模块宽的条和空交替出现的图形(见图 5)。

在可用空间对限定式 RSS 太窄的情况下,对小项目可以使用层排式 RSS-14,而不使用限定式 RSS。而且,层排式 RSS 比较窄允许使用较大的 X 尺寸,印刷质量容易保证。但是,在空间允许而不必减少 X 尺寸的情况下,限定式 RSS 或截短式 RSS-14 应当优先于层排式 RSS 符号使用,因为它们容易被光笔或线性扫描器扫描。

层排式 RSS-14 可以采用光笔、手持激光扫描器、线性和二维图像式扫描器识读。它不能被全向式扫描器有效识读。

5.3.2.2 全向层排式 RSS-14

全向层排式 RSS-14(图 6)是完整高度的 RSS-14 符号的两行排列形式。一个由 3 个 1X(最小)高的行组成 3X(最小)高的层分隔符用来分隔符号的两行。

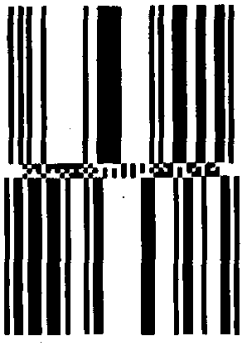


图 6 表示数据(01)00034567890125 的全向层排式 RSS-14 符号

层分隔符各行的前 4 个和最后 4 个模块总是空。

除了定位符第 1、2、3 单元之下的 13 个模块之外,层分隔符第 1 行的单元是由符号上面行条/空颜色的互补色形成的。上述 13 个模块,在相邻定位符的条之下的是浅色,在相邻定位符的空之下的是深浅交替的形式。

除了层分隔符行的两端的各 4 个模块之外,层分隔符的第 2 行由交替的条空模块组成。

除了定位符第 1、2、3 单元(从右到左)之上的 13 个模块之外,层分隔符第 3 行的单元是符号下面行条/空颜色的互补色形成的。上述 13 个模块,在相邻的定位符的条之上的是浅色,在相邻定位符的空之上的是以深色开始的深浅交替的形式。出现在数值为 3 的定位符上面的 13 个模块中的单个深色模块

被转换成右侧的一个模块，以便使得这个深色模块能够位于3个模块宽的定位符条的起始部分的上方。

符号的每行最小高度为33X，在两行之间有一个3X高的层分隔符。这种形式的RSS符号的整体尺寸是50X宽、69X高(最小)。

对全向扫描的应用场合，在需要不同的符号宽、符号高比时可使用全向层排式RSS-14来替代标准RSS-14符号。

6 限定式RSS条码符号的要求

6.1 限定式RSS的基本特点

限定式RSS是线性符号，能够对4000000000000(4×10^{12})个数值编码(见6.2.3)。包装指示符为0和1的贸易项目代码可以被编码，此外，还提供了连接标志。

限定式RSS可以采用光笔、手持激光扫描器、一维和二维图像式扫描器识读。它不能被全向通道式扫描器有效识读。图7是一个限定式RSS符号的示例。

图7 表示数据(01)15012345678907的限定式RSS符号

6.2 限定式RSS的符号结构

限定式RSS符号是由左侧保护符、左侧数据符、定位符(又是校验符)、右侧数据符和右侧保护符共5个区域(从左到右)，74个模块组成，结构如图8所示。5个区域的条、空及模块组成如下：

a) 左侧保护符：由1个模块的空和1个模块的条组成；

b) 左侧数据符：由7个空和7个条共26个模块组成，$(n,k)=(26,7)$；

c) 定位符：由7个空和7个条共18个模块组成，$(n,k)=(18,7)$；

d) 右侧数据符：由7个空和7个条共26个模块组成，$(n,k)=(26,7)$；

e) 右侧保护符：由1个模块的空和1个模块的条组成。

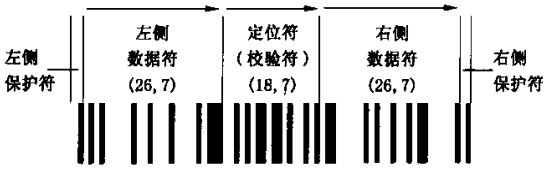


图8 限定式RSS符号结构示意图

限定式RSS符号包括46个单元，见附录E.2。符号最小高度为10X。

限定式RSS不需要空白区。如果相邻的背景区域颜色(左边是浅色或右边是深色)与保护符外侧单元的颜色相同，那么第一个或最后一个单元会出现宽于1个模块的情况，但不会影响符号的识读。

6.2.1 数据符结构

每个数据符的模块组配为(n,k)结构。n的值是26，k的值是7。

两个数据符的单元从左到右排序，如图8所示。

每个数据符的单元集合包含奇和偶两个子集，这里的奇和偶指的是每个子集中单元序号的奇、偶。奇数单元是空，偶数单元是条。例如：每个数据符奇子集由第1、第3、第5、第7、第9、第11和第13单元组成，排序从最左侧的单元开始。这种7个单元的子集的模块数目从最小7个到最大19个。在数据符中，奇子集和偶子集中模块数目的总和等于26。

6.2.2 数据符的值

对每个数据符值，由一个算法给出奇子集和偶子集中各单元宽度的值(以模块为单位)组成的序列。

需要给该算法提供单元数量、模块数量、最大单元宽度以及子集中所有的单元是否都比1个模块宽等信息。附录B给出了用C语言程序实现的生成限定式RSS条码符号数据符单元的算法。

有效的偶子集至少有一个单模块单元，而奇子集可以所有单元都大于1个模块。对偶数单元的上述限制保证了数据符的各个边缘到相似边缘距离(条加空与空加条)有唯一的模块和。

表6描述了(26,7)子集的特征，列出了7组奇和偶子集对。2个子集的模块数都为奇数。最宽的单元的宽度被规定下来，以使在一对相邻单元中的模块数之和不会超过9。在有限制要求的情况下，构成一个(26,7)数据符的条空组合的模式共有2013571种，能表示2013571个值。

表6 数据符(26,7)的特征

数据符值的范围	组	前面各组所能表示的数据符值的总数(G_{SUM})	奇/偶子集模块数	奇数/偶数最宽单元	奇子集值的总数(T_{ODD})	偶子集值的总数(T_{EVEN})
0～183063	1	0	17/9	6/3	6538	28
183064～820063	2	183064	13/13	5/4	875	728
820064～1000775	3	820064	9/17	3/6	28	6454
1000776～1491020	4	1000776	15/11	5/4	2415	203
1491021～1979844	5	1491021	11/15	4/5	203	2408
1979845～1996938	6	1979845	19/7	8/1	17094	1
1996939～2013570	7	1996939	7/19	1/8	1	16632

数据符值V_D与奇子集值、偶子集值的关系见下式：

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

式中：

T_{EVEN}——偶子集值的总数；

V_{ODD}——奇子集的值；

V_{EVEN}——偶子集的值；

G_{SUM}——前面各组所能表示数据符值的总数。

通过以下两式把数据符值V_D编码为V_{ODD}与V_{EVEN}：

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

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

例如：对数据符值917879进行编码，从表6可知，数据符的值在第3组范围内，因此$G_{SUM}=820064$，$T_{EVEN}=6454$，采用上面的公式：

$V_{ODD}=(917879-820064)\text{div}\ 6454=97815\ \text{div}\ 6454=15$

$V_{EVEN}=(917879-820064)\text{mod}\ 6454=97815\ \text{mod}\ 6454=1005$

使用附录B中的算法，第3组中的数据符(见表6)的奇子集有9个模块，子集值15是28个连续值(0～27)中的一个；偶子集有17个模块，子集值1005是6454个连续值(0～6453)中的一个。各奇数单元的宽度是{1 2 1 1 1 1 2}，各偶数单元的宽度是{1 2 3 5 1 2 3}，从左到右给出数据符各单元的宽度为{1 1 2 2 1 3 1 5 1 1 1 2 2 3}。

6.2.3 符号的值

符号的值与左、右侧数据符值的关系见下式：

$$V_{SYMBOL}=(2013571\times V_{DLEFT})+V_{DRIGHT}$$

式中：

V_{SYMBOL}——符号的值；

V_{DLEFT}和V_{DRIGHT}——左侧和右侧数据符值。

通过以下两式将符号的值 V_{SYMBOL} 编码为 V_{DLEFT} 和 V_{DRIGHT}：

$V_{DLEFT}=V_{SYMBOL}$ div 2013571

$V_{DRIGHT}=V_{SYMBOL}$ mod 2013571

左、右侧数据符的值结合在一起可以产生 4054468172041 个值，但只有 4000000000000 个值被使用，分成值为 0～1999999999999 和值为 2015133531096～4015133531095 的两组。这样分组是为了能由左侧数据符奇、偶子集的模块数目（反映左侧数据符值的范围）来确定是否存在二维部分，而不需要对右侧数据符译码的结果。独立的限定式 RSS-14 左侧数据符值的范围为 0～993260，而在 EAN·UCC 复合码中，限定式 RSS-14 左侧数据符值的范围为 1000776～1994036。

符号的值的第二组（值的范围为 2015133531096～4015133531095），表明连接标志为“1”，即表示伴随限定式 RSS 有 1 个二维部分。原数据值则是从限定式 RSS 符号的值中减去 2015133531096 的结果，在 0～1999999999999 之间，与第一个数值组一致，构成了贸易项目代码的前 13 位。

数值 0～1999999999999 表示 14 位的全球贸易项目代码的前 13 位数字，在这里包装指示符只有 0 和 1 两个可能的数值。隐含的模 10 校验码被计算出来，并加到传输数据的末端形成 14 位的全球贸易项目代码。应用标识符 01 被添加到传输数据中，跟在必须传输的码制标识符]e0 或]C1 之后。

6.2.4 校验符

限定式 RSS 符号有一个校验符，它也是符号的定位符，位于左、右侧数据符之间。

6.2.4.1 校验符的结构

校验符的模块组配结构为(18,7)，对 89(0～88)个数值进行编码。每个校验符包含 9 个模块组成的 7 个空和 9 个模块组成的 7 个条。在选择校验符图形时，去掉了这样的一些图形，这些 14 个单元组成的图形如果条空颜色互换和/或从左到右镜像反转会与其他图形相同。附录 C 列出了 89 个编码数值的校验符单元宽度。

6.2.4.2 校验符值的计算

校验符的值等于数据符单元宽度加权的和模 89 运算的结果。计算公式如下：

$(W_{1,1}E_{1,1}+W_{1,2}E_{1,2}+\cdots+W_{2,1}E_{2,1}+\cdots+W_{2,14}E_{2,14})$mod 89

式中：

$W_{N,M}$——数据符 N 中序号为 M 的单元的权（见表 7）；

$E_{N,M}$——数据符 N 中单元 M 的宽度模块数；

$W_{N,M}E_{N,M}$——两者的乘积。

表 7 中权的值是 3 的连续次幂模 89 运算的结果，按公式 $W_{N,M}=3^{M+14N\ -\ 15}$ mod 89 计算。

表 7 校验符值计算的单元的权

数据符序号(N)	数据符单元序号(M)													
	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
注：单元由左向右排序。														

附录 F.2 给出了一个限定式 RSS 符号的编码示例。

6.2.5 定位符

定位符是通过 14 个单元的数据符宽度与 14 个单元的校验符（即定位符）宽度之比 26∶18∶26 被识别的。此外，还可用校验符特有的有效空/条图形对条码符号进行识别。可能出现在条码符号中的镜像反转和条空置换的校验符图形已经被删除。

6.2.6 参考译码算法

条码识读系统设计成可在现行算法允许的范围内识读有缺陷的条码符号。本部分叙述了

GB/T 14258 中描述的用于检测符号质量的可译码度值计算中使用的参考译码算法。

算法包括下列译码步骤：

a) 通过查找宽度比为(26±1.5)：18：(26±1.5)的 3 个 14 个单元的序列找到符号(±1.5 的允许偏差是考虑到可能出现的扫描加速度的影响)。

b) 检查中间的序列是否为一个有效的定位符。利用下面步骤 c)中的 1)和 2)来确定定位符的单元宽度，这时需把步骤 c)的 2)中与 p 相除的数由 26 改为 18。单元宽度的查找见附录 C 的表。

c) 数据符译码如下：

1) 获得 13 个宽度的测量值：p、e_1、e_2、e_3、e_4、e_5、e_6、e_7、e_8、e_9、e_{10}、e_{11}及 e_{12}(见图 9)。

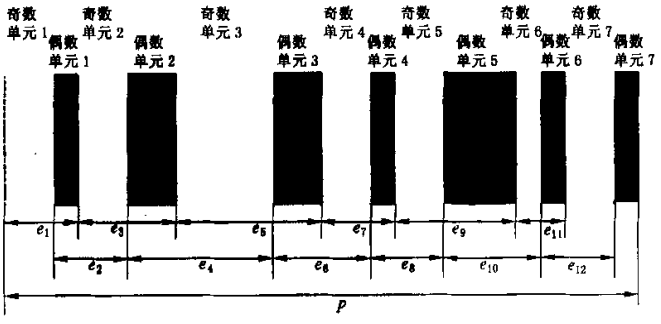


图 9 译码测量

2) 将测量值 e_1、e_2、e_3、e_4、e_5、e_6、e_7、e_8、e_9、e_{10}、e_{11}及 e_{12}转换为表示整数模块宽度(E_i)的标称值 E_1、E_2、E_3、E_4、E_5、E_6、E_7、E_8、E_9、E_{10}、E_{11}及 E_{12}。下面的方法用于 E_i($i=1,2,\cdots,11,12$)的确定：

如果 $1.5p/26 \leqslant e_i < 2.5p/26$，则 $E_i=2$；

如果 $2.5p/26 \leqslant e_i < 3.5p/26$，则 $E_i=3$；

如果 $3.5p/26 \leqslant e_i < 4.5p/26$，则 $E_i=4$；

如果 $4.5p/26 \leqslant e_i < 5.5p/26$，则 $E_i=5$；

如果 $5.5p/26 \leqslant e_i < 6.5p/26$，则 $E_i=6$；

如果 $6.5p/26 \leqslant e_i < 7.5p/26$，则 $E_i=7$；

如果 $7.5p/26 \leqslant e_i < 8.5p/26$，则 $E_i=8$；

如果 $8.5p/26 \leqslant e_i < 9.5p/26$，则 $E_i=9$。

否则字符出错。

3) 从 E 值确定字符各单元的标称宽度。$n(n=26)$个模块中剩余的模块分配给最后一个单元，得出该单元的宽度，而不是从 E 值中计算出来。有效的单元宽度集合是没有单元宽度小于 1 个模块、并且至少有一个偶数单元是 1 个模块宽。例如，图 9 中值 $E_1 \sim E_{12}$的值为{3 3 4 6 6 4 3 3 5 4 2 3}，可能的单元宽度的集合是{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}或{3 0 3 1 5 1 3 0 3 2 2 0 3 0}，其中只有 14 个单元宽度为{2 1 2 2 4 2 2 1 2 3 1 1 2 1}的集合满足要求，因此被选作字符各单元的宽度。如果导出的单元宽度的集合都是无效的，那么字符出错。附录 G 给出了这种单元宽度译码算法的 C 语言程序。

4) 采用附录 B 中的程序确定奇子集和偶子集的值。

d) 当两个数据符和定位符全部被译码之后，验证从定位符得到的模 89 的校验符值与数据符的单元宽度加权和模 89 运算的结果是否一致。

e) 从两个数据符计算连接标志和贸易项目标识代码。

f) 考虑到具体的识读设备和设想中的应用环境，需对扫描加速度、绝对计时以及尺寸等进行其他的附加检查。

7 扩展式 RSS 条码符号的要求

7.1 扩展式 RSS 的基本特点

扩展式 RSS 是一种可变长度的线性条码符号，它能将应用标识符(AI)单元数据串的多达 74 个的数字或 41 个的字母符号编码，内在地表现为 1 串二进制数字。在扫描器及应用软件附加了该码制的相应程序的前提下，扩展式 RSS 可用来对零售点及其他应用中使用的主要数据和补充数据进行编码。

可以对多达 22 个段的扩展式 RSS 进行扫描及译码，然后重组，因此可以采用全向式扫描器进行扫描。图 10 为一个扩展式 RSS 条码符号的示例。

图 10 表示数据(01)98898765432106(3202)012345(15)991231 的扩展式 RSS 符号

7.2 扩展式 RSS 的符号结构

7.2.1 整体符号结构

符号中的第 1 个符号字符是校验符，用来对校验和及符号长度(符号字符总数)进行编码，图 11 所给出的条码符号包含 1 个校验符和 5 个数据符。符号字符包括校验符和数据符。

扩展式 RSS 符号被构建成为 1 种三元序列，每一序列都包含 2 个符号字符以及它们中间的定位符。如果存在奇数个符号字符，则最后 1 个符号字符后面接有 1 个定位符。编号为奇数的符号字符的单元排序从左至右，编号为偶数的符号字符的单元排序则从右至左。编号为 1、2、5、6、9、10 等的符号字符，第 1 个单元(即距离相邻定位符最远的那个单元)为空；编号为 3、4、7、8、11、12 等的符号字符，第 1 个单元为条。左右两边的保护符总是位于条码的起始及终止位置，或者位于层排式符号每一行的起始及终止位置。图 11 所示的有 6 个符号字符的扩展式 RSS 符号由 11 个区域组成(从左至右)：

a) 左侧保护符：由 1 个模块空及 1 个模块条组成；

b) 校验符：由 4 个空及 4 个条共 17 模块组成，$(n,k)=(17,4)$；

c) 定位符 A1：由 3 个空及 2 个条共 15 个模块组成；

d) 数据符 1：由 4 个条及 4 个空共 17 个模块组成，$(n,k)=(17,4)$；

e) 数据符 2：由 4 个条及 4 个空共 17 个模块组成，$(n,k)=(17,4)$；

f) 定位符 B2：由 3 个条及 2 个空共 15 个模块组成；

g) 数据符 3：由 4 个空及 4 个条共 17 个模块组成，$(n,k)=(17,4)$；

h) 数据符 4：由 4 个空及 4 个条共 17 个模块组成，$(n,k)=(17,4)$；

i) 定位符 B1：由 3 个空及 2 个条共 15 个模块组成；

j) 数据符 5：由 4 个条及 4 个空共 17 个模块组成，$(n,k)=(17,4)$；

k) 右侧保护符：由 1 个模块的条及 1 个模块的空组成。

图 11 所示的条码符号有 67 个单元，共 151 个模块。附录 E.3 列出了图 11 中所示扩展式 RSS 符号的全部 67 个单元。符号的最小高度为 $34X$。

注：图中箭头表示每个字符单元排序的方向。

图 11　扩展式 RSS 符号结构示意图

扩展式 RSS 不需要空白区。如果保护符外侧单元颜色与相邻的背景区域的颜色相同，那么第 1 个或最后 1 个单元会出现宽于 1 个模块的情况。

注：扩展式 RSS 条码符号右侧保护符的形式不是固定的，根据位于最后的符号字符或定位符的条空组合模式，右侧保护符可能是 1 个模块的条加 1 个模块的空的形式，也可能是 1 个模块的空加 1 个模块的条的形式。

7.2.2　符号字符结构

每个符号字符的模块组配为(n,k)结构。n 值为 17，k 值为 4。

编号为奇数的符号字符(校验符及偶数数据符)的单元从左至右排序，编号为偶数的符号字符(奇数数据符)的单元从右至左排序。因此符号字符单元朝着相邻的定位符排序，如图 11 中的箭头所示。

每 1 个符号字符单元的集合均包含 1 个奇子集和 1 个偶子集。这里的奇和偶指的是每个子集中单元序号的奇、偶。每 1 个符号字符的奇子集由第 1、第 3、第 5 及第 7 单元组成，且以离定位符最远的单元开始排序。偶子集同样是从离相邻定位符最远的单元开始，第 8 单元(序号为偶数，位于内侧)与定位符相邻。

7.2.3　符号字符值

对每个符号字符值，由 1 个算法给出奇子集和偶子集中各单元宽度的值(以模块为单位)组成的序列，需要为这个算法提供单元数量、模块数量、单元最大宽度以及子集中所有的单元是否都比 1 个模块宽等信息。附录 B 给出了用 C 语言程序实现的生成扩展式 RSS 符号字符单元的算法。

有效的奇子集至少有 1 个单模块单元，而有效的偶子集则无须有 1 个单模块单元的限制。对奇数单元的上述限制保证了符号字符的各个边缘到相似边缘(条加空与空加条)有唯一的模块和。

除上述限制外，第 1 个(离定位符最远的)奇数单元的宽度总是小于 5 个模块宽度。这种限制可避免在相邻的符号字符之间出现错误的定位符。

表 8 描述了(17,4)子集的特征，列出了 5 组奇、偶子集对。奇子集含有偶数个模块，偶子集则含有奇数个模块。最宽的单元的宽度被规定下来，保证在一对相邻单元中的模块数之和不会超过 9。在有限制要求的情况下，构成一个(17,4)符号字符的条空组合的模式共有 4192 种，能表示 4192 个值。

表 8　符号字符(17,4)的特征

符号字符值的范围	组	前面各组所能表示的符号字符值总数(G_{SUM})	奇/偶子集模块数	奇数/偶数最宽单元	奇子集值的总数(T_{ODD})	偶子集值的总数(T_{EVEN})
0～347	1	0	12/5	7/2	87	4
348～1387	2	348	10/7	5/4	52	20
1388～2947	3	1388	8/9	4/5	30	52
2948～3987	4	2948	6/11	3/6	10	104
3988～4191	5	3988	4/13	1/8	1	204

符号字符值 V_S 与奇子集值和偶子集值的关系见下式：

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

式中：

T_{EVEN}——偶子集值的总数；

V_{ODD}——奇子集的值；

V_{EVEN}——偶子集的值；

G_{SUM}——前面各组所能表示的符号字符值的总数。

通过下面的两式把符号字符值 V_S 编码为 V_{ODD} 和 V_{EVEN}：

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

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

例如：对一个值为 3544 的(17,4)符号字符进行编码，从表 8 可知，符号字符的值位于第 4 组内，因此 $G_{SUM}=2948$，而 $T_{EVEN}=104$，应用上面的公式：

$V_{ODD}=(3544-2948)$ div $104=596$ div $104=5$

$V_{EVEN}=(3544-2948)$ mod $104=596$ mod $104=76$

使用附录 B 中的算法，第 4 组中的数据符(见表 8)的奇子集有 6 个模块，值 5 是 10 个连续值(0～9)中的一个；偶子集单元有 11 个模块，值 76 是 104 个连续值(0～103)中的一个。各奇数单元宽度是{1 3 1 1}，各偶数单元宽度是{4 1 4 2}，给出数据符各单元宽度为{1 4 3 1 1 4 1 2}。

7.2.4 符号的二进制值

数据符值的范围从 0～4095，每 1 个值都代表 1 个对符号值编码的 12 位二进制数。代表各个数据符值的二进制数链接成为编码的二进制数字串。第 1 个数据符(即第 2 个符号字符)包含最高阶二进制位。

条码符号大小及相应二进制数字串的长度列于表 9 中。

表 9 条码符号大小对应的二进制容量

符号字符数	数据符数	编码的二进制位数
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

表 9(续)

符号字符数	数据符数	编码的二进制位数
18	17	204
19	18	216
20	19	228
21	20	240
22	21	252

7.2.5 数据编码

被编码成 RSS 符号的用户数据，总是由遵循 GS1 通用规范数据标准的应用标识符及数据字段组成，并被准确地格式化，与被编码成 UCC/EAN-128 条码符号时相同。对 RSS 符号进行编码时，应遵循 GB/T 15425 标准中 AI 单元数据串的链接规则——如：利用 FNC1 把可变长度单元数据串与其后的单元数据串分隔开。

扩展式 RSS 的二进制数字串可被分成多达 5 个的二进制字段，所有扩展式 RSS 条码符号均要求具有前面的两个字段，以及其他 3 个字段中的 1 个或多个字段。这些字段分别是：

a) 二维部分连接标志字段(见 7.2.5.1)；

b) 编码方法字段(见 7.2.5.2)；

c) 可变长度符号位字段(见 7.2.5.3)；

d) 压缩数据字段(见 7.2.5.4)；

e) 通用数据压缩字段(见 7.2.5.5)。

这些二进制字段按顺序链接起来，并被编码成条码符号的二进制数字串，编码方法字段总是编码的第 1 个字段。固定长度的编码方法以压缩数据准确地充满条码符号特定的二进制数字串。可变长度的编码方法以可选的通用数据压缩字段跟接二进制填充位作为结束，填充位填满适当长度符号的二进制数字串中未被使用的二进制位。

在本章中，用加双引号的相应二进制数来表示各个二进制字段。

7.2.5.1 二维部分连接标志字段

这是 1 个用来指示扩展式 RSS 符号是否作为 EAN·UCC 复合码的一部分打印出来的标志位，标志字段值为“0”时表示扩展式 RSS 是单独的条码符号，为“1”时则表示扩展式 RSS 是 EAN·UCC 复合码的一维部分。

7.2.5.2 编码方法字段

编码方法字段由 1 个或多个二进制位组成，位于二维部分连接标志字段之后。它定义条码符号是 1 种通用符号还是 1 种以面向应用的压缩数据字段(比如用来有效代表 1 个项目标识的 AI 单元数据串的字段)开始的符号。编码方法字段定义于表 10 中。

编码方法字段“1”用于对 AI(01)主标识与附加信息的 AI 单元数据串进行编码。

编码方法字段“00”用于对不使用 AI(01)主标识的项目进行编码，它对 4 个或多个符号字符的可变长度符号进行定义。此时符号包含通用数据压缩字段，不包含压缩数据字段。

编码方法字段“0100”及“0101”用于对可变质量项目的主标识及质量进行编码。而编码方法字段“0111000”～“0111111”则用于对主标识、质量及 4 个 AI 单元数据串即 AI(11)、AI(13)、AI(15)、AI(17)中的任何一个进行编码。

编码方法字段“01100”及“01101”用于对主标识及价格进行编码。

注：主标识指的是 14 位的全球贸易项目代码(GTIN)。

表 10 编码方法字段及其特点

编码方法字段	符号字符的数目（个）	压缩数据字段长度（二进制位）	是否跟接通用数据压缩字段	AI 单元数据串
1	5～22	44	是	(01)+其他 AI 数据串
00	4～22	无	是	任何 AI 数据串
0100	固定长度 6	55	否	(01)与(3103)
0101	固定长度 6	55	否	(01)+(3202)或(01)+(3203)
01100	6～22	42	是	(01)+(392*x*)
01101	7～22	52	是	(01)+(393*x*)
0111000	固定长度 8	76	否	(01)+(310*x*)+(11)
0111001	固定长度 8	76	否	(01)+(320*x*)+(11)
0111010	固定长度 8	76	否	(01)+(310*x*)+(13)
0111011	固定长度 8	76	否	(01)+(320*x*)+(13)
0111100	固定长度 8	76	否	(01)+(310*x*)+(15)
0111101	固定长度 8	76	否	(01)+(320*x*)+(15)
0111110	固定长度 8	76	否	(01)+(310*x*)+(17)
0111111	固定长度 8	76	否	(01)+(320*x*)+(17)

7.2.5.3 可变长度符号位字段

这种字段只出现在可变长度符号中，并出现在编码方法字段“1”、“00”、“01100”及“01101”之后。此字段包含两个二进制位，第 1 位指示条码符号中含有的符号字符个数的奇偶，符号字符的个数为偶数时为“0”，为奇数时为“1”。第 2 位指示条码符号的大小，条码符号中的符号字符数小于或等于 14 时为“0”，大于 14 时为“1”。这两个位连同定位符集，为在校验符中所定义的符号字符数提供了一种双重校验（参见 7.2.6）。

7.2.5.4 压缩数据字段

可根据具体的编码方法字段来对这一字段中的二进制数据进行解释。除编码方法字段“00”外的所有符号均包含有一个压缩数据字段。

7.2.5.4.1 编码方法字段“1”——通用项目标识数据

如果 AI(01)单元数据串出现在被编码数据信息的起始位置，则可采用此编码方法字段。从项目标识 AI(01)单元数据串中去掉前两个十进制数字 01 和末尾的校验码，余下的 13 个十进制数字分成 5 个组，编码成一个 44 个二进制位的压缩数据字段。这 5 个组分别由 1、3、3、3、3 个十进制数字组成，并被编码为 4、10、10、10、10 个二进制位。采用直接把十进制转换为二进制的方法对 5 组数据进行编码，其他的附加 AI 数据作为通用数据压缩字段被编码在 44 位压缩数据字段之后。

译码器通过将这 44 个二进制位分为 4、10、10、10、10 个二进制位的 5 个组，并将其分别转换，形成 13 个十进制数字，对压缩数据字段的数据进行重构，并在重构的十进制数字前面增加两位数字 01 作为 AI 前缀，同时在这个 AI 单元数据串末尾增加计算出的模 10 校验码，然后对二进制数字串中剩余的通用数据压缩字段部分进行译码。

举例说明编码方法字段“1”：对(01)00012345678905(10)ABC123 进行编码，实际上只需要将粗体数字部分编码成压缩数据字段，而将 AI(10)及批号 ABC123 编码成通用数据压缩字段，并直接附在 44 个二进制位的压缩数据字段后面。译码器将传输]e0010001234567890510ABC123。

编码方法字段“1”对含有通用数据压缩字段的 5 个或多个符号字符的可变长度符号进行了定义。

7.2.5.4.2 编码方法字段“0100”——可变质量项目(增量为 0.001 kg)

当被编码的数据信息只含有两个 AI 单元数据串——AI(01)后接 AI(3103)时,可采用此编码方法字段。AI(01)项目标识单元数据串必须具有包装指示符数字 9。AI(3103)可变质量单元数据串指定的质量不能超过 32.767 kg。这两个 AI 单元数据串分别被压缩成 40 位二进制数和 15 位二进制数,总的压缩数据字段长度则为 55 个二进制位。编码方法字段“0100”用于不含有通用数据压缩字段的、具有 6 个符号字符的固定长度条码符号的编码。

在对这两个 AI 单元数据串进行编码时,先从 AI(01)单元数据串中去掉前 3 个数字 019 及末尾的校验码,其余的 12 个数字在 40 个二进制位中编码成压缩数据,分为 4 组,每组 3 个数字被编码为 10 位二进制数;再从 AI(3103)单元数据串中去掉前 4 个数字 3103,然后将余下的介于 000000～032767 之间的 6 位数字编码成 15 位二进制数的压缩数据,这样形成总长度为 55 个二进制位的压缩数据字段。

译码时,译码器将前 40 位二进制数分 4 个组(每组 10 位二进制数)译码成 12 个十进制数字(每组三个数),在这 12 个数字前面增加前缀 019;后面增加计算出的模 10 校验码,成为第一个 AI 单元数据串。再将余下的 15 位二进制数转换为十进制数,并在得出的数字前填充 0 以形成 6 位数字,最后将 AI 前缀“3103”添加到译码得出的 6 位数字前,形成数据信息的第二个 AI 单元数据串。

举例说明编码方法字段“0100”,对(01)90012345678908(3103)001750 进行编码,实际上只将粗体数字编码成压缩数据字段。译码器将传输]e0019001234567890831030017 50。

7.2.5.4.3 编码方法字段“0101”——可变质量项目(增量为 0.004536 kg(0.01 lb)或 0.0004536 kg (0.001 lb))

当被编码的数据信息只含有两个 AI 单元数据串——AI(01)后跟 AI(3202)或 AI(3203)时,可使用此编码方法字段。AI(01)项目标识单元数据串必须具有包装指示符数字 9,AI(3202)可变质量单元数据串指定的质量不能超过 45.36 kg(99.99 lb),而 AI(3203)可变质量单元数据串指定的质量不能超过 10.327 kg(22.767 lb)。两个 AI 单元数据串分别被压缩成 40 位二进制数和 15 位二进制数,总的压缩数据字段长度则为 55 个二进制位。编码方法字段“0101”用于不含有通用数据压缩字段的、具有 6 个符号字符的固定长度条码符号的编码。

在对两个 AI 单元数据串进行编码时,先从 AI(01)单元数据串中去掉前 3 个数字 019 及末尾的校验码。其余的 12 个数字在 40 个二进制位的压缩数据字段中进行编码,分为 4 组,每组 3 个数字被编码为 10 个二进制位。从质量单元数据串中去掉前 4 位 AI 数字。对于 AI(3202),将余下的介于 0～009999 之间的 6 个数字编码成 15 位二进制数,添加到压缩数据字段中;对于 AI(3203),将余下的介于 0～022767 之间的 6 个数加上 10000 后编码成 15 位二进制数,添加到压缩数据字段中。

译码时,译码器将前 40 位二进制数分 4 个组(每组 10 位二进制数)译码成 12 个十进制数(每组 3 个数),在这 12 个数字前面增加前缀 019;后面增加计算出的模 10 校验码,成为第一个 AI 单元数据串。再将余下的 15 位二进制数转换为十进制数值。如果该数值小于 10000,则将其转换成 6 位数字,并在其前面加上 3202 作为 AI 前缀,形成数据报文的 AI 单元数据串;否则,从该数值中减去 10000,再在得出的数字前填充 0 以形成 6 位数字,AI 前缀 3203 添加到 6 位数字前,形成数据报文的第 2 个 AI 单元数据串。

举例说明编码方法字段“0101”,对(01)90012345678908(3202)000156 进行编码,实际上只将粗体数字部分编码成压缩数据字段。译码器将传输]e0019001234567890832020001 56。

7.2.5.4.4 编码方法字段“0111000”～“0111111”——可变质量贸易项目加日期

这些编码方法字段可在被编码的数据信息包含 2 个或 3 个 AI 单元数据串,即 AI(01)、AI(310x)或 AI(320x)(x 从 0～9)、以及 AI(11)或 AI(13)或 AI(15)或 AI(17)中的任何一个的情况下使用。AI(01)项目标识单元数据串必须具有包装指示符数字“9”,可变质量 AI 单元数据串可以是 0～099999 之间的任意值。编码方法字段“0111000”～“0111111”用于不含有通用数据压缩字段的、具有 8 个符号字符的固定长度条码符号的编码。

对可变质量产品及日期进行编码的八种方法：

1. 用编码方法字段“0111000”定义 AI(01)＋AI(310x)＋AI(11)，附加信息为公制质量及生产日期；

2. 用编码方法字段“0111001”定义 AI(01)＋AI(320x)＋AI(11)，附加信息为英制质量及生产日期；

3. 用编码方法字段“0111010”定义 AI(01)＋AI(310x)＋AI(13)，附加信息为公制质量及包装日期；

4. 用编码方法字段“0111011”定义 AI(01)＋AI(320x)＋AI(13)，附加信息为英制质量及包装日期；

5. 用编码方法字段“0111100”定义 AI(01)＋AI(310x)＋AI(15)，附加信息为公制质量及保质期；

6. 用编码方法字段“0111101”定义 AI(01)＋AI(320x)＋AI(15)，附加信息为英制质量及保质期；

7. 用编码方法字段“0111110”定义 AI(01)＋AI(310x)＋AI(17)，附加信息为公制质量及有效期；

8. 用编码方法字段“0111111”定义 AI(01)＋AI(320x)＋AI(17)，附加信息为英制质量及有效期。

3 个 AI 单元数据串被压缩成 76 个二进制位的压缩数据字段，其中，项目标识为 40 个二进制位，质量标识为 20 个二进制位，日期为 16 个二进制位。

在对这 3 个 AI 单元数据串进行编码时，先从 AI(01)单元数据串中去掉前 3 个数字 019 及末尾的校验码，其余的 12 个数字在 40 个二进制位的压缩数据字段中进行编码，分为 4 组，每组 3 个数字被编码为 10 个二进制位。再从质量单元数据串中去掉前 3 个数字 310 或 320，然后从余下的 7 个数字的 AI 单元数据串数据中将第 2 个数字(一个 0)去掉，形成 6 个数字。这 6 个数字由最后 1 位 AI 数字及后 5 位质量数字组成，将其编码成 20 个二进制位并添加到压缩数据字段中。16 个二进制位的日期压缩数据是去掉 AI(11、13、15 或 17)两位 AI 标识符，剩下的 6 位“YYMMDD(年月日)”数字按下式转换成值在 0～38399 之间的 16 位二进制压缩数据，再添加到压缩数据字段中。

(YY×384)＋((MM－1)×32)＋(DD)

在 16 个二进制位子字段中的值 38400，用来指示没有日期字段被编码。当需要对编码方法字段“0100”或“0101”不支持的质量值编码时，可选择这种方式来对贸易项目标识与质量值编码。

译码时，译码器将前 40 位二进制数分 4 个组(每组 10 位二进制数)译码成 12 个十进制数(每组 3 个数)，在这 12 个数的前面增加前缀“019”，后面增加计算出的模 10 校验码，成为第 1 个 AI 单元数据串。再将其余的 20 位二进制数转换成 6 位十进制数字，并在第 1 个数字后面插入一个 0 以形成 7 位数字，再在这 7 位数字前面加上 310 或 320 作为 AI 前缀，形成数据报文的第 2 个 AI 单元数据串；最后将剩下的 16 位二进制数转换成十进制数值，如果这最后 16 位的值为 38400，则不再对其他数据进行译码。

如果对日期进行了编码，译码器将根据上述公式提取 6 位日期数字，并由译码器根据编码方法字段的定义添加一个相应的应用标识符 AI(11、13、15 或 17)前缀，再将这 8 位数字作为第 3 个 AI 单元数据串添加到译码报文中。

举例说明编码方法字段“0111000”，对(01)90012345678908(3103)012233(15)991231 进行编码，实际上只将粗体数字编码成压缩数据字段。译码器将传输]e0019001234567890831030122331599123l。

7.2.5.4.5 编码方法字段“01100”——变量贸易项目及应付款金额

此编码方法字段可在被编码的数据信息以两个 AI 单元数据串——AI(01)跟接 AI(392x)开始的情况下使用。AI(01)项目标识单元数据串必须具有包装指示符数字 9，AI(392x)中表示小数点位置的数字 x，其值只能为 0、1、2 或 3。AI(01)单元数据串被压缩成 40 位二进制数，而 AI(392x)中表示小数点位置的数字 x 被编码成两位二进制数。然后用数字编码模式将价格数字编码，放在通用数据压缩字段中。利用通用数据压缩字段编码方法可对任何附加的 AI 单元数据串进行编码。

在对上述两个 AI 单元数据串进行编码时，先从 AI(01)单元数据串中去掉前 3 个数字 019 及末尾

的校验码。其余的12个数字在40个二进制位中编码成压缩数据，分为4组，每组3个数字被编码为10个二进制位；再从AI(392x)单元数据串中去掉前3个AI数字392，对表示小数点位置的数字(允许值为0～3)直接编码成后2个二进制位，这样形成总长度为42个二进制位的压缩数据字段。位于表示小数点位置的数字后面的数据用数字编码模式编码成通用数据压缩字段。

译码时，译码器将前40位二进制数分4个组(每组10位二进制数)译码成12个十进制数(每组3个数)，在这12个数的前面增加前缀019；后面增加计算出的模10校验码，成为第一个AI单元数据串。添加3位AI数字392，其后跟被编码成2个二进制位的表示小数点位置的十进制数字，最后将剩余的通用数据压缩字段部分译码。

举例说明编码方法字段"01100"，对(01)90012345678908(3922)795进行编码，实际上只将粗体数字编码成压缩数据字段，译码器将传输]e00190012345678908392795。

7.2.5.4.6 编码方法字段"01101"——变量贸易项目及带有ISO货币代码的应付款金额

此编码方法字段可在被编码的数据信息以两个AI单元数据串——AI(01)跟接AI(393x)开始的情况下使用。AI(01)项目标识单元数据串必须具有包装指示符数字9，AI(393x)中表示小数点位置的数字x，其值只能为0、1、2或3。AI(01)单元数据串被压缩成40位二进制数，而AI(393x)中表示小数点位置的数字x被编码成2位二进制数。3位数字的ISO货币码被编码成10位二进制数。然后再用数字编码模式将价格数字编码，放在通用数据压缩字段中。

在对上述两个AI单元数据串进行编码时，先从AI(01)单元数据串中去掉前3个数字"019"及末尾的校验码，其余的12个数字在40个二进制位中编码成压缩数据，分为4组，每组3个数字10个二进制位；再从AI(393x)单元数据串中去掉前3个AI数字393，接下来的表示小数点位置的数x(允许值为0～3)直接编码成后面的2位二进制数；再将后3位ISO货币代码直接编成为后10位二进制数，这样形成总长度为52个二进制位的压缩数据字段。然后再用数字编码模式将3位ISO货币代码后面的数据编码，放在通用数据压缩字段中。

译码时，译码器将前40位二进制数分成4个组(每组10位二进制数)译码成12个十进制数(每组3个数)，在这12个数的前面增加前缀"019"；后面增加计算出的模10校验码，成为第一个AI单元数据串。添加3位AI数字393，其后跟被编码成两个二进制位的表示小数点位置的数字，并将其后的10位二进制数直接译码成3位数字的ISO货币码，最后将剩余的通用数据压缩字段部分译码。

举例说明编码方法字段"01101"，对(01)90012345678908(3932)0401234进行编码，实际上只将黑体数字编码成压缩数据字段。译码器将传输]e00190012345678908393320401234。

7.2.5.5 通用数据压缩字段

通用数据压缩字段采用3种不同编码模式的组合将AI单元数据串编码成二进制数字串：

a) 数字编码模式用来对数字及FNC1字符进行编码，每2个字符要求用7位二进制位编码；

b) 字母数字编码模式用来对数字及大写字母混合的字符进行编码，每一数字或FNC1字符要求用5位二进制位编码，每一字母要求用6位二进制位编码；

c) "GB/T 1988"编码模式用来对英文大小写字母、数字及大多数标点符号进行编码，每一数字、字母及标点符号分别要求用5、7、8位二进制位编码。

除那些需用特定压缩数据编码方法来进行编码的AI单元数据串外，通用数据压缩字段可对各种AI单元数据串进行编码。通用数据压缩字段是可变长度条码符号的最后字段。一旦利用通用数据压缩字段编码方法对全部数据进行了处理，通用数据的编码过程将以7.2.5.5.4中所描述的填充步骤结束。

7.2.5.5.1 数字编码模式

数字编码模式是一种默认的编码模式，在通用数据压缩字段编码的开始就有效。它将两位数字或一位数字与一个FNC1字符的组合按各自的顺序编码在7位二进制数中，这7位二进制数的值的计算公式如下：

value=(11×D_1)+D_2+8

其中 D_1、D_2 为数字值或 FNC1 字符的值(FNC1 被赋值为 10)。这些 7 位二进制数"0001000"～"1111111"相应的数值范围为 8～127(两个 FNC1 字符不能被编码成 7 位二进制数)。位于通用数据压缩字段的开头或者位于前一个 7 位二进制位的数字编码之后的由 4 个"0"组成的二进制序列"0000"(即字母数字锁定码),发出一个锁定或转变到字母数字编码模式的信号(见表 11)。

表 11 数字编码模式

字符	编码的二进制数字
数字-数字、数字-FNC1 以及 FNC1-数字对	0001000～1111111
字母数字锁定码	0000

数字编码模式持续地对数据的字符对进行编码,直到满足下列条件之一为止:

a) 如果剩下两个或多个字符,且不能用数字编码,则将 1 个字母数字锁定码编入数据压缩字段中。

b) 如果还剩下 1 个非数字的字符,则将 1 个字母数字锁定码编进数据压缩字段中。

c) 如果还剩下 1 个数字的字符,则首先计算当前二进制数字串编码所需的条码符号大小,然后再加大未使用位数,使得总位数为 12 的偶数倍。

1) 如果未使用位数大于或等于 7 位,则将数字及 1 个 FNC1 填充字编码进下 1 个 7 位中。该 FNC1 字将作为填充字,识读器不对其进行译码;

2) 如果未使用位数为 4～6 位,则将数字值加上 1 后再将其编码进下一个 4 位中;

3) 否则,使用更大的符号,将数字及 FNC1 填充字编码到下一个 7 位中,该 FNC1 字将作为填充字,识读器不对其进行译码;

采用 7.2.5.5.4 中所描述的填充步骤对未使用位进行编码。

d) 已不剩任何字符,根据 7.2.5.5.4 中所描述的填充步骤对任何剩下的位进行编码。

只要字母数字锁定码被编码,编码模式即行转变以响应该锁定码。如果数据的下 1 个字符需要"GB/T 1988"编码模式,则在字母数字锁定码之后编入 1 个"GB/T 1988"锁定码。

在译码时,当数字压缩在条码符号的末端有效时,必须进行以下特殊检查:

a) 如果紧靠填充序列前面的最后 7 位二进制位数字编码是 1 个后接 FNC1 字符的数字时,则忽略该 FNC1 字符。

b) 如果在条码符号中仅剩的二进制位为 4～6 位时,数字编码仍有效,则将剩下位数中的前 4 位的二进制数字串转换为十进制值。

1) 如果此值为 0,则数据信息结束;

2) 否则,它将被译码,将该十进制值减去 1 的结果作为数据信息的最后 1 位数据。

7.2.5.5.2 字母数字编码模式

字母数字编码模式对数字、FNC1/数字锁定码、大写字母、5 个标点符号和两个锁定码字符进行编码。字符的编码二进制位数字串不具有固定的位长,根据每一被编码字符的位长分配位数,字符被编码成 3～6 个二进制位不等,如表 12 所示。

表 12 字母数字编码模式

字符	ASCII 值	编码的值	编码的二进制数字
0～9	48～57	ASCII 值减去 43(5 位二进制数)	00101～01110
FNC1/数字锁定码		15(5 位二进制数)	01111
A～Z	65～90	ASCII 值减去 33(6 位二进制数)	100000～111001
*(星号)	42	58(6 位二进制数)	111010
,(逗号)	44	59(6 位二进制数)	111011
-(减号或连字符)	45	60(6 位二进制数)	111100
.(句号)	46	61(6 位二进制数)	111101
/(斜线或斜杠)	47	62(6 位二进制数)	111110
数字锁定码		0(3 位二进制数)	000
"GB/T 1988"锁定码		4(5 位二进制数)	00100

通过把每一个字符的编码二进制数字添加到通用数据压缩字段中来对数据编码。下列情况除外：

a) 如果数据的下一个字符是 FNC1，则用字母数字编码模式将其编码；

b) 如果数据的下一个字符只能用"GB/T 1988"编码模式来进行编码，则将一个"GB/T 1988"锁定码编码到通用数据压缩字段中；

c) 如果数据后续的 6 个字符可以用数字编码模式来进行编码，则将一个数字锁定码编码到通用数据压缩字段中；

d) 如果数据后续的 4 个或 5 个字符可以用数字编码模式来进行编码，并且位于数据字符串的末端，则将一个数字锁定码编码到通用数据压缩字段中。

只要锁定码被编码，编码模式即行转变以响应该锁定码。

译码时，通过首先检查紧随前面已经译码字符后字段中的前 1 位或前 3 位即字段的开头，对已编码的二进制位字段进行译码。

a) 如果第 1 位为"1"，则将下一个字符作为一个 6 二进制位的字符进行译码；

b) 如果前 3 位为"000"，则它是一个数字锁定码；

c) 否则，将下一个字符作为一个 5 二进制位的字符进行译码。

7.2.5.5.3 "GB/T 1988"编码模式

此模式用于对扩展式 RSS 条码表示的 GB/T 1988 规定的信息交换用七位编码字符集中的数字、大小写字母、21 个标点符号，以及 FNC1 字符和两个锁定码字符进行编码。字符的编码二进制数字串不具有固定的位长，根据每一个被编码字符的位长来分配位数。字符被编码成 3～8 位，如表 13 所示。

表 13 "GB/T 1988"编码模式

字符	ASCII 值	编码的值	编码的二进制数字
0～9	48～57	ASCII 值减去 43(5 位二进制数)	00101～01110
FNC1/数字锁定码		15(5 位二进制数)	01111
A～Z	65～90	ASCII 值减去 1(7 位二进制数)	1000000～1011001
a～z	97～122	ASCII 值减去 7(7 位二进制数)	1011010～1110011
!(惊叹号)	33	232(8 位二进制数)	11101000
"(双引号)	34	233(8 位二进制数)	11101001
%(百分比号)	37	234(8 位二进制数)	11101010

表 13(续)

字符	ASCII 值	编码的值	编码的二进制数字
&(与号)	38	235(8 位二进制数)	11101011
'(单引号)	39	236(8 位二进制数)	11101100
((左圆括号)	40	237(8 位二进制数)	11101101
)(右圆括号)	41	238(8 位二进制数)	11101110
*(星号)	42	239(8 位二进制数)	11101111
+(加号)	43	240(8 位二进制数)	11110000
,(逗号)	44	241(8 位二进制数)	11110001
-(减号或连字符)	45	242(8 位二进制数)	11110010
.(句号)	46	243(8 位二进制数)	11110011
/(斜线或斜杠)	47	244(8 位二进制数)	11110100
:(冒号)	58	245(8 位二进制数)	11110101
;(分号)	59	246(8 位二进制数)	11110110
<(小于号)	60	247(8 位二进制数)	11110111
=(等号)	61	248(8 位二进制数)	11111000
>(大于号)	62	249(8 位二进制数)	11111001
?(问号)	63	250(8 位二进制数)	11111010
_(下划线或低线)	95	251(8 位二进制数)	11111011
空格	32	252(8 位二进制数)	11111100
数字锁定码		0(3 位二进制数)	000
字母数字锁定码		4(5 位二进制数)	00100

通过把每 1 个字符的编码二进制数字添加到通用数据压缩字段中来对数据编码。下列情况除外：

a) 如果数据的下一个字符是 FNC1，则采用“GB/T 1988”编码模式对其编码；

b) 如果数据接下来的 4 个字符可以用数字编码模式进行编码，且后续的 10 个字符可以不采用“GB/T 1988”编码模式进行编码，则将一个数字锁定码编码进通用数据压缩字段中；

c) 如果接下来的 5 个数据符可以用字母数字编码模式进行编码，且后续的 10 个字符可以不采用“GB/T 1988”编码模式进行编码，则将一个字母数字锁定码编码进通用数据压缩字段中。

作为上述各个情况的例外，如果数据以少于 10 个字符结束，则先在数据末端完成 10 个字符的测试。如果锁定码被编码，编码模式即行转变以响应该锁定码。

译码时，通过首先检查紧随前面已经译码字符后字段中的前 3 位或前 5 位即字段的开头，对已编码的二进制位字段进行译码。

a) 如果前 3 位为“000”，则它是一个数字锁定码；

b) 否则，得出前 5 位二进制数的十进制值，如果该值：

1) 小于或等于 15，则将下一个字符作为 1 个 5 位二进制位的字符译码；

2) 在 16～28 之间，则将下一个字符作为 1 个 7 位二进制位的字符译码；

3) 大于或等于 29，则将下一个字符作为 1 个 8 位二进制位的字符译码。

7.2.5.5.4 用于通用数据压缩字段的填充位

条码符号中的符号字符数，应为对该符号表示的数据进行编码所需的最小值。但是，在数据被编码

到通用数据压缩字段中后，可能会存在一些未使用的二进制位，这些位将用填充序列来填充直到符号的数据容量被填满为止。

填充位串通过重复5位的填充序列"00100"而产生，此序列既是字母数字编码模式中的"GB/T 1988"锁定码，又是"GB/T 1988"编码模式中的字母数字锁定码，因此对于编码模式及锁定码的转换，无需编码更多的数据。如果没有足够的未使用位留在符号中，则可将最后一个填充序列舍去。

如果数据编码以数字编码结束，则在交替更换的"00100"锁定码/填充序列之前要求有一个4位的字母数字锁定码"0000"。例如，如果编码以数字编码结束，同时还留有7个未使用位，则它们将被编码成"0000001"，也就是在字母数字锁定码"0000"之后接上GB/T 1988模式锁定码"00100"的前3位"001"。如果要填充的位少于4个未使用位，则第一个4位的锁定码本身将被缩短。

7.2.6 校验符

扩展式RSS符号中的第一个符号字符为校验符，它对符号长度(符号字符总数)及数据符单元宽度加权的"校验和"进行编码。只有前面的4009个校验符值(0～4008)被用到。

符号中的符号字符数S(4～22)及校验和的值被编码到校验符中，即：

校验符值$=211\times(S-4)+$校验和的值，其中，S为符号字符数。

校验和的值等于条码符号中所有的数据符单元宽度加权和的模211运算的结果。

加权的模211校验和的值由下式计算：

$$(W_{1,1}E_{1,1}+W_{1,2}E_{1,2}+\cdots+W_{1,8}E_{1,8}+\cdots+W_{x,8}E_{x,8})\ \mathrm{mod}\ 211$$

式中：

$W_{N,M}$——在表14中的标号为N的数据符中序号为M的单元的权；

$E_{N,M}$——标号为N的数据符中M单元的宽度模块数；

$W_{N,M}E_{N,M}$——两者的乘积；

下标x——条码符号中标号最大的数据符的标号。

表14中的权是3的连续次幂模211运算的结果，用公式$W_{N,M}=3^{M+8N-9}\ \mathrm{mod}\ 211$计算。

注：数据符的标号与数据符的序号是不同的。校验和值的计算中出现的数据符标号是实际的条码符号中所有数据符的标号，它们的出现不一定是连续的，而是根据条码符号中符号字符的个数，由表16中的序列决定的。

附录F.3给出了一个扩展式RSS符号的编码示例。

表14 模211校验和计算的数据符单元的权

数据符		数据符单元序号(M)							
与定位符位置关系	标号(N)	1	2	3	4	5	6	7	8
A1右	1	1	3	9	27	81	32	96	77
A2左	2	20	60	180	118	143	7	21	63
A2右	3	189	145	13	39	117	140	209	205
B1左	4	193	157	49	147	19	57	171	91
B1右	5	62	186	136	197	169	85	44	132
B2左	6	185	133	188	142	4	12	36	108
B2右	7	113	128	173	97	80	29	87	50
C1左	8	150	28	84	41	123	158	52	156
C1右	9	46	138	203	187	139	206	196	166
C2左	10	76	17	51	153	37	111	122	155
C2右	11	43	129	176	106	107	110	119	146

表 14(续)

数据符与定位符位置关系	数据符标号(N)	数据符单元序号(M) 1	2	3	4	5	6	7	8
D1 左	12	16	48	144	10	30	90	59	177
D1 右	13	109	116	137	200	178	112	125	164
D2 左	14	70	210	208	202	184	130	179	115
D2 右	15	134	191	151	31	93	68	204	190
E1 左	16	148	22	66	198	172	94	71	2
E1 右	17	6	18	54	162	64	192	154	40
E2 左	18	120	149	25	75	14	42	126	167
E2 右	19	79	26	78	23	69	207	199	175
F1 左	20	103	98	83	38	114	131	182	124
F1 右	21	161	61	183	127	170	88	53	159
F2 左	22	55	165	73	8	24	72	5	15
F2 右	23	45	135	194	160	58	174	100	89

注 1：N 为数据符的标号，例如“C1 右”是指定位符 C1 右边的数据符，标号为 9。

注 2：定位符 A1 左边的符号字符是校验符而不是数据符，校验符不进行单元加权。

7.2.7 定位符

扩展式 RSS 条码符号有 12 个唯一的定位符，这些定位符位于符号字符对之间。由于定位符与符号字符相邻，可以对条码符号的由符号字符与相邻定位符组成的段分段进行扫描和译码。

如果条码符号中含有奇数个符号字符，则符号以最后一个定位符及右侧保护符结束。在这种情况下，最后一个定位符位于最后一个数据符的右边，并与其相邻。

这 12 个定位符是基于六个从 A 至 F 的基本图形，每一基本图形都具有两种形式(形式 1 和形式 2)。形式 1 图形的单元 1 是最左侧的空，而在形式 2 图形中相对应的单元是最右侧的条。这种条—空的倒转可用来区分两种形式的图形。12 个定位符分别为 A1、A2、B1、B2、C1、C2、D1、D2、E1、E2、F1 及 F2。采用形式 1 图形的定位符最左侧的空作为单元 1，而采用形式 2 图形的定位符则是将形式 1 图形从左至右镜像并且条空转换得到的图形。表 15 列出了这 6 个基本图形各单元的宽度。

每一个定位符都由 5 个单元组成，共 15 个模块。“形式 1”的定位符中的单元 2 及单元 3 中的模块之和为 10～12，而单元 4 及单元 5 中的模块之和均为 2。宽单元对(单元 2 和单元 3)与从第 2 至第 5 这 4 个单元的总宽度之比介于 10∶12～12∶14 之间，这也是定位符逻辑识别的第 1 步的基础。同样，“形式 2”的定位符则是将第 1～第 4 这 4 个单元的总宽度与宽单元对(单元 3 和单元 4)的宽度进行比较。

表 15 定位符的单元宽度

基本图形(形式 1)	单元宽度(单元 1 为空) 单元 1	单元 2	单元 3	单元 4	单元 5	基本图形(形式 2)	单元宽度(单元 1 为条) 单元 1	单元 2	单元 3	单元 4	单元 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

定位符以 10 种依据符号长度分配的唯一的集合方式来在符号中使用(见表 16)。这些集合被分成两个组,在每一组中,每一集合至少有一个唯一的定位符子集,以将其与组中的其他集合区分开。这种集合的选择可避免由错误译码的校验符引起的误读而得出错误的符号长度。

表 16 定位符序列

条码符号中段的数目	定位符顺序										
	1	2	3	4	5	6	7	8	9	10	11
组 1											
4	A1	A2									
5 或 6	A1	B2	B1								
7 或 8	A1	C2	B1	D2							
9 或 10	A1	E2	B1	D2	C1						
11 或 12	A1	E2	B1	D2	D1	F2					
13 或 14	A1	E2	B1	D2	E1	F2	F1				
组 2											
15 或 16	A1	A2	B1	B2	C1	C2	D1	D2			
17 或 18	A1	A2	B1	B2	C1	C2	D1	E2	E1		
19 或 20	A1	A2	B1	B2	C1	C2	D1	E2	F1	F2	
21 或 22	A1	A2	B1	B2	C1	C2	D1	E2	E1	F2	F1

7.2.8 层排扩展式 RSS

层排扩展式 RSS 可以排列为 2 行～11 行,每一行为 34X 高,行与行之间有一个 3X 高的层分隔符。图 12 给出了一个两行符号的示例,图 12 中符号表示的数据与图 10 中符号表示的数据相同,以便于进行比较。当符号可用区域或印制装置的宽度不足以容纳整个单行的符号时,使用层排扩展式 RSS。

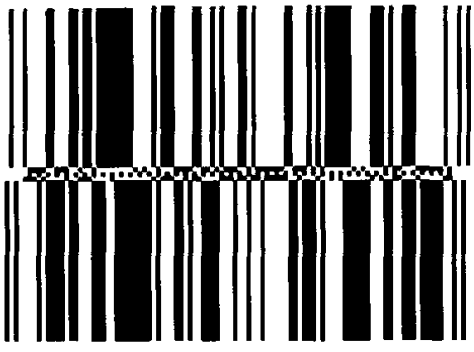


图 12 表示数据(01)98898765432106(3202)012345(15)991231 的层排扩展式 RSS 符号

层排扩展式 RSS 符号的行应按从上到下的顺序进行排序。除最后一行外,符号每行中的符号字符数均应为偶数。最后一行最少应含有两个带额外填充(如果需要)的符号字符。当层排扩展式 RSS 符号是 EAN·UCC 复合码的一个线性部分时,符号中的第一行最少应具有 4 个符号字符,二维部分应打印在符号第一行的上面。

在每行中,符号字符对之间都有一个定位符,且每行都具有各 2 个单元的左、右保护符。最后一行可以含有奇数个符号字符,此时最后一个定位符紧靠最右边的保护符。

除最底行以外,第一行及其后的奇数行将以空开始,而第二行及其以后的偶数行则将以条开始,参

见图 13。如果每行含有偶数个段对(如 2、4 个对,也就是 4、8 个段等),则将以倒转的单元次序来打印偶数行,也就是作为一个镜像来打印,保证偶数行以条开始的。如果每行含有奇数个段对(如 1、3、5 个对,也就是 2、6、10 个段等),则偶数行将自动以条开始。表 17 列出了这些倒转行。

表 17 层排扩展式 RSS 中的倒转行

行序号	行中包含的段的数目									
	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	-		-			—		-	

注:"F"为正常顺序的行,"R"为倒转行,"—"表示不可能出现的情况。

使用一种由 3 个 1X 高的行组成的 3X 高的层分隔符来分隔层排扩展式 RSS 条码符号的行。

层分隔符各行的前 4 个和最后 4 个模块总是空。

除了"形式 1"的定位符的第 1、第 2、第 3 单元和"形式 2"的定位符的第 3、第 4、第 5 单元之下的各 13 个模块之外,层分隔符第一行的单元是由相邻的符号上面行条/空颜色的互补色形成的。上述各 13 个模块,在相邻定位符的条之下的是浅色,在相邻定位符的空之下的是以深色开始的深浅交替的形式。

除了层分隔符行的两端的各 4 个模块之外,层分隔符的第二行由交替的条空模块组成。

除了"形式 1"的定位符的第 1、第 2、第 3 单元和"形式 2"的定位符的第 3、第 4、第 5 单元之上的各 13 个模块之外,层分隔符第三行的单元是由相邻的符号下面行条/空颜色的互补色形成的。上述各 13 个模块,在相邻定位符的条之上的是浅色,在相邻定位符的空之上的是以深色开始的深浅交替的形式。

图 12 所示的层排扩展式 RSS 符号有 8 个符号字符,层排为两行,每一行有 4 个符号字符。第一行由前 4 个符号字符组成,并在右边增加有一个 1X 的空及 1X 的条的保护符。而第二行则打印为一个镜像图案,它从左边开始,打印的是后 4 个符号字符的镜像图案,并附加一个 1X 的条及 1X 的空的保护符。图 12 中示例的符号的整体尺寸为 102X 宽、71X 高。

对于行倒转的要求有一个例外,如果最后一行应从左至右进行镜像,但它又是一个含有奇数个定位符的不完全宽度行,则将其向右偏移一个模块并在行的左边增加一个"浅"模块。这是一个必要的措施,因为含有奇数个定位符的某些行拥有对称的保护条,而它在倒转后的图形将与倒转前一样,图 13 给出了一个要求将最后一行进行偏移的层排扩展式 RSS 的例子。

图 13 表示数据(01)95012345678903(3103)000123 的层排扩展式 RSS 条码符号

7.2.9 参考译码算法

条码识读系统设计成在现行算法允许的范围内识读有缺陷的条码符号。这部分叙述了 GB/T 14258 中描述的用于检测符号质量的可译码度值计算中使用的参考译码算法。算法包括下列译码步骤：

a) 通过从左到右和从右到左寻找一段 4 个单元的序列并计算其中相应单元宽度的比，找到符号：

1) 从左到右：

9.5∶12≤(单元 1+单元 2)∶(单元 1+单元 2+单元 3+单元 4)≤12.5∶14

2) 从右到左：

9.5∶12≤(单元 3+单元 4)∶(单元 1+单元 2+单元 3+单元 4)≤12.5∶14

注：上面的单元 1、单元 2、单元 3、单元 4 的序号是一段 4 个单元的序列中的单元序号，顺着扫描方向排序，它们与定位符中单元的序号是不同的。

通过上述 1)或 2)比率的确定可识别出“形式 1”的定位符的第 2 个～第 5 个单元。采用同样的方法可识别出“形式 2”的定位符的第 1 个～第 4 个单元，但要将上面 1)中的“从左到右”改为“从右到左”；将上面 2)中的“从右到左”改为“从左到右”。

使用步骤 b)中的方法对定位符进行译码，利用“形式 1”定位符前 4 个单元或“形式 2”定位符后 4 个单元的宽度和(p)，以找到标称的相似边缘间距数值 E_1 和 E_2，因为定位符前或定位符后 4 个单元的宽度和(p)的模块总数为 14，此时需将步骤 b)2)中与 p 相除的数由 17 改为 14。验证数值 E_1 和 E_2 是否符合有效的扩展式 RSS 的定位符。

b) 检查每一相邻的符号字符与定位符间距之间有效比值是否为(17±1.5)∶15，然后每一符号字符按下列步骤译码：

1) 获得 7 个宽度的测量值 p、e_1、e_2、e_3、e_4、e_5 和 e_6(图 14)。

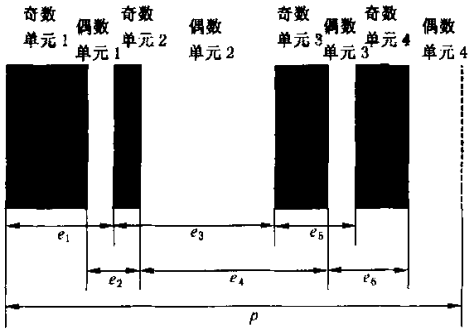


图 14 译码测量

注：上图表示左起条单元为第一个单元，但数据符也可以是上图从左到右镜像或条空反转的形式。

2) 将测量值 e_1、e_2、e_3、e_4、e_5 和 e_6 转换为表示整数模块宽度(E_i)的标称值 E_1、E_2、E_3、E_4、E_5

和 E_6。下面的方法用于 $E_i(i=1,2,\cdots,5,6)$的确定：

如果 $1.5p/17 \leqslant e_i < 2.5p/17$，那么 $E_i=2$；

如果 $2.5p/17 \leqslant e_i < 3.5p/17$，那么 $E_i=3$；

如果 $3.5p/17 \leqslant e_i < 4.5p/17$，那么 $E_i=4$；

如果 $4.5p/17 \leqslant e_i < 5.5p/17$，那么 $E_i=5$；

如果 $5.5p/17 \leqslant e_i < 6.5p/17$，那么 $E_i=6$；

如果 $6.5p/17 \leqslant e_i < 7.5p/17$，那么 $E_i=7$；

如果 $7.5p/17 \leqslant e_i < 8.5p/17$，那么 $E_i=8$；

如果 $8.5p/17 \leqslant e_i < 9.5p/17$，那么 $E_i=9$。

否则字符出错。

3) 从 E 值确定符号字符各单元的标称宽度。$n(n=17)$个模块中剩余的模块分配给最后一个单元，得出该单元的宽度，而不是从 E 值中计算出来。有效的单元宽度集合是没有单元宽度小于一个模块、并且至少有一个偶数单元是一个模块宽。例如：图 14 中 $E_1 \sim E_6$ 的值是{4 2 6 7 3 3}。可能的单元宽度的集合是{2 2 0 6 1 2 1 3}(注意不应有 0 宽度单元)、{3 1 1 5 2 1 2 2}或{4 0 2 4 3 0 3 1}，其中只有 8 个单元宽度为{3 1 1 5 2 1 2 2}的集合满足要求，因此被选作字符各单元的宽度。如果导出的单元宽度的集合都是无效的，那么字符出错。附录 G 给出了单元宽度译码算法的 C 语言程序。

4) 采用附录 B 中的程序确定奇子集和偶子集的值。

5) 从奇子集和偶子集数值计算数据符的值。

6) 查看表 15 表示的图形，用上述从 E 值确定标称单元宽度的方法对定位符译码。

7) 计算并存储单元宽度加权的和用于校验和的计算。

c) 当校验符中标明的全部数据符和定位符被译码之后，验证从校验符得到的模 211 的校验和值与数据符单元宽度加权的和模 211 的校验和值相符合。

d) 将数据符值转换为二进制数字串，再转换为 AI 单元数据串。

e) 此外，为稳妥起见，考虑到具体的识读设备和设想中的应用环境，需对扫描加速度、绝对计时以及尺寸等进行其他的附加检查。

设计实际的识读扩展式 RSS 的扫描器时，可参见附录 H 中为使误读最小化而附加的符号译码考虑。

8 符号质量

8.1 一维条码符号质量参数

RSS 系列符号采用 GB/T 14258 定义的对条码符号进行测量和评级的标准方法进行质量评价。应该把本标准中定义的 RSS 系列符号的参考译码算法用于 GB/T 14258 标准中对“译码”及“可译码度”等参数的评价。层排式 RSS 的层分隔符的所有单元应是人眼可辨识的，在对符号质量进行评价时，不对层分隔符进行评级。有关符号打印方面的指导见附录 J。

8.2 附加的判定规则

国家标准 GB/T 14258 允许条码码制标准规定一些附加的判定规则。对于 RSS 条码，附加的规则是在每次尝试扫描时，两个保护符内侧的单元必须存在，且宽度不大于 3Z。对不满足此项要求的任何单独的扫描曲线应该判定为 0 级。

8.3 层排式符号的质量

依据 GB/T 14258，层排式 RSS 符号的每一行应作为一个单独的符号来进行评价。为将相邻行的窜行的影响降到最低，如 GB/T 14258 所规定的，扫描线应该通过符号每行的高度的 80%的检测带。符号每行扫描线数取 10 或行高除以测量孔径得到的值中的较小者。符号所有行的符号等级中的最小值

作为符号的总符号等级。

9 传输的数据

RSS 系列符号在设计和设想上要与 ISO/IEC 15424 规定的码制标识符一起使用。在使用 RSS 系列符号的应用中，需开启识读器的码制标识符处理功能。EAN·UCC 系统要求使用码制标识符。未使用码制标识符的应用系统将不能识别 RSS 符号中的应用标识符数据，或把其他码制条码符号的数据误认为是应用标识符数据。

可以使用一个码制标识符前缀“]e0”对 RSS 系列符号进行传输。如果 RSS 系列线性符号与一个二维部分相结合，AI 单元数据串的数据紧跟在线性部分数据后面。

例如，某个 RSS-14 或限定式 RSS 符号会被传输为]e0011001234567890 2，其中粗体部分 1001234567890 在符号中进行编码。而码制标识符前缀、作为项目标识的应用标识符“01”和 EAN/UCC 模 10 计算得到的校验码“2”被加在传输的数据串中。

扩展式 RSS 对应用标识符进行编码，因此只将码制标识符前缀作为编码数据的前缀附加在传输数据中。扩展式 RSS 可对非数据字符 FNC1 进行编码，在作为符号的最后一个字符时 FNC1 不被传输；其他情况下 FNC1 作为<GS>(ASCII 值 29)被传输。

对于 EAN·UCC 复合码，二维部分中的 AI 单元数据串，应在线性部分数据后直接被传输。通常情况下，如果线性部分设置了连接标志，则识读器必须对两部分进行译码。然而，识读器可能支持另外一个模式，即：只有一维部分被译码，并在传输过程中忽略连接标志。这种模式支持那些只要求有贸易项目主标识的应用场合。

识读器还支持选择 UCC/EAN-128 模拟模式。这种模式在进行数据传输时模拟 UCC/EAN-128 符号，其码制标识符为]C1。超出 48 个数据的字符的扩展式 RSS 符号将被以两条信息传输，每条信息都将拥有一个符号前缀]C1，并且不会超过 48 个数据的字符，信息将在两个单元数据串之间被分割，见附录 D。

10 供人识读字符

打印时，供人识读字符应以清晰的字体出现在条码符号之下。GB/T 15425 对供人识读字符提出了附加要求。

11 最小模块宽度(*X* 尺寸)

应通过考虑应用系统中条码生成设备和识读设备的技术性能来决定所能使用的 *X* 尺寸的最小值。对一般的应用场合，*X* 尺寸的推荐值为 0.25 mm。

在一个给定的符号中，*X* 尺寸应保持一致。当连接二维部分时，RSS 符号应具有与二维部分一样的 *X* 尺寸。

12 应用参数

GS1 通用规范中规定了 RSS 的数据容量、*X* 尺寸、最小符号高度、最低符号质量等级、符号类型、符号位置等参数，以及应用所要求的其他参数。

附 录 A
（规范性附录）
EAN·UCC 校验码的计算

RSS-14、层排式 RSS-14 以及限定式 RSS 可以对 EAN/UCC-14 编码结构进行编码，即可以对 14 位数字的全球贸易项目代码进行编码。采用应用标识“01”的扩展式 RSS 也可用来对 EAN/UCC-14 编码结构进行编码。通过使用应用标识符“00”，扩展式 RSS 还可用来对 SSCC-18 编码结构（系列货运包装箱代码）进行编码。

EAN/UCC-14、SSCC-18 编码结构和其中校验码的计算见表 A.1。

表 A.1 EAN·UCC 编码结构中校验码的计算

编码结构	数字及其位置																	
EAN/UCC-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}
计算步骤一	每个位置的数乘以下面相应的数值																	
	×3	×1	×3	×1	×3	×1	×3	×1	×3	×1	×3	×1	×3	×1	×3	×1	×3	
计算步骤二	乘积累加求和，结果＝SUM																	
计算步骤三	以大于或等于 SUM 的且与 SUM 最接近的 10 的整数倍的数减去 SUM，结果＝校验码（N_{14} 或 N_{18}）																	

EAN·UCC 校验码不在 RSS-14、限定式 RSS 及扩展式 RSS 符号中被编码，而是隐含在被编码的 EAN/UCC-14 结构的其他数字中。EAN·UCC 校验码能在供人识读字符中被显示。在译码时，译码器根据被译出的 EAN/UCC-14 结构的其他数字计算出 EAN·UCC 校验码并进行传输。

附 录 B
（规范性附录）
单元宽度编码和译码的C语言程序

符号的值通过计算被编码，得到每一个(n,k)字符的子集值，然后得出子集中各单元的宽度及单元排列的模式即单元宽度的序列。采用C语言的编码程序(getRSSwidths)根据子集值计算得到子集的单元宽度序列；采用C语言的译码程序(getRSSvalue)根据子集的单元宽度序列计算得到子集值。

为子集赋值即把连续值分配给各子集的单元宽度序列。赋值的规则是，排序靠前的宽度值越小的单元宽度序列，赋予的值越小。赋值从排序靠前的宽度值都是一个模块宽的子集，即前几个单元的宽度最窄且从有效的子集开始(被赋予0值的第一个子集是除最后一个单元外的单元都是一个模块宽，且最后一个单元的宽度不超出最大单元宽度限制的那个子集)。接下来的值将被赋予余下的子集中那个靠前序号单元的宽度最窄的有效子集。例如，6模块的子集包含有以下0～9的数值，对应的以模块为单位的单元宽度序列为：

子集值	单元宽度序列
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

单元宽度超出最大宽度(max Width)的单元宽度序列被跳过。如果单元宽度序列中所有的单元都超过一个模块宽(noNarrow=0)，则该序列被排除，并跳过。

```
/*********************************************************
* getRSSwidths(获得 RSS 单元宽度)
* 本程序对一个给定的值，生成 RSS 各单元的宽度
*
* 调用变量(或参数)：
* val=要求的值
* n=模块的数量
* elements=子集中的单元的数量(RSS-14 和扩展式 RSS=4，限定式 RSS=7)
* maxWidth=一个单元的最大宽度(单位为模块)
* noNarrow=0 将跳过那些没有一个模块宽的单元的单元宽度序列
*
* Return(返回)：
* static int widths [ ]=各单元的宽度
*********************************************************/
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))
    {
      /*取得所有组合*/
      subVal=combins (n-elmWidth-1, elements-bar-2);
      /*去掉没有单个模块单元的组合*/
      if ((! noNarrow) && (narrowMask==0) &&
        (n-elmWidth-(elements-bar-1) >=elements-bar-1))
      {
      subVal-=combins (n-elmWidth-(elements-bar), elements-bar-2);
      }
      /*去掉单元宽度超过最大要求值的组合*/
      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(获得 RSS 值)
* 对给定的各单元宽度,计算子集值的程序
*
* 调用变量(或参数):
* widths[ ]=给定的各单元宽度
* elements(单元数量)=在(RSS-14& 扩展式=4;限定式 RSS=7)子集中的单元数量
* maxWidth=一个单元的最大宽度(以模块为单位)
* noNarrow=0  将跳过那些没有一个模块宽的单元的单元宽度序列
*
* Return(返回);
* 子集值
*****************************************/
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)) {
     /* 取得所有(n,k)组合 */
      subVal=combins (n-elmWidth-1, elements-bar-2);
     /* 去掉没有单个模块单元的组合 */
      if ((! noNarrow) && (narrowMask==0) &&
               (n-elmWidth-(elements-bar-1) >=elements-bar-1))
     {
       subVal-=combins (n-elmWidth-(elements-bar),
```

```
                        elements－bar－2)；
    }
    /＊去掉单元宽度超过最大要求值的组合＊/
    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－elmWitdth>maxWidth)
      {
          subVal－－；
      }
      val+=subval；
    }
    n－=elmWidth；
  }
  return (val)；
}

/＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊ combins (n,r)：返回从n选择的r的组合数；
＊ Combinations(组合)=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);
}
```

附 录 C
（规范性附录）
限定式 RSS 校验符的单元宽度

限定式 RSS 条码符号校验符的各单元宽度及相应的单元宽度序列见表 C.1。

表 C.1 限定式 RSS 校验符的单元宽度

校验符的值	单元宽度序列号	单元宽度(从最左边的“空”到最右边的“条”)													
		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	2	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	3	1	2	3	1	1	1
5	5	1	1	1	1	1	1	1	1	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	2	1	2	3	1	1	1
8	8	1	1	1	1	1	2	1	1	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	2	1	2	3	1	1	1
12	12	1	1	1	2	1	1	1	1	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	2	1	2	3	1	1	1
17	17	1	2	1	1	1	1	1	1	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

表 C.1(续)

校验符的值	单元宽度序列号	单元宽度(从最左边的"空"到最右边的"条")													
		S1	B1	S2	B2	S3	B3	S4	B4	S5	B5	S6	B6	S7	B7
29	29	1	1	1	1	1	2	1	2	2	1	2	1	1	1
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	1	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

表 C.1(续)

校验符的值	单元宽度序列号	单元宽度(从最左边的"空"到最右边的"条")													
		S1	B1	S2	B2	S3	B3	S4	B4	S5	B5	S6	B6	S7	B7
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
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

注：S1、S2…S7 分别表示第 1 个空、第 2 个空…第 7 个空；B1、B2…B7 分别表示第 1 个条、第 2 个条…第 7 个条。

注：表 C.1 中的"单元宽度序列号"是用附录 B 中的程序"getRSSwidths(获得 RSS 宽度)"生成的以所有 8 个模块的组合中的前 6 个"空"与前 6 个"条"组配而形成的 21×21=441 种单元宽度序列的序列编号(编号为 0～440)，该序列编号的计算为：("空单元序列的值"×21)+"条单元序列的值"。在定义校验符集时可以使用序列号来代替条/空宽度序列。

生成条和空单元的"getRSSwidths(获得 RSS 宽度)"程序所调用的变量为：

val(值)="空单元序列的值"(对于空)或"条单元序列的值"(对于条)(为 0～20)

n(模块数量)=8

elements(单元数量)=6

maxWidth(单元最大宽度)=3

noNarrow=1

附 录 D
（规范性附录）
分割较长的扩展式 RSS 符号进行 UCC/EAN-128 模拟传输

对于超过 48 个数据的字符的扩展式 RSS 符号，在进行传输前，必须将其分割成两个 UCC/EAN-128 模拟的部分。分割应在超出 48 个字符限制的单元数据串的开头进行。单元数据串由 FNC1 分隔符确定，或者由表 D.1 中给出的固定长度的 AI 单元数据串的末端确定。识读器采用下面的步骤查找可能超出 48 个数据字符的单元数据串：

a) 如果扩展式 RSS 符号表示的数据超过 48 个字符，则从符号的起始位置开始检查；

b) 检查一维部分数据接下来的两个数字（AI 的前两个数字）；

c) 如果这两个数字包含在表 D.1 中，则跳过表 D.1 中该 AI 标识的字符数；

d) 如果表中不存在该 AI，则表明该 AI 字符串为非预定义长度，继续搜索数据直到第一次出现 FNC1 或者直到符号的末端（取二者中首先出现的）；

e) 如果发现字符数超出 48 个的限制，则在被处理的最后一个单元数据串的开始处将数据分割，否则转到步骤 b）。

如果第一条信息以 FNC1 结束，则不传输 FNC1。

表 D.1 预定义长度单元数据串的应用标识符及字符总数

应用标识符的前两个数字	字符数（包括应用标识符及数据字段）	应用标识符的前两个数字	字符数（包括应用标识符及数据字段）
00	20	18	8
01	16	19	8
02	16	20	4
03	16	23	$2n+4$[a]
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		

[a] 这里 n 为应用标识符的第 3 个数字（直接跟在 23 后面）。

只有那些头两个数字与表 D.1 中的数字相匹配的应用标识符单元数据串，才被认为是“固定长度”的。其他一些应用标识符的单元数据串可以自然形成只有一种长度，但是因为这些应用标识符不在预定义长度的表格中，这些单元数据串仍被认为是“可变长度”数据串。表 D.1 不会再增加其他预定义长度的 AI 数据串。

附 录 E
（资料性附录）
RSS 条码符号的单元

表 E.1、表 E.2、表 E.3 分别描述了各种类型 RSS 符号按顺序排列的各个单元。

表 E.1 RSS-14 的单元

单 元	类 型	描 述
1	空	左侧保护符的外侧单元(1 个模块宽)
2	条	左侧保护符的内侧单元(1 个模块宽)
3	空	数据符 1 的第 1 个奇数单元
4	条	数据符 1 的第 1 个偶数单元
5	空	数据符 1 的第 2 个奇数单元
6	条	数据符 1 的第 2 个偶数单元
7	空	数据符 1 的第 3 个奇数单元
8	条	数据符 1 的第 3 个偶数单元
9	空	数据符 1 的第 4 个奇数单元
10	条	数据符 1 的第 4 个偶数单元
11	空	左侧校验符(定位符)的第 1 个单元
12	条	左侧校验符(定位符)的第 2 个单元
13	空	左侧校验符(定位符)的第 3 个单元
14	条	左侧校验符(定位符)的第 4 个单元(1 个模块宽)
15	空	左侧校验符(定位符)的第 5 个单元(1 个模块宽)
16	条	数据符 2 的第 4 个偶数单元
17	空	数据符 2 的第 4 个奇数单元
18	条	数据符 2 的第 3 个偶数单元
19	空	数据符 2 的第 3 个奇数单元
20	条	数据符 2 的第 2 个偶数单元
21	空	数据符 2 的第 2 个奇数单元
22	条	数据符 2 的第 1 个偶数单元
23	空	数据符 2 的第 1 个奇数单元
24	条	数据符 4 的第 1 个奇数单元
25	空	数据符 4 的第 1 个偶数单元
26	条	数据符 4 的第 2 个奇数单元
27	空	数据符 4 的第 2 个偶数单元
28	条	数据符 4 的第 3 个奇数单元
29	空	数据符 4 的第 3 个偶数单元
30	条	数据符 4 的第 4 个奇数单元

表 E.1(续)

单　元	类　型	描　述
31	空	数据符 4 的第 4 个偶数单元
32	条	右侧校验符(定位符)的第 5 个单元(1 个模块宽)
33	空	右侧校验符(定位符)的第 4 个单元(1 个模块宽)
34	条	右侧校验符(定位符)的第 3 个单元
35	空	右侧校验符(定位符)的第 2 个单元
36	条	右侧校验符(定位符)的第 1 个单元
37	空	数据符 3 的第 4 个偶数单元
38	条	数据符 3 的第 4 个奇数单元
39	空	数据符 3 的第 3 个偶数单元
40	条	数据符 3 的第 3 个奇数单元
41	空	数据符 3 的第 2 个偶数单元
42	条	数据符 3 的第 2 个奇数单元
43	空	数据符 3 的第 1 个偶数单元
44	条	数据符 3 的第 1 个奇数单元
45	空	右侧保护符的内侧单元(1 个模块宽)
46	条	右侧保护符的外侧单元(1 个模块宽)

表 E.2　限定式 RSS 的单元

单　元	类　型	描　述
1	空	左侧保护符的外侧单元(1 个模块宽)
2	条	左侧保护符的内侧单元(1 个模块宽)
3	空	左侧数据符的第 1 个单元及第 1 个奇数单元
4	条	左侧数据符的第 2 个单元及第 1 个偶数单元
5	空	左侧数据符的第 3 个单元及第 2 个奇数单元
6	条	左侧数据符的第 4 个单元及第 2 个偶数单元
7	空	左侧数据符的第 5 个单元及第 3 个奇数单元
8	条	左侧数据符的第 6 个单元及第 3 个偶数单元
9	空	左侧数据符的第 7 个单元及第 4 个奇数单元
10	条	左侧数据符的第 8 个单元及第 4 个偶数单元
11	空	左侧数据符的第 9 个单元及第 5 个奇数单元
12	条	左侧数据符的第 10 个单元及第 5 个偶数单元
13	空	左侧数据符的第 11 个单元及第 6 个奇数单元
14	条	左侧数据符的第 12 个单元及第 6 个偶数单元
15	空	左侧数据符的第 13 个单元及第 7 个奇数单元
16	条	左侧数据符的第 14 个单元及第 7 个偶数单元
17	空	校验符(定位符)的第 1 个单元

表 E.2(续)

单　元	类　型	描　述
18	条	校验符(定位符)的第 2 个单元
19	空	校验符(定位符)的第 3 个单元
20	条	校验符(定位符)的第 4 个单元
21	空	校验符(定位符)的第 5 个单元
22	条	校验符(定位符)的第 6 个单元
23	空	校验符(定位符)的第 7 个单元
24	条	校验符(定位符)的第 8 个单元
25	空	校验符(定位符)的第 9 个单元
26	条	校验符(定位符)的第 10 个单元
27	空	校验符(定位符)的第 11 个单元
28	条	校验符(定位符)的第 12 个单元
29	空	校验符(定位符)的第 13 个单元(1 个模块宽)
30	条	校验符(定位符)的第 14 个单元(1 个模块宽)
31	空	右侧数据符的第 1 个单元及第 1 个奇数单元
32	条	右侧数据符的第 2 个单元及第 1 个偶数单元
33	空	右侧数据符的第 3 个单元及第 2 个奇数单元
34	条	右侧数据符的第 4 个单元及第 2 个偶数单元
35	空	右侧数据符的第 5 个单元及第 3 个奇数单元
36	条	右侧数据符的第 6 个单元及第 3 个偶数单元
37	空	右侧数据符的第 7 个单元及第 4 个奇数单元
38	条	右侧数据符的第 8 个单元及第 4 个偶数单元
39	空	右侧数据符的第 9 个单元及第 5 个奇数单元
40	条	右侧数据符的第 10 个单元及第 5 个偶数单元
41	空	右侧数据符的第 11 个单元及第 6 个奇数单元
42	条	右侧数据符的第 12 个单元及第 6 个偶数单元
43	空	右侧数据符的第 13 个单元及第 7 个奇数单元
44	条	右侧数据符的第 14 个单元及第 7 个偶数单元
45	空	右侧保护符的内侧单元(1 个模块宽)
46	条	右侧保护符的外侧单元(1 个模块宽)

表 E.3　扩展式 RSS(6 段格式)的单元

单　元	类　型	描　述
1	空	左侧保护符的外侧单元(1 个模块宽)
2	条	左侧保护符的内侧单元(1 个模块宽)
3	空	符号字符 1 的第 1 个奇数单元(限制在 4 个模块宽或更少)
4	条	符号字符 1 的第 1 个偶数单元

表 E.3(续)

单　元	类　型	描　述
5	空	符号字符 1 的第 2 个奇数单元
6	条	符号字符 1 的第 2 个偶数单元
7	空	符号字符 1 的第 3 个奇数单元
8	条	符号字符 1 的第 3 个偶数单元
9	空	符号字符 1 的第 4 个奇数单元
10	条	符号字符 1 的第 4 个偶数单元
11	空	定位符 A1 的第 1 个单元
12	条	定位符 A1 的第 2 个单元
13	空	定位符 A1 的第 3 个单元
14	条	定位符 A1 的第 4 个单元(1 个模块宽)
15	空	定位符 A1 的第 5 个单元(1 个模块宽)
16	条	符号字符 2 的第 4 个偶数单元
17	空	符号字符 2 的第 4 个奇数单元
18	条	符号字符 2 的第 3 个偶数单元
19	空	符号字符 2 的第 3 个奇数单元
20	条	符号字符 2 的第 2 个偶数单元
21	空	符号字符 2 的第 2 个奇数单元
22	条	符号字符 2 的第 1 个偶数单元
23	空	符号字符 2 的第 1 个奇数单元(限制在 4 个模块宽或更少)
24	条	符号字符 3 的第 1 个奇数单元(限制在 4 个模块宽或更少)
25	空	符号字符 3 的第 1 个偶数单元
26	条	符号字符 3 的第 2 个奇数单元
27	空	符号字符 3 的第 2 个偶数单元
28	条	符号字符 3 的第 3 个奇数单元
29	空	符号字符 3 的第 3 个偶数单元
30	条	符号字符 3 的第 4 个奇数单元
31	空	符号字符 3 的第 4 个偶数单元
32	条	定位符 B2 的第 5 个单元(1 个模块宽)
33	空	定位符 B2 的第 4 个单元(1 个模块宽)
34	条	定位符 B2 的第 3 个单元
35	空	定位符 B2 的第 2 个单元
36	条	定位符 B2 的第 1 个单元
37	空	符号字符 4 的第 4 个偶数单元
38	条	符号字符 4 的第 4 个奇数单元
39	空	符号字符 4 的第 3 个偶数单元

表 E.3(续)

单　元	类　型	描　述
40	条	符号字符 4 的第 3 个奇数单元
41	空	符号字符 4 的第 2 个偶数单元
42	条	符号字符 4 的第 2 个奇数单元
43	空	符号字符 4 的第 1 个偶数单元
44	条	符号字符 4 的第 1 个奇数单元(限制在 4 个模块宽或更少)
45	空	符号字符 5 的第 1 个奇数单元(限制在 4 个模块宽或更少)
46	条	符号字符 5 的第 1 个偶数单元
47	空	符号字符 5 的第 2 个奇数单元
48	条	符号字符 5 的第 2 个偶数单元
49	空	符号字符 5 的第 3 个奇数单元
50	条	符号字符 5 的第 3 个偶数单元
51	空	符号字符 5 的第 4 个奇数单元
52	条	符号字符 5 的第 4 个偶数单元
53	空	定位符 B1 的第 1 个单元
54	条	定位符 B1 的第 2 个单元
55	空	定位符 B1 的第 3 个单元
56	条	定位符 B1 的第 4 个单元(1 个模块宽)
57	空	定位符 B1 的第 5 个单元(1 个模块宽)
58	条	符号字符 6 的第 4 个偶数单元
59	空	符号字符 6 的第 4 个奇数单元
60	条	符号字符 6 的第 3 个偶数单元
61	空	符号字符 6 的第 3 个奇数单元
62	条	符号字符 6 的第 2 个偶数单元
63	空	符号字符 6 的第 2 个奇数单元
64	条	符号字符 6 的第 1 个偶数单元
65	空	符号字符 6 的第 1 个奇数单元(限制在 4 个模块宽或更少)
66	条	右侧保护符的内侧单元(1 个模块宽)
67	空	右侧保护符的外侧单元(1 个模块宽)

附 录 F
（资料性附录）
编 码 示 例

F.1 RSS-14

在图 F.1 所示的复合码中，一维部分 RSS-14 中的连接标志为 1（代表连接二维部分），对贸易项目代码 24012345678905 编码。

图 F.1 RSS-14 复合码示例

一维部分 RSS-14 的单元宽度按以下步骤计算：

a) 符号值 V_{SYMBOL} 为：

V_{SYMBOL} = 10000000000000（连接标志）+ 24012345678905（贸易项目代码）去掉校验码

= 12401234567890

b) 左、右侧数据符对值 V_{LPAIR}、V_{RPAIR} 分别为：

V_{LPAIR} = 12401234567890 div 4537077 = 2733309

V_{RPAIR} = 12401234567890 mod 4537077 = 1170097

c) 4 个数据符的值为：

数据符 1 的值 $V_{D1} = V_{LPAIR}$ div 1597 = 2733309 div 1597 = 1711

数据符 2 的值 $V_{D2} = V_{LPAIR}$ mod 1597 = 2733309 mod 1597 = 842

数据符 3 的值 $V_{D3} = V_{RPAIR}$ div 1597 = 1170097 div 1597 = 732

数据符 4 的值 $V_{D4} = V_{RPAIR}$ mod 1597 = 1170097 mod 1597 = 1093

d) 这 4 个数据符的奇子集值（V_{ODD}）及偶子集值（V_{EVEN}）为：

数据符 1 结构为（16，4），而值（V_{D1}）1711 位于第 3 组中，奇/偶子集模块数为 8/8。利用下式从 V_{D1} 求出 V_{ODD} 及 V_{EVEN}：

$V_{ODD1} = (V_{D1} - 961)$ div 34 = (1711 − 961) div 34 = 750 div 34 = 22

$V_{EVEN1} = (V_{D1} - 961)$ mod 34 = (1711 − 961) mod 34 = 750 mod 34 = 2

数据符 2 结构为（15，4），而值（V_{D2}）842 位于第 2 组中，奇/偶子集模块数为 7/8。利用下式从 V_{D2} 求出 V_{EVEN} 及 V_{ODD}：

$V_{EVEN2} = (V_{D2} - 336)$ div 20 = (842 − 336) div 20 = 506 div 20 = 25

$V_{ODD2} = (V_{D2} - 336)$ mod 20 = (842 − 336) mod 20 = 506 mod 20 = 6

数据符 3 结构为（16，4），而值（V_{D3}）732 位于第 2 组中，奇/偶子集模块数为 10/6。利用下式从 V_{D3} 求出 V_{ODD} 及 V_{EVEN}：

$V_{ODD3} = (V_{D3} - 161)$ div 10 = (732 − 161) div 10 = 571 div 10 = 57

$V_{EVEN3} = (V_{D3} - 161)$ mod 10 = (732 − 161) mod 10 = 571 mod 10 = 1

数据符 4 结构为（15，4），而值（V_{D4}）1093 位于第 3 组中，奇/偶子集模块数为 9/6。利用下式从 V_{D4} 求出 V_{EVEN} 及 V_{ODD}：

$V_{EVEN4}=(V_{D4}-1036)$ div $20=(1093-1036)$ div $48=57$ div $48=1$

$V_{ODD4}=(V_{D4}-1036)$ mod $20=(1093-1036)$ mod $48=57$ mod $48=9$

e) 用附录 B 中的 RSS 子集宽度算法由子集值求得以下单元宽度序列：

奇子集 1(值 22)=3 1 1 3

偶子集 1(值 2)=1 1 3 3

因此，数据符 1 的单元宽度序列=3 1 1 1 1 3 3 3；

奇子集 2(值 6)=1 2 3 1

偶子集 2(值 25)=3 1 1 3

因此，数据符 2 的单元宽度序列=1 3 2 1 3 1 1 3(从左至右镜像反转)；

奇子集 3(值 57)=3 3 3 1

偶子集 3(值 1)=1 1 2 2

因此，数据符 3 的单元宽度序列=3 1 3 1 3 2 1 2(从左至右镜像反转)；

奇子集 4(值 9)=1 2 4 2

偶子集 4(值 1)=1 1 2 2

因此，数据符 4 的单元宽度序列=1 1 2 1 4 2 2 2 。

注：按照书写习惯，单元宽度序列都是从左至右排列，但对于有些符号字符，如上面的数据符 2 和数据符 3，字符单元的排序实际上是从右至左的。

f) 计算校验和的值：

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

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

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

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

2125

因此，校验和的值=2125 mod 79=71。

g) 从校验和的值计算两个校验符(定位符)：

71 大于或等于 8，因此 temp 的值为 71+1=72

72 大于或等于 72，因此 temp 的值为 72+1=73

左校验符的值为 73div 9=8

右校验符的值为 73 mod 9=1

左校验符(值 8)的单元序列=1 3 9 1 1

右校验符(值 1)的单元序列=3 5 5 1 1(从左至右镜像反转)。

h) 包括左侧保护符、数据符 1、左侧校验符/定位符、数据符 2(反转)、数据符 4、右侧校验符/定位符(反转)、数据符 3(反转)及右侧保护符在内的条码符号单元宽度为：

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 限定式 RSS

图 F.2 中的限定式 RSS 符号对贸易项目代码 00098765432105 进行编码。

图 F.2 限定式 RSS 符号示例

限定式 RSS 的单元宽度按下列步骤计算：

a) 求出符号值 V_{SYMBOL}：

符号值等于贸易项目代码(去掉校验码)：V_{SYMBOL}=00098765432105 去掉校验码=9876543210

b) 计算左、右侧数据符的值 V_{DLEFT}、V_{DRIGHT}：

V_{DLEFT}=9876543210 div 2013571=4904

V_{DRIGHT}=9876543210 mod 2013571=1991026

c) 计算两个数据符的奇、偶子集值 V_{ODD}、V_{EVEN}：

左侧数据符值(V_{DLEFT})为 4904，位于第 1 组中，奇/偶子集模块数为 17/9，因此左侧数据符的奇子集值(V_{ODDL})和偶子集值(V_{EVENL})分别为：

V_{ODDL}=(V_{DLEFT}−0) div 28=4904 div 28=175

V_{EVENL}=(V_{DLEFT}−0) mod 28=4904 mod 28=4

右侧数据符值(V_{DRIGHT})为 1991026，位于第 6 组中，奇/偶子集模块数为 19/7，因此右侧数据符的奇子集值(V_{ODDR})和偶子集值(V_{EVENR})分别为：

V_{ODDR}=(V_{DRIGHT}−1979845) div 1=11181 div 1=11181

V_{EVENR}=(V_{DRIGHT}−1979845) mod 1=11181 mod 1=0

d) 利用附录 B 中的 RSS 子集宽度算法由子集值得出以下单元宽度序列：

左侧数据符奇子集 (值 175)=1 1 2 2 2 4 5

左侧数据符偶子集 (值 4)=1 1 1 1 2 2 1

因此，左侧数据符单元宽度序列=1 1 1 1 2 1 2 1 2 2 4 2 5 1；

右侧数据符奇子集 (值 11181)=3 3 1 3 5 2 2

右侧数据符偶子集 (值 0)=1 1 1 1 1 1 1

因此，右侧数据符单元宽度序列=3 1 3 1 1 1 3 1 5 1 2 1 2 1。

e) 计算校验符的值：

左侧数据符单元宽度加权和=

1×1+1×3+1×9+1×27+2×81+1×65+2×17+1×51+2×64+2×14+4×42+2×37+5×22+1×66=926

右侧数据符单元宽度加权和=3×20+1×60+3×2+1×6+1×18+1×54+3×73+1×41+5×34+1×13+2×39+1×28+2×84+1×74=995

因此校验符的值=(926+995)mod 89=52

f) 用附录 B 给出的算法得到，值为 52 的校验符单元宽度序列=1 1 1 2 1 1 2 1 1 2 2 1 1 1。

g) 包括左侧保护符、左侧数据符、校验符、右侧数据符及右侧保护符在内的条码符号单元宽度为：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。

F.3 扩展式 RSS

图 F.3 中的扩展式 RSS 符号对 AI 单元数据串(10)12A 编码，本示例的数据没有包含主要项目标识，只用于示范指导。

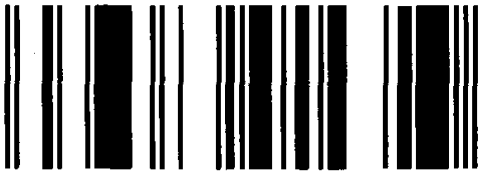


图 F.3 扩展式 RSS 符号示例

扩展式 RSS 的单元宽度按下列步骤计算：

a) 符号对 1012A 编码。

b) 二进制字段为：

连接标记＝0(无二维部分)

编码方法＝00(无应用标识符 01)

可变长度位＝00(偶数个段，组 1)

数据位：10＝0010011(数字编码)

12＝0010101(数字编码)

0000(字母数字锁定码)

A＝100000(字母数字编码)

填充＝0010000

因此按顺序链接成二进制数字串为：00000001001100101010000100000010000。

注：两位数字(如 10、12)的编码方法见 7.2.5.5.1 中的公式，大写字母(如 A)的字母数字编码方法见表 12。

c) 把二进制数字串以 12 位分成一组，本示例分为三组，值分别为：000000010011、001010100001 及 000000010000。相应的 3 个数据符值为：

数据符 1 的值(V_{D1})＝000000010011＝19

数据符 2 的值(V_{D2})＝001010100001＝673

数据符 3 的值(V_{D3})＝000000010000＝16

d) 计算这 3 个数据符的奇、偶子集值 V_{ODD}、V_{EVEN}：

数据符 1 的值(V_{D1})为 19，位于第 1 组，奇/偶子集模块数为 12/5，因此数据符 1 的奇子集值(V_{ODD1})和偶子集值(V_{EVEN1})分别为：

$V_{ODD1}=(V_{D1}-0)\ \text{div}\ 4=19\ \text{div}\ 4=4$

$V_{EVEN1}=(V_{D1}-0)\ \text{mod}\ 4=19\ \text{mod}\ 4=3$

数据符 2 的值(V_{D2})为 673，位于第 2 组，奇/偶子集模块数为 10/7，因此数据符 2 的奇子集值(V_{ODD2})和偶子集值(V_{EVEN2})分别为：

$V_{ODD2}=(V_{D2}-348)\ \text{div}\ 20=325\ \text{div}\ 20=16$

$V_{EVEN2}=(V_{D2}-348)\ \text{mod}\ 20=325\ \text{mod}\ 20=5$

数据符 3 的值(V_{D3})为 16，位于第 1 组，奇/偶子集模块数为 12/5，因此数据符 3 的奇子集值(V_{ODD3})和偶子集值(V_{EVEN3})分别为：

$V_{ODD3}=(V_{D3}-0)\ \text{div}\ 4=16\ \text{div}\ 4=4$

$V_{EVEN3}=(V_{D3}-0)\ \text{mod}\ 4=16\ \text{mod}\ 4=0$

e) 用附录 B 中给出的扩展式 RSS 子集单元宽度算法由子集值求出以下单元宽度序列：

数据符 1 的奇子集(值 4)＝1 1 7 3

数据符 1 的偶子集(值 3)＝2 1 1 1

因此，数据符 1 的单元宽度序列＝1 2 1 1 7 1 3 1(从左至右镜像反转)

数据符 2 的奇子集(值 16)＝1 5 1 3

数据符 2 的偶子集(值 5)＝1 2 2 2

因此，数据符 2 的单元宽度序列＝1 1 5 2 1 2 3 2

数据符 3 的奇子集(值 4)＝1 1 7 3

数据符 2 的偶子集(值 0) ＝1 1 1 2

因此，数据符 3 的单元宽度序列 ＝1 1 1 1 7 1 3 2(从左至右镜像反转)

注：按照书写习惯，单元宽度序列都是从左至右排列，但对于偶数序号的符号字符(奇数的数据符)，如上面的数据符 1 和数据符 3，字符单元的排序实际上是从右至左的。

f) 计算校验和：

数据符 1 的单元宽度加权和＝1×1＋2×3＋1×9＋1×27＋7×81＋1×32＋3×96＋1×77＝1007

数据符 2 的单元宽度加权和＝1×20＋1×60＋5×180＋1×118＋1×143＋2×7＋3×21＋2×63＝1562

数据符 3 的单元宽度加权和＝1×189＋1×145＋1×13＋1×39＋7×117＋1×140＋3×209＋2×205＝2382

因此，校验和＝(1007＋1562＋2382)mod 211＝98

g) 计算校验符：

校验符的值(V_C)＝211(符号字符数－4)＋校验和＝211(4－4)＋98＝98

校验符值(V_C)为 98，位于第 1 组中，奇/偶子集模块数为 12/5，因此校验符的奇子集值(V_{ODDC})和偶子集值(V_{EVENC})分别为：

$V_{ODDC}=(V_C-0)\ \text{div}\ 4=98\text{div}\ 4=24$

$V_{EVENC}=(V_C-0)\ \text{mod}\ 4=98\ \text{mod}\ 4=2$

用附录 B 给出的扩展式 RSS 子集单元宽度算法，由子集值求出以下单元宽度序列：

校验符奇子集(值 24)＝1 5 1 5

校验符偶子集(值 2)＝1 2 1 1

因此，校验符单元宽度序列＝1 1 5 2 1 1 5 1。

h) 包括左侧保护符、校验符、定位符 A1、数据符 1(反转)、数据符 2、定位符 A2、数据符 3(反转)及右侧保护符在内的条码符号单元宽度为：

1 1,1 1 5 2 1 1 5 1,1 8 4 1 1,1 3 1 7 1 1 2 1,1 1 5 2 1 2 3 2,1 1 4 8 1,2 3 1 7 1 1 1 1,1 1。

附 录 G
（资料性附录）
单元宽度译码的C语言程序

```
# include <stdio. h>
/**********************************************
*"elements ( )"和"elementsExp ( )"这两个程序
* 需输入以下值:
*     eDist [ ]=前(2 * k-2 )个边缘到相似边缘距离的标称值(Ei)
*       (字符图形总是从距相邻定位符最远的单元开始!!),以及
*     确定字符结构的 n 和 k(注意在程序中 n 和 k 分别被表示为 N 和 K)。
* 这两个程序输出以下值:
*     widths [ ]=2 * k 个推算出的单元宽度。
*
* 对于 RSS-14,调用程序:
*    对于内侧数据符,调用 elements ( * eDist, * widths,15,4);
*    对于外侧数据符,调用 elements ( * eDist, * widths,16,4)。
*
* 对于限定式 RSS 的数据符,调用 elements ( * eDist, * widths,26,7)。
*
* 对于扩展式 RSS 的数据符,调用 elementsExp ( * eDist, * widths,17,4)。
*
**********************************************/

/**********************************************
*   elements ( ) 程序确定(n,k)符号字符的各单元宽度,这种(n,k)符号字符至少有一个
*   偶数单元是一模块宽。
* (注意:偶数单元——第 2、第 4、第 6 等,具有奇数个模块。)
**********************************************/
void elements (int * eDist, int * widths,int N, int K) {

int i;
int minEven;
int barSum;

/ * 从标称的边缘到相似边缘距离尺寸推算出各单元宽度 * /
    minEven=10 ; / * 以一个相当大的最小值开始 * /
    barSum=widths [0]=1; / * 首先假设第一个条为 1 模块宽 * /
    for (i=1; i < K * 2-2; i+=2) {
        widths [i]=eDist [i-1]-widths [i-1];
        widths [i+1]=eDist [i]-widths [i];
        barSum+ =Widths [i]+ widths [i+1];
```

```
        if (widths [i]< minEven) minEven=widths [i];
}
Widths [K * 2—1]=N—barSum; / * n个模块中剩余的模块分配给最后的偶数单元 * /
if (widths [K * 2—1] < minEven) minEven=widths [K * 2—1];
if (minEven > 1) {

/ * 如果最窄偶数单元的宽度太大,重新调整,使最窄偶数单元为1模块宽 */
    for ( i=0; i< K * 2; i+=2) {
        widths [i]+=minEven—1;
        widths [i+1]—=minEven—1;
    }
  }
  return;
}

/****************************************
*  elementsExp ( ) 程序确定(n,k)符号字符的各单元宽度,这种(n,k)符号字符至少有一个
*  奇数单元是一模块宽。
* (注意:奇数单元——第1、第3、第5等,具有偶数个模块。)
****************************************/
Void elementsExp (int *eDist, int *widths,int N, int K) {

int i;
int minOdd;
int barSum;

/ * 从标称的边缘到相似边缘距离尺寸推算出各单元宽度 */
    minOdd=8;  /* 以最小值开始——假设的第一个单元宽度 */
    barSum=widths [0]=8; / * 首先假设第一个条最宽,为8模块宽 */
    for (i=1; i < K * 2—2; i+=2) {
        widths [i]=eDist [i—1]—widths [i—1];
        widths [i+1]=eDist [i]—widths [i];
        barSum+=Widths [i]+ widths [i+1];
        if (widths [i+1]< minOdd) minOdd=widths [i+1];
    }
    Widths [K * 2—1]=N—barSum; / * n个模块中剩余的模块分配给最后的偶数单元 * /
    if (minOdd > 1) {

    / * 如果最窄奇数单元的宽度太大,重新调整,使最窄奇数单元为1模块宽 */
        for ( i=0; i< K * 2; i+=2) {
            widths [i]—=minOdd—1;
            widths [i+1]+=minOdd—1;
        }
```

```
        }
        return;
}

/*****************************************
* "main ( )"示范对于 RSS-14、限定式 RSS 和扩展式 RSS 的"elements"子程序
*****************************************/
void main (void) {

int eDist14 [6]={ 5,3,3,3,4,5 };
int eDistLim [12]={ 2,3,4,3,4,4,6,6,2,2,3,4 };
int eDistExp [6]={ 2,3,3,3,3,3};
int widths [14];
int i;

/ * 对于 RSS-14 外侧数据符 * /
    elements (eDist14, widths,16,4);
    printf( "\nRSS 14 (16,4) elements: ");
    for ( i=0; i < 8; i++) printf ("%d", widths'[i] );

/ * 对于 RSS-14 内侧数据符 * /
    elements (eDist14, widths,15,4);
    printf ( "\nRSS 14 (15,4) elements: ");
    for ( i=0; i < 8; i++) printf ("%d", widths'[i] );

/ * 对于限定式 RSS * /
    elements (eDistLim, widths,26,7);
    printf ( "\nRSS Limited elements: ");
    for ( i=0; i < 14; i++) printf ("%d", widths'[i] );

/ * 对于扩展式 RSS * /
    elementsExp (eDistExp, widths,17,4);
    printf ( "\nRSS Expanded elements: ");
    for ( i=0; i < 8; i++) printf ("%d", widths'[i] );

    printf ("\n");
    return;
}
```

附 录 H
（资料性附录）
为使误读最小化的译码考虑

所有 RSS-14 及扩展式 RSS 的版本均被设计成能分段扫描，每一段均含有一个定位符及相邻的符号字符。为了使由可能的错误段而导致的误读率最小，还需要有其他的译码步骤。

H.1 按行译码

如果扫描线可以定向地通过一个符号或符号的行，那么译码器只从相邻的两段或多个段上接收数据，即按定位符—字符—字符—定位符，或者字符—定位符—字符的顺序进行接收，因此单独的定位符和相邻符号字符会被拒绝。

H.2 按段译码

一些扫描器可能要求对单个段进行译码，以提供有效的全向扫描，这些扫描器要对被译码的段进行下列有效性检查：

a) 间距检查：检查定位符 4 个被选单元的总宽度与相邻符号字符宽度之比是否超出标称值的 7%（定位符的 4 个被选单元不包括窄单元对的外侧单元。例如，具有图形{1,5,7,1,1}的定位符间距应为前 4 个单元的总宽度）；

b) 油墨扩散检查：检查定位符的条宽平均偏差（所有条的宽度与各自标称宽度之差的平均值）与相邻符号字符的条宽平均偏差之差的值是否超过 $0.3X$；

c) 表决：提供一个表决方案来选择被译码次数最多的段候选对象，对每一种段位置（RSS-14 有 4 种段位置，扩展式 RSS 有 24 种段位置），需保存一个所有段候选对象的清单；

d) 阈值：要求一个段位置上的最佳段候选对象的被译码次数至少比第二位的段候选对象的被译码次数多两次。

附 录 I
（资料性附录）
印刷注意事项

I.1 保护符注意事项

所有的 RSS 条码符号的起始位置和终止位置都有一个包含一个模块宽的条和一个模块宽的空组成的保护符。因为没有空白区，所以保护符外侧单元的颜色可能与背景相同，一个空单元可能与亮的背景颜色融合在一起，一个条单元可能与暗的背景颜色融合在一起。符号保护符的外侧单元是必要的。如图 2、图 8 和图 11 所示，左侧保护符的白色的外侧单元与白色背景无法区分开，但它还是必须存在的。

I.2 基于像素的印制

基于像素印制的打印机上使用的图形软件应能精确调整每一个条及空的宽度与打印机的像素密度相匹配。对于包括所有 RSS 系列条码在内的采用相似边之间距离进行译码的符号体系，组成每一个符号字符的像素数必须是符号或符号字符中模块数的一个固定不变的整数倍。因此，一种打印机只能打印一组特定 X 尺寸的符号。

在统一对印制过程中出现的条宽的增加（或减少）进行补偿时，必须对符号中所有的条/空采用相同的调整量，可以通过对符号中的每个条/空对及符号的最后一个条以同一方式由黑到白或由白到黑地来改变整数个像素得以实现。例如，可将沿符号中每一个条同侧边的所有像素由黑变成白，或者沿每一个条两边的像素由黑变成白，只要打印机有足够高的解析度支持完成这种符号要求的操作即可。任何一组黑至白或白至黑的像素改变都是可以接受的，只要在整个符号范围内进行一致的调整，且不改变相似边之间的距离或整个符号字符的宽度即可。不遵循这些原则将导致符号质量下降，而且经常会产生不可识读的符号。

设计用来支持各种打印机的通用打印软件，应能为用户提供调整 X 尺寸及增加或减少条宽的能力。

编程示例：

在进行数字条码设计时，可以遵循下列原则：

a) 将所需的放大倍数或 X 尺寸转换为以像素为单位的一个模块宽度，并将其四舍五入修约为最接近的整数，如果这样得到的 X 尺寸小于应用所允许的最小值，则向上修约；

b) 把统一补偿印制过程条宽增加或减少所需的补偿量换为以像素为单位的条宽调整量，并将其向上修约为整数；

c) 应用上述结果来确定符号中每一条和空的像素数。

示例：

采用解析度为 24 点（像素）/mm 的打印设备及相应的数字条码设计文件，制作一个 X 尺寸为 0.27 mm、条宽减少量为 0.06 mm 的符号。计算如下：

——设置的模块宽度（X_S）为：24 像素/mm×0.27 mm/模块＝6.5 个像素/模块，向下修约为每模块 6 像素，即 $X_S=6$（像素）；

——设置的条宽减少量（BWR_S）为：24 像素/mm×0.06 mm＝1.4 个像素，向上修约为 2 个像素，即 $BWR_S=2$（像素）；

——条单元的设置宽度＝X_S×组成条单元的模块数－BWR_S；

——空单元的设置宽度＝X_S×组成空单元的模块数＋BWR_S。

注：条宽的调整和空宽的调整总是采用相同的调整量值，而调整的方向相反。

设组成条、空单元的模块数均为2，则条单元设置宽度＝6×2－2＝10(像素)；空单元的设置宽度＝6×2＋2＝14(像素)。组成条、空单元的模块数为1～4的设置单元宽度像素数示例见表I.1。

表I.1 考虑打印机图像解析度和条宽减少量进行校正所设置的单元宽度像素数示例

组成条、空单元的模块数	单元宽度像素数	
	条	空
1	4	8
2	10	14
3	16	20
4	22	26

I.3 基于像素印刷的软件用户指南

当第一次使用一个由条码打印软件及打印设备所组成的打印系统来打印条码符号时，用户应根据GB/T 14258来检验打印的条码符号是否满足印制质量等级及X尺寸的要求。如果打印出的符号没有达到所要求的符号质量等级，则用户可能需要增加X尺寸或改变条宽的增加量或减少量。如果需要，重复这一过程直到达到所要求的符号质量等级。注意，并不是所有的打印系统都能打印出具有小X尺寸的可接受的条码符号。

I.4 过程控制注意事项

为了进行过程控制，应对平均条宽增加或减少、以及为减小这种影响而采取的校正措施是否合适等进行评估。根据GB/T 14258所测得的“可译码度”参数，会受到系统的条宽增加或减少以及相似边之间的距离变化的双重影响。

I.5 多个符号的分开

对于多个RSS符号，应把它们充分地分开，以避免它们同时出现在一个扫描器的视野内。

I.6 打印分隔符

印制一个连接标志为1的作为一维部分的RSS条码符号时应同时印制所要求的分隔符。直到连接的二维部分被印制出来，整个复合码才是有效的。

有时可能很难打印一个模块高的分隔符。比如，分隔符的一些单元可能会由于打印墨水的扩散或浸润而消失。

为对打印过程中的垂直失真进行预先补偿，可以对分隔符中的单元进行适当的条高缩减，打印后就能显示图案中的所有单元。

如果在视觉上不能辨识出分隔符，则应将图案中每一行的高度最大增加为两个模块高。这样，符号高度将相应增加，而符号字符的高度不应低于规定的最小值。

如果在视觉上仍不能辨识出分隔符，则需要增加符号的X尺寸。这样，符号的高度和宽度会相应增加。

附 录 J
（资料性附录）
RSS 系列符号特点总汇

RSS 系列条码符号的特点总汇见表 J.1。

表 J.1 RSS 系列条码符号的特点

符号类型	RSS-14	截短式 RSS-14	层排式 RSS-14	全向层排式 RSS-14	限定式 RSS	扩展式 RSS	层排扩展式 RSS
全向扫描	是	否	否	是	否	是	是
左侧/右侧保护符[a]	s;b/s;b	s;b/s;b	s;b/b;s b;s/s;b	s;b/b;s b;s/s;b	s;b/s;b	s;b/不固定	不固定
传输数据	AI(01)加 14 位 GTIN	AI(01)加 14 位 GTIN	AI(01)加 14 位 GTIN	AI(01)加 14 位 GTIN	AI(01)加 14 位 GTIN	主标识和其他 AI 单元数据串	主标识和其他 AI 单元数据串
最大数据容量[b]	16 位数	16 位数	16 位数	16 位数	16 位数	74 个数字或 41 个字母符号	74 个数字或 41 个字母符号
可编码字符集[c]	0～9	0～9	0～9	0～9	0～9	GB/T 1988 字符集的子集[h]	GB/T 1988 字符集的子集[h]
支持的包装指示符[d]	0～9	0～9	0～9	0～9	0 和 1	0～9	0～9
符号字符/定位符的数目	4/2	4/2	4/2	4/2	2/1	4/2～22/11	4/2～22/11
行数	1	1	2	2	1	1	2～11
符号结构(区域)[e]	d1/lf/d2/ d4/rf/d3	d1/lf/d2/ d4/rf/d3	d1/lf/d2/ d4/rf/d3	d1/lf/d2/ d4/rf/d3	ld/ck/rd	三元序列(符号字符/定位符/符号字符)	三元序列(符号字符/定位符/符号字符)
单元数目[f]	46	46	上面行 25，下面行 25	上面行 25，下面行 25	46	可变	可变
模块[g]数	96	96	上面行 50，下面行 50	上面行 50，下面行 50	74	可变	可变
符号最小高度	33X	13X	13X (5X+1X+ 7X)	69X (33X+3X+ 33X)	10X	34X	71X(2 行) 404X(11 行)
符号最大高度	不限	33X	13X	不限	不限	不限	不限

注：主标识指的是 AI(01)加 14 位 GTIN 。

表 J.1(续)

符号类型	RSS-14	截短式 RSS-14	层排式 RSS-14	全向层排式 RSS-14	限定式 RSS	扩展式 RSS	层排扩展式 RSS

[a] s:b/s:b 表示左侧保护符为空/条;右侧保护符为空/条。因此,符号的第一个单元为一个空。

[b] 大多数 RSS 符号只能对 AI(01)贸易项目标识进行编码。扩展式 RSS 可以支持 AI 单元数据串,如 AI(01)和 AI(8004)等。

[c] 除了扩展式 RSS,RSS 只能对 10 个数字进行编码。扩展式 RSS 可以对以下字符进行编码:0～9,A～Z,a～z,21 个特定字符/标点符号和 FCN1。

[d] 限定式 RSS 限制为 0 和 1。其他的 RSS 符号中,9 作为变量项目的包装指示。

[e] 一个区域是特定功能单元的图形,如:保护符,数据符或定位符。示例:d1 是指数据符 1;lf 是指左侧定位符;d2 是指数据符 2;d4 是指数据符 4;rf 是指右侧定位符;d3 是指数据符 3;ld 是指左侧数据符;ck 是指校验符;rd 是指右侧数据符。所有的符号都是以保护符开始和结束。

[f] 单元是条和空,每个都是特定数目的模块组合。

[g] 模块是指最窄的条或最窄的空的宽度,又称为 X 尺寸。

[h] 如表 13 定义。