ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

ПРАВИЛА КОДИРОВАНИЯ АСН.1

Часть 2

Спецификация правил уплотненного кодирования (PER)

Издание официальное

БЗ 4—2002/58

ГОССТАНДАРТ РОССИИ
Москва

Предисловие

1 РАЗРАБОТАН Федеральным государственным унитарным предприятием Государственный научно-исследовательский и конструкторско-технологический институт «ТЕСТ» Министерства Российской Федерации по связи и информатизации

ВНЕСЕН Министерством Российской Федерации по связи и информатизации

2 ПРИНЯТ И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 13 мая 2003 г. № 141-ст

3 Настоящий стандарт содержит полный аутентичный текст ИСО/МЭК 8825-2—98 «Информационная технология. Правила кодирования АСН.1. Часть 2. Спецификация правил уплотненного кодирования (PER)» с Изменением № 1 (1999 г.) и Дополнением № 1 (2000 г.)

4 ВВЕДЕН ВПЕРВЫЕ

Содержание

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

ПРАВИЛА КОДИРОВАНИЯ АСН.1

Часть 2

Спецификация правил уплотненного кодирования (PER)

Information technology. ASN.1 encoding rules.
Part 2. Specification of Packed Encoding Rules (PER)

Дата введения 2004—07—01

1 Область применения

В настоящем стандарте определен набор правил уплотненного кодирования, который может быть использован для получения синтаксиса передачи значений типов, определенных в ГОСТ Р ИСО/МЭК 8824-1. Правила уплотненного кодирования также применимы для декодирования указанного синтаксиса передачи с целью идентификации переданных значений данных.

Правила кодирования, определенные в настоящем стандарте:

- используются во время передачи;
- предназначены для использования в случаях, когда при выборе правил кодирования большое значение имеет минимизация размера представления значений;
- допускают расширение абстрактного синтаксиса путем добавления дополнительных значений, сохраняя кодирование существующих значений, для всех видов расширения, описанных в ГОСТ Р ИСО/МЭК 8824-1.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие стандарты:

ГОСТ 34.971—91 (ИСО 8822—88) Системы обработки информации. Взаимосвязь открытых систем. Спецификация услуг уровня представления для режима с установлением соединения (см. также Рекомендацию МСЭ-Т X. 216)

ГОСТ 34.972—91 (ИСО 8823—88) Системы обработки информации. Взаимосвязь открытых систем. Спецификация протокола уровня представления для режима с установлением соединения (см. также Рекомендацию МСЭ-Т X. 226)

ГОСТ Р ИСО/МЭК 7498-1—97 Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 1. Базовая модель (см. также Рекомендацию МСЭ-Т X. 200)

ГОСТ Р ИСО/МЭК 8824-1—2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН. 1). Часть 1. Спецификация основной нотации (см. также Рекомендацию МСЭ-Т X. 680)

ГОСТ Р ИСО/МЭК 8824-2—2001 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН. 1). Часть 2. Спецификация информационного объекта (см. также Рекомендацию МСЭ-Т X. 681)

ГОСТ Р ИСО/МЭК 8824-3—2002 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН. 1). Часть 3. Спецификация ограничения (см. также Рекомендацию МСЭ-Т X. 682)

ГОСТ Р ИСО/МЭК 8824-4—2003 Информационная технология. Абстрактная синтаксическая нотация версии один (АСН. 1). Часть 4. Параметризация спецификаций АСН. 1 (см. также Рекомендацию МСЭ-Т X. 683)

Издание официальное

ГОСТ Р ИСО/МЭК 8825-1—2003 Информационная технология. Правила кодирования АСН. 1. Спецификация правил основного (BER), канонического (CER) и различающего (DER) кодирований (см. также Рекомендацию МСЭ-Т X. 690)

ИСО/МЭК 10646-1—93* Информационная технология. Универсальный, многооктетный кодовый набор символов (UCS). Часть 1. Архитектура и основная многоязычная плоскость

ИСО/МЭК 646—91* Информационная технология. 7-битнокодированный набор символов ИСО для обмена информацией

3 Определения

В настоящем стандарте используют следующие термины с соответствующими определениями.

3.1 Определение услуг уровня представления

В настоящем стандарте используют следующие термины, определенные в ГОСТ 34.971:

а) **множество определенных контекстов;**

б) **идентификатор контекста уровня представления.**

3.2 Спецификация основной нотации

В настоящем стандарте используют все определения из ГОСТ Р ИСО/МЭК 8824-1.

3.3 Спецификация информационного объекта

В настоящем стандарте используют все определения из ГОСТ Р ИСО/МЭК 8824-2.

3.4 Спецификация ограничения

В настоящем стандарте используют следующие термины, определенные в ГОСТ Р ИСО/МЭК 8824-3:

а) **ограничение связи компонентов;**

б) **табличное ограничение.**

3.5 Спецификация параметризации АСН. 1

В настоящем стандарте используют следующий термин, определенный в ГОСТ Р ИСО/МЭК 8824-4, — **переменное ограничение.**

3.6 Правила основного кодирования

В настоящем стандарте используют следующие термины, определенные в ГОСТ Р ИСО/МЭК 8825-1:

а) **динамическое соответствие;**

б) **статическое соответствие;**

в) **значение данных;**

г) **кодирование (значений данных);**

д) **отправитель;**

е) **получатель.**

3.7 Дополнительные определения

В настоящем стандарте используют следующие определения:

3.7.1 **двоично-дополнительное до 2 целое кодирование:** Кодирование целого числа в выровненном по октету поле битов заданной длины или в количестве октетов, минимально необходимом для размещения этого числа, закодированного как целое двоичное дополнение до 2, обеспечивающее представление целых чисел, которые равны, больше или меньше нуля, как определено в 10.4.

Примечания

1 Значение целого двоичного дополнения числа до 2 получается путем нумерации битов в октетах содержимого, начиная с бита 1 последнего октета как бита 0 и заканчивая битом 8 первого октета. Каждому биту присваивается числовое значение 2^N, где N — номер бита в описанной выше нумерации. Значение целого двоичного дополнения числа до 2 получается суммированием присвоенных числовых значений тех битов, которые равны единице, исключая бит 8 первого октета, и последующего уменьшения этой суммы на числовое значение, присвоенное биту 8 первого октета, если тот бит равен единице.

2 Целое число в данном контексте является синонимом математического термина «целое число». Его следует отличать от целочисленного типа integer АСН. 1.

3.7.2 **значение абстрактного синтаксиса:** Значение из абстрактного синтаксиса (определенного как множество значений единственного типа АСН. 1), которое должно быть закодировано PER или которое должно быть получено декодированием PER.

---

* Международные стандарты — во ВНИИКИ Госстандарта России.

Примечание — Единственный тип ACH.1, связанный с абстрактным синтаксисом, формально идентифицируется объектом класса ABS-TRACT-SYNTAX.

3.7.3 **битовое поле:** Результат некоторой части метода кодирования, состоящий из упорядоченного набора битов, число которых не обязательно кратно восьми и которые в полном кодировании значения абстрактного синтаксиса не обязательно начинаются на границе октета.

3.7.4 **каноническое кодирование:** Полное кодирование значения абстрактного синтаксиса, полученное применением правил кодирования, которые не имеют зависящих от реализации опции; такие правила приводят к определению однозначного отображения между недвусмысленными и уникальными битовыми строками в синтаксисе передачи и значениями в абстрактном синтаксисе.

3.7.5 **составной тип:** Тип «множество», «последовательность», «множество-из», «последовательность-из», «встроенное-здп», выборочный, внешний или неограниченный тип символьных строк.

3.7.6 **составное значение:** Значение составного типа.

3.7.7 **ограниченное целое число:** Целое число, которое ограничено видимым для PER требованием, находится в диапазоне от lb до ub, где lb меньше или равно ub, и значения lb и ub являются допустимыми.

Примечание — Ограниченные целые числа встречаются в кодировании, идентифицирующем: выбранную альтернативу из выборочного типа, длину строк символов, октетов и битов, которая ограничена некоторой максимальной длиной видимым для PER образом, счетчик числа компонентов в типе «последовательность-из» или «множество-из», максимальное число которых ограничено видимым для PER образом, значение целочисленного типа, которое ограничено видимым для PER образом, и значение, которое обозначает перечисление в перечислимом типе.

3.7.8 **эффективное ограничение размера** (для ограничения типов строк): Единственное конечное ограничение размера, которое может быть применено к встроенному типу строки, результатом которого является допущение всех строк и только тех длин, которые могут иметь строки ограничиваемого типа.

Примечание — Например, следующая нотация имеет эффективное ограничение размера:

A :: = IA5String (SIZE(1..4)|SIZE(10..15)),

так как она может быть переписана с единственным ограничением размера, который относится ко всем значениям:

A :: = IA5String (SIZE(1..4|10..15)),

тогда как следующая нотация не имеет эффективного ограничения размера, так как строка может быть произвольной длины, если она не содержит никаких символов, кроме 'a', 'b' и 'c':

B :: = IA5String (SIZE(1..4)|FROM («abc»))

3.7.9 **эффективное ограничение допустимого алфавита «PermittedAlphabet»** (для ограниченных типов символьных строк): Единственное ограничение PermittedAlphabet, которое может быть применено к встроенному типу символьной строки известной кратности, результатом которого является допущение всех строк, содержащих только те символы, которые могут присутствовать в любой позиции любого значения ограниченного типа символьных строк.

Примечание — Ограничением эффективного допустимого алфавита (PermittedAlphabet) является либо весь алфавит неограниченного типа символьных строк, либо спецификация PermittedAlphabet, которая может оказаться супермножеством всех ограничений PermittedAlphabet, наложенных на тип. Например, в

Ax :: = IA5String (FROM («AB»)|FROM («CD»))

Bx :: = IA5String (SIZE (1..4)|FROM («abc»))

«Ax» имеет эффективное ограничение PermittedAlphabet, которое состоит из всего алфавита IA5String, так как нет ограничения PermittedAlphabet, применимого ко всем значениям «Ax». То же самое справедливо и для «Bx». С другой стороны, следующий пример имеет эффективное ограничение PermittedAlphabet «ABCDE», так как существует ограничение PermittedAlphabet, применимое ко всем значениям:

A :: = IA5String (FROM («AB»)|FROM («CD»)|FROM («ABCDE»))

3.7.10 **индекс перечисления:** Неотрицательное целое число, связанное с «EnumerationItem» в перечислимом типе. Индексы перечисления определяются сортировкой «EnumerationItem» в возрастающем порядке их значений перечисления и назначением индексов перечисления, начиная с нуля для первого «EnumerationItem», единицы — для второго, и далее до последнего «EnumerationItem» в отсортированном перечне.

Примечание — «EnumerationItem» в «RootEnumeracion» сортируются отдельно от элементов перечисления в «AdditionalEnumeracion».

3.7.11 **расширяемый для кодирования PER:** Свойство типа, определение которого содержит маркер расширения, влияющий на кодирование PER.

3.7.12 **список полей:** Упорядоченное множество значений битовых полей и (или) выровненных по октету битовых полей, которые создаются в результате применения настоящих правил кодирования для компонентов значения.

Примечание — В модели, применяемой в настоящем стандарте, использован термин «список полей» для указания связанного списка буферов, каждый из которых содержит кодирование «битового поля» или «выровненного по октету битового поля», его длину в битах и указатель выравнивания по октету. Каждое кодирование является кодированием значения типа ACH. 1. Указатель выравнивания по октету сообщает, должно ли кодирование быть выравнено в полном кодировании по границе октета, когда оно используется для формирования полного кодирования значения абстрактного синтаксиса, или оно должно быть добавлено непосредственно после последнего бита предыдущего кодирования. Термин «список полей» существует только для целей описания, а не предлагает метод реализации.

3.7.13 **неопределенная длина:** Кодирование, длина которого более 64K—1 или максимальная длина которого не может быть определена из нотации ACH. 1.

3.7.14 **тип фиксированной длины:** Тип, в кодировании которого крайнее значение детерминанта длины может быть определено (используя методы, установленные в настоящем стандарте) из нотации типа (после применения ограничений, видимых для PER), и это значение относится ко всем допустимым значениям типа.

3.7.15 **фиксированное значение:** Значение, которое может быть определено (используя методы, установленные в настоящем стандарте) как единственное допустимое значение (после применения ограничений, видимых для PER) управляющего типа.

3.7.16 **тип символьной строки известной кратности:** Ограниченный тип символьной строки, число октетов в кодировании которой для всех допустимых значений символьной строки равно произведению известного фиксированного числа на число символов в строке. Типами символьных строк известной кратности являются IA5String, PrintableString, VisibleString, NumericString, UniversalString и BMPString.

3.7.17 **детерминант длины:** Счетчик (битов, октетов, символов или компонентов), определяющий длину части или всего кодирования PER.

3.7.18 **обычно маленькое неотрицательное целое число:** Часть кодирования, которая представляет значения неограниченного неотрицательного целого, в которой маленькие значения встречаются с большей вероятностью, чем большие.

3.7.19 **обычно маленькая длина:** Длина кодирования, которая представляет значения неограниченной длины, маленькие значения которой встречаются с большей вероятностью, чем большие.

3.7.20 **выровненное по октету битовое поле:** Результат некоторой части метода кодирования, состоящий из упорядоченного набора битов, число которых не обязательно кратно восьми, но которые в полном кодировании значения абстрактного синтаксиса должны начинаться на границе октета.

3.7.21 **неотрицательное двоичное целое кодирование:** Кодирование ограниченного или полуограниченного целого числа либо в битовом поле заданной длины, либо в выровненном по октету битовом поле заданной длины, либо в минимальном количестве октетов, в которых поместится это целое число, закодированное как неотрицательное двоичное целое, обеспечивающее представление для целых чисел, больших или равных нулю, как установлено в 10.3.

Примечание — Значение дополнения до 2 двоичного числа получается нумерацией битов в октетах содержимого, начиная с бита 1 последнего октета как бита 0 и заканчивая битом 8 первого октета. Каждому биту присваивается числовое значение 2^N, где N — номер бита в описанной выше нумерации. Значение дополнения до 2 двоичного числа получается суммированием присвоенных числовых значений битов, равных единице.

3.7.22 **видимое для PER ограничение:** Использования нотации ограничения ACH. 1, которое влияет на кодирование значения PER.

3.7.23 **надежно передающее кодирование:** Полное кодирование значения абстрактного синтаксиса, которое может быть декодировано (включая любые вложенные кодирования) без знания множества определенных контекстов уровня представления, формирующих среду, в которой было выполнено кодирование.

3.7.24 **полуограниченное целое число:** Целое число, ограниченное видимым для PER образом

так, что оно должно быть больше или равно некоторому значению lb, но не являющееся ограниченным целым числом.

Примечание — Полуограниченные целые числа встречаются в кодировании: длин неограниченных (или некоторых ограниченных) типов строк символов, октетов и битов, счетчиков числа компонентов в неограниченных (или некоторых ограниченных) типах «последовательность-из», «множество-из» и значений целочисленного типа, ограниченных тем, что должны быть больше некоторого минимального значения.

3.7.25 **простой тип:** Тип, не являющийся составным.

3.7.26 **текстуально зависимый:** Применяется для идентификации случая, когда некоторое ссылочное имя используется для вычисления множества элементов; значение множества элементов рассматривается как зависящее от этого имени, даже если фактически сформированное окончательное арифметическое множество не зависит от фактического значения множества элементов, присвоенного ссылочному имени.

Примечание — Например следующее определение «Foo» текстуально зависит от «Bar», хотя «Bar» не влияет на набор значений «Foo» (таким образом, согласно 9.3.4, ограничение на «Foo» не является видимым для PER, так как для «Bar» устанавливается табличное ограничение, а «Foo» текстуально зависит от «Bar»).

```
MY-CLASS :: = CLASS {&name PrintableString, &age INTEGER} WITH SYNTAX {&name, &age}
MyObjectSet MY-CLASS :: = {{«Jack», 7}|{«Jill», 5}}
Bar :: = MY CLASS. &age ({MyObjectSet})
Foo : : = INTEGER (Bar|1..100)
```

3.7.27 **неограниченное целое число:** Целое число, которое не ограничивается видимым для PER образом.

Примечание — Неограниченные целые числа встречаются только при кодировании значений целого типа.

4 Сокращения

В настоящем стандарте используют следующие сокращения:
АСН. 1 — абстрактная синтаксическая нотация версии 1;
ВОС — взаимосвязь открытых систем;
здп — значение данных уровня представления;
BER — правила основного кодирования (Basic Encoding Rules) АСН. 1;
PER — правила уплотненного кодирования (Packed Encoding Rules) АСН. 1;
CER — правила канонического кодирования (Canonical Encoding Rules) АСН. 1;
DER (DER) — правила различающего кодирования (Distinguished Encoding Rules) АСН. 1;
16К — 16384;
32К — 32768;
48К — 49152;
64К — 65536.

5 Нотация

В настоящем стандарте использована нотация, определенная в ГОСТ Р ИСО/МЭК 8824-1.

6 Соглашения

6.1 В настоящем стандарте значения октетов в кодировании определены с использованием терминов «старший значащий бит» и «младший значащий бит».

Примечание — В спецификациях нижних уровней используют те же самые обозначения для определения порядка передачи битов в последовательной линии связи или для распределения битов в параллельных каналах.

6.2 В настоящем стандарте биты октета нумеруют от 8 до 1, где бит 8 — «старший значащий бит», а бит 1 — «младший значащий бит».

6.3 В настоящем стандарте термин «октет» используют вместо термина «восемь бит». Приме-

нение этого термина не подразумевает какого-либо выравнивания. Когда речь идет о выравнивании, это явно указывается в настоящем стандарте.

7 Правила кодирования, определенные в настоящем стандарте

7.1 В настоящем стандарте определены четыре правила кодирования (и связанные с ним идентификаторы объектов), которые могут быть использованы для кодирования и декодирования значений абстрактного синтаксиса, определенного как значения одного (известного) типа АСН. 1. Настоящий раздел описывает эти правила и их свойства.

7.2 Без знания типа закодированного значения невозможно определить структуру кодирования (по любому из алгоритмов правил кодирования PER). В частности, конец кодирования не может быть определен из самого кодирования без знания типа закодированного значения.

7.3 Кодирование PER всегда является надежно передающим при условии, что абстрактные значения типов EXTERNAL, EMBEDDED PDV и CHARACTER STRING не содержат идентификаторы контекста уровня представления.

7.4 Алгоритмом наиболее общего правила кодирования, из определенного в настоящем стандарте, является BASIC-PER, который в общем случае не приводит к каноническому кодированию.

7.5 Вторым алгоритмом правила кодирования, определенным в настоящем стандарте, является CANONICAL-PER, который приводит к каноническим кодированиям. Он определен как ограничение на зависящие от реализации альтернативы в кодировании BASIC-PER. CANONICAL-PER приводит к каноническим кодированиям, которые применяются, когда к абстрактным значениям необходимо применять аутентификаторы, как описано в ГОСТ Р ИСО/МЭК 8825-1.

Примечание — Любая реализация, соответствующая кодированию CANONICAL-PER, удовлетворяет и кодированию BASIC-PER. Любая реализация, соответствующая декодированию BASIC-PER, удовлетворяет декодированию CANONICAL-PER. Таким образом, кодирования, созданные по CANONICAL-PER, являются кодированиями, которые допускаются BASIC-PER.

7.6 Если тип, кодируемый по BASIC-PER или CANONICAL-PER, содержит типы EMBEDDED PDV, CHARACTER STRING или EXTERNAL, то внешнее кодирование перестает быть надежно передающим, если синтаксис передачи, используемый для всех типов EMBEDDED PDV, CHARACTER STRING и EXTERNAL, не является надежно передающим. Если тип, кодируемый по BASIC-PER или CANONICAL-PER, содержит типы EMBEDDED PDV, CHARACTER STRING или EXTERNAL, то внешнее кодирование перестает быть каноническим, если синтаксис передачи, используемый для всех типов EMBEDDED PDV, CHARACTER STRING и EXTERNAL, не является каноническим.

Примечание — Символьные синтаксисы передачи, поддерживающие все символьные абстрактные синтаксисы вида {iso standard 10646 level-1 (1). . . .}:.}, являются каноническими. Символьные синтаксисы передачи, которые поддерживают {iso standard 10646 level-2 (2). . . .} и {iso standard 10646 level-3 (3). . . .}, не всегда являются каноническими. Все вышеуказанные символьные синтаксисы передачи являются надежно передающими.

7.7 Как BASIC-PER, так и CANONICAL-PER содержат по два варианта: ALIGNED (с выравниванием) и UNALIGNED (без выравнивания). В варианте ALIGNED вставляются заполняющие биты для восстановления выравнивания по октету. В варианте UNALIGNED заполняющие биты никогда не вставляются.

7.8 Между вариантами ALIGNED и UNALIGNED взаимодействие невозможно.

7.9 Кодирование PER является саморазграниченным, только если известен тип кодируемого значения. Кодовые представления всегда кратны восьми битам. При передаче в типе EXTERNAL кодирования должны передаваться в альтернативе OCTET STRING, если только сам тип EXTERNAL не кодируется по PER, когда значение может быть закодировано как единственный тип АСН. 1 (то есть открытый тип). При передаче в протоколе уровня представления ВОС должно использоваться «полное кодирование» (как определено в ГОСТ 34.972) с выбранной альтернативой OCTET STRING.

7.10 Правила настоящего стандарта применяют к обоим алгоритмам и вариантам, если не оговорено иное.

7.11 Приложение С является справочным и содержит рекомендации, в каких комбинациях следует реализовывать PER для увеличения вероятности взаимодействия.

8 Соответствие

8.1 Динамическое соответствие устанавливается в разделе 9.

8.2 Статическое соответствие устанавливается теми стандартами, которые определяют применение настоящих правил уплотненного кодирования.

Примечание — В приложении С настоящего стандарта дано руководство по статическому соответствию относительно обеспечения двух вариантов двух алгоритмов правил кодирования. Это руководство предназначено для обеспечения взаимодействия при использовании для некоторых приложений преимуществ кодирований, не являющихся ни надежно передающими, ни каноническими.

8.3 Правила в настоящем стандарте специфицированы в терминах процедуры кодирования. Реализации не обязаны зеркально отображать указанную процедуру при условии, что для применяемого синтаксиса передачи битовая строка, созданная как полное кодирование значения абстрактного синтаксиса, идентична одной из строк, определенных в настоящем стандарте.

8.4 Требуется, чтобы реализации, выполняющие декодирование, создавали значение абстрактного синтаксиса, соответствующее любой полученной битовой строке, которая могла бы быть создана отправителем согласно правилам кодирования, идентифицированным в связанном с декодируемыми данными синтаксисе передачи.

Примечания

1 Вообще говоря, нет альтернативных кодирований для определенного в настоящем стандарте BASIC-PER. BASIC-PER становится каноническим, если в других стандартах определить надежно передающую операцию и ограничиться некоторыми из альтернатив кодирования. CANONICAL-PER представляет альтернативу для правил различающего и канонического кодирования (см. ГОСТ Р ИСО/МЭК 8825-1), когда требуется каноническое и надежно передающее кодирование.

2 Когда для обеспечения канонического кодирования используется CANONICAL-PER, рекомендуется, чтобы каждое полученное из него хеш-кодирование было связано идентификатором алгоритма, который идентифицирует CANONICAL-PER как преобразование из значения абстрактного синтаксиса в исходную битовую строку (которая затем была хеширована).

9 Подход к кодированию, используемый PER

9.1 Применение нотации типа

9.1.1 Настоящие правила кодирования используют нотацию типа АСН. 1 и могут быть применены только для кодирования значений одного типа АСН. 1, определенного с использованием этой нотации.

9.1.2 В частности, но не исключительно, они зависят от следующей информации, находящейся в модели типа и значения АСН. 1, лежащей в основе использования нотации:

а) вложенность выборочных типов внутри выборочных типов;

б) теги, установленные в компонентах типа «множество», альтернативах выборочного типа и перечислимых значениях;

в) являются ли компоненты типа «множество» или «последовательность» факультативными или нет?

г) имеют ли компоненты типа «множество» или «последовательность» значение DEFAULT или нет?

д) ограниченный диапазон значений типа, который возникает вследствие применения видимых (только) для PER ограничений;

е) является ли компонент открытым типом?

ж) присутствует ли маркер расширения?

9.2 Использование тегов для обеспечения канонического порядка

Настоящий стандарт требует, чтобы компоненты типа «множество» и выборочного типа были канонически упорядочены независимо от текстуального порядка компонентов. Канонический порядок устанавливается путем сортировки тегов компонентов, как определено в ГОСТ Р ИСО/МЭК 8824-1, 8.4.

9.3 Видимые для PER ограничения

Примечание — То обстоятельство, что некоторые ограничения АСН. 1 могут быть невидимыми для PER в отношении целей кодирования и декодирования, не влияет на использование ограничений при обработке ошибок, обнаруженных во время декодирования, и не подразумевает, что значения, нарушающие такие ограничения, являются допустимыми для передачи соответствующим отправителем.

9.3.1 Ограничения, выраженные в удобочитаемом для человека тексте или в комментарии АСН. 1, не являются видимыми для PER.

9.3.2 Переменные ограничения не являются видимыми для PER (см. ГОСТ Р ИСО/МЭК 8824-4, 10.4 и 10.5)

9.3.3 Табличные ограничения не являются видимыми для PER (см. ГОСТ Р ИСО/МЭК 8824-3).

9.3.4 Ограничения, вычисление которых текстуально зависит от табличного ограничения или ограничения связи компонентов, не являются видимыми для PER (см. ГОСТ Р ИСО/МЭК 8824-3).

9.3.5 Ограничения связи компонентов не являются видимыми для PER (см. ГОСТ Р ИСО/МЭК 8824-3).

9.3.6 Ограничения на ограниченные типы символьных строк, не являющиеся (см. ГОСТ Р ИСО/МЭК 8824-1, раздел 36) типами символьных строк известной кратности, не являются видимыми для PER (см. 3.7.16).

9.3.7 С учетом вышесказанного, все ограничения размера являются видимыми для PER.

9.3.8 Эффективное ограничение размера для ограничиваемого типа является единственным и таким, что размер допустим только в том случае, если существует некоторое значение ограниченного типа, которое имеет этот (допустимый) размер. Если ограниченный тип имеет значения размера, которые не удовлетворяют ограничению, то эффективного ограничения размера не существует.

9.3.9 Ограничения допустимого алфавита PermittedAlphabet в типах символьных строк известной кратности являются видимыми для PER.

9.3.10 Эффективное ограничение PermittedAlphabet для ограничиваемого типа является единственным и таким, что символ допустим только в том случае, если имеется некоторое значение ограниченного типа, которое содержит этот символ. Если все символы ограничиваемого типа могут быть представлены в некотором значении ограниченного типа, то эффективное ограничение PermittedAlphabet является набором символов, определенным для неограниченного типа.

Примечания

1 В определении ограниченного типа кратные, видимые для PER ограничения, могут быть применены либо прямо, либо с использованием конструкции «ContainedSubtype».

2 Результат совместного действия ограничений, которые по отдельности являются видимыми для PER, см. в приложении В.

9.3.11 Внутреннее ограничение типа, примененное к вещественному типу, является видимым для PER.

9.3.12 Внутреннее ограничение типа, примененное к неограниченному типу символьных строк или типу «встроенное-здп», является видимым для PER только в том случае, когда оно используется для ограничения значения компонента «syntaxes» до единственного значения или для ограничения «identification» до альтернативы «fixed» (см. разделы 24 и 27).

9.3.13 Ограничения на полезные типы (по ГОСТ Р ИСО/МЭК 8824-1) не являются видимыми для PER.

9.3.14 С учетом вышесказанного, все другие ограничения являются видимыми для PER только в том случае, если они применяются к целочисленному типу или, исключая ограничение до единственного значения, к типам символьных строк известной кратности.

9.3.15 Если видимое для PER ограничение имеет внешнюю альтернативу ElementSetSpec, которая создает расширяемое множество значений (в соответствии с ГОСТ Р ИСО/МЭК 8824-1, раздел 46), то полученный тип является расширяемым для кодирований PER (пока он не ограничен далее, см. ГОСТ Р ИСО/МЭК 8824-1, 47.5); в противном случае тип не является расширяемым для кодирований PER.

Примечания

1 Если маркер расширения присутствует в ограничении ConstraintSpec, которое не является видимым для PER, а другого маркера расширения в ограничении нет, то тип кодируется PER так, как если бы у него отсутствовал маркер расширения.

2 Если имеется несколько спецификаций SizeConstraint, применяемых к типу, и одна из них является расширяемой, то тип кодируется PER так, как если бы маркеры расширения присутствовали во всех спецификациях SizeConstraint.

9.3.16 Тип является расширяемым для кодирований PER, если выполнено любое из следующих условий:

а) он получен из типа ENUMERATED (образованием подтипа, ссылкой на тип или тегированием) и имеется маркер расширения в продукции «Enumerations», или

б) он получен из типа SEQUENCE (образованием подтипа, типом, ссылкой на тип или тегированием) и имеется маркер расширения в продукциях «ComponentTypeLists» или «SequenceType», или

в) он получен из типа SET (образованием подтипа, ссылкой на тип или тегированием) и имеется маркер расширения в продукциях «ComponentTypeLists» или «SetType», или

г) он получен из типа CHOICE (образованием подтипа, ссылкой на тип или тегированием) и имеется маркер расширения в продукции «AlternativeTypeLists».

9.4 Модель типа и значения, используемая для кодирования

9.4.1 Тип ACH. 1 является либо простым, либо построен с использованием других типов. Нотация допускает использование ссылок на тип и тегирование типов. Для настоящих правил кодирования использование ссылок на тип и тегирование не влияет на кодирование и не видно в модели, за исключением требований, установленных в 9.2. Нотация также допускает применение ограничений и спецификаций ошибок. Видимые для PER ограничения представлены в модели как ограничения значений типа. Другие ограничения и спецификации ошибок не влияют на кодирование и не видны в модели типа и значения PER.

9.4.2 Значение, которое должно быть закодировано, может рассматриваться либо как простое, либо как составное, построенное с помощью метода структурирования из компонентов, которые являются простыми или составными значениями, подобно структуре определения типа ACH. 1.

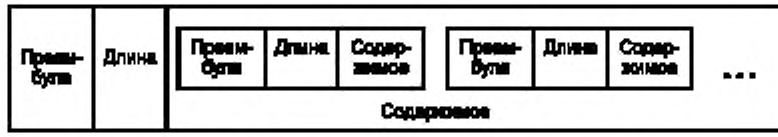
9.5 Структура кода

9.5.1 Настоящие правила кодирования специфицируют:

а) кодирование простого значения в список полей, и

б) кодирование составного значения в список полей, используя списки полей, порожденные применением настоящих правил кодирования к компонентам составного значения, и

в) преобразование списка полей самого внешнего значения в полное кодирование значения абстрактного синтаксиса (см. 10.1).



Примечание — Преамбула, длина и содержимое являются «полями», которые, сцепленные вместе, образуют «список полей». Список полей составного типа, отличного от выборочного, может состоять из полей нескольких значений, сцепленных вместе. Либо преамбула, либо длина и (или) содержимое любого значения могут быть опущены.

Рисунок 1 — Кодирование составного значения в список полей

9.5.2 Кодирование значения компонента данных либо:

а) состоит из трех частей, показанных на рисунке 1, которые появляются в следующем порядке:

1) преамбула (см. разделы 18, 20 и 22);

2) детерминант длины (см. 10.9);

3) содержимое, либо

б) (когда содержимое большое) состоит из произвольного числа частей, как показано на рисунке 2, первая из которых является преамбулой (см. разделы 18, 20 и 22), а последующие являются парами выровненных по октету битовых полей: первое поле — детерминант длины для фрагмента содержимого, второе — фрагмент содержимого; последняя пара полей идентифицируется частью детерминанта длины, как определено в 10.9.

Рисунок 2 — Кодирование значения длинных данных

9.5.3 Каждая часть, упомянутая в 9.5.2, порождает:

а) либо вырожденное поле (null), либо

в) битовое поле, либо

г) выровненное по октету битовое поле, либо

д) список полей, который может содержать либо битовые поля, либо выровненные по октету битовые поля, либо те и другие.

9.6 Кодируемые типы

9.6.1 Дальнейшие разделы специфицируют кодирование следующих типов в списках полей: булевского, целочисленного, перечислимого, действительного, «битовая строка», «строка октетов», вырожденного, «последовательность», «последовательность-из», «множество», «множество-из», выборочного, открытого, «идентификатор объекта», «относительный идентификатор объекта», «встроенное-здп», внешнего, ограниченных и неограниченных типов символьных строк.

9.6.2 Тип ANY, определенный в ГОСТ Р ИСО/МЭК 8824—93, должен кодироваться как открытый тип.

9.6.3 Селективный тип должен кодироваться как выбранный тип.

9.6.4 Кодирование тегированных типов не включено в настоящий стандарт, так как, кроме исключений в 9.2, тегирование невидимо в модели типа и значения, используемой для данных правил кодирования. Тегированные типы кодируются в соответствии с кодированием типа, который был тегирован.

9.6.5 Следующие «полезные типы» раздела 40 ГОСТ Р ИСО/МЭК 8824-1 должны кодироваться так, как если бы они были заменены своими определениями по ГОСТ Р ИСО/МЭК 8824-1:

- обобщенное время;
- всемирное время;
- описатель объекта.

Ограничения на полезные типы не являются видимыми для PER.

Ограничения на кодирование типов обобщенного и всемирного времени, установленные в CER и DER (ГОСТ Р ИСО/МЭК 8825-1, 11.7 и 11.8), должны применяться и здесь.

10 Процедуры кодирования

10.1 Создание полного кодирования

10.1.1 Список полей, полученный в результате применения настоящего стандарта к самому внешнему значению, должен использоваться для создания полного кодирования значения абстрактного синтаксиса следующим образом: каждое поле в списке рассматривают поочередно и присоединяют (с предшествующими дополнительными нулевыми битами заполнения) к концу битовой строки, которая должна образовать полное кодирование значения абстрактного синтаксиса.

10.1.2 В варианте UNALIGNED настоящих правил кодирования все поля должны быть сцеплены без заполнения. Если результат кодирования самого внешнего значения является пустой битовой строкой, то битовая строка должна быть заменена единственным октетом со всеми битами, равными 0. Если результат является непустой битовой строкой и не кратен восьми битам, то к строке должны быть добавлены нулевые биты (не более семи) для кратности восьми.

10.1.3 В варианте ALIGNED настоящих правил кодирования все битовые поля в списке полей должны быть сцеплены без заполнения, а все выровненные по октету битовые поля должны быть сцеплены после присоединения (от нуля до семи) нулевых битов для получения длины кодирования, кратной восьми битам. Если результат кодирования самого внешнего значения является пустой битовой строкой, то битовая строка должна быть заменена единственным октетом со всеми битами, равными 0. Если результат является непустой битовой строкой и не кратен восьми битам, то к строке должны быть добавлены нулевые биты (не более семи) для кратности восьми.

Примечание — Кодирование самого внешнего значения является пустой битовой строкой, если, например, значение абстрактного синтаксиса является вырожденным типом или ограниченным до единственного значения целочисленным типом.

10.1.4 Результирующая битовая строка является полным кодированием значения абстрактного синтаксиса.

10.2 Поля открытого типа

10.2.1 Для кодирования полей открытого типа значение фактического типа, занимающее поле, должно быть закодировано в списке полей, который затем должен быть преобразован в полное

кодирование значения абстрактного синтаксиса, как определено в 10.1, для создания строки октетов длиной *n*.

10.2.2 Затем список полей для значения, в которое должен быть вставлен открытый тип, должен быть добавлен (как определено в 10.9) к неограниченной длине *n* (в октетах) и соответствующему выровненному по октету полю битов, содержащему результат 10.2.1.

Примечание — Когда число октетов в кодировании открытого типа большое, должны использоваться процедуры фрагментации 10.9, и кодирование открытого типа разбивается без учета положения границы фрагмента в кодировании типа, занимающего поле открытого типа.

10.3 Кодирование как неотрицательное двоичное целое

Примечание — В настоящем подразделе дано точное определение термина «неотрицательное двоичное целое кодирование», которое размещает единственное целое в поле, содержащем фиксированное число битов, фиксированное число октетов или минимальное количество октетов, необходимых для его размещения.

10.3.1 Последующие разделы ссылаются на генерацию неотрицательного двоичного целого кодирования неотрицательного целого числа в битовом поле заданной длины, единственном октете, паре октетов или минимальном для значения количестве октетов. Настоящий подраздел специфицирует кодирование, которое должно при этом использоваться.

10.3.2 Головной бит поля определяется как старший значащий бит первого октета, а завершающий бит поля — как младший значащий бит последнего октета.

10.3.3 Только для последующего определения биты должны быть перенумерованы, начиная от нуля для завершающего бита с шагом единица до головного бита поля.

10.3.4 При неотрицательном двоичном целом кодировании значение целого числа, представленного этим кодированием, должно быть суммой значений, определенных каждым битом. Бит, равный «0», имеет нулевое значение. Бит с номером *n*, равный «1», имеет значение 2^n.

10.3.5 Кодирование, сумма которого (определенная выше) равна кодируемому значению, является кодированием этого значения.

Примечание — Если размер поля кодирования фиксирован (битовое поле заданной длины, единственный октет или пара октетов), то существует единственное кодирование, сумма которого равна кодируемому значению.

10.3.6 Неотрицательное двоичное целое кодирование целого числа в минимальном количестве октетов (когда заранее не определено количество октетов, используемых для кодирования) имеет поле, кратное восьми битам, и удовлетворяет условию, что не все головные восемь битов поля равны нулю, за исключением случая, когда поле имеет длину ровно восемь битов.

Примечание — Это является необходимым и достаточным условием для создания единственного кодирования.

10.4 Кодирование как двоично-дополнительное до 2 целое

Примечание — В настоящем подразделе дано точное определение термина «двоично-дополнительное до 2 целое кодирование», которое располагает единственное целое в поле, содержащем минимальное количество октетов, необходимых для его размещения. На эти процедуры ссылаются последующие спецификации кодирования.

10.4.1 Последующие разделы ссылаются на генерацию двоично-дополнительного до 2 целого кодирования целого числа (которое может быть отрицательным, нулем или положительным) в минимальном для значения количестве октетов. Настоящий подраздел точно специфицирует кодирование, которое должно применяться при подобных ссылках.

10.4.2 Головной бит поля определяется как старший значащий бит первого октета, а завершающий бит поля — как младший значащий бит последнего октета.

10.4.3 Только для последующего определения биты должны быть перенумерованы, начиная от нуля для завершающего бита с шагом единица до головного бита поля.

10.4.4 При двоично-дополнительном до 2 целом кодировании значение целого числа, представленного этим кодированием, должно быть суммой значений, определенных каждым битом. Бит, равный «0», имеет нулевое значение. Бит с номером *n*, равный «1», имеет значение 2^n, если он не является головным; в последнем случае он имеет (отрицательное) значение -2^n.

10.4.5 Кодирование, сумма которого (определенная выше) равна кодируемому значению, является кодированием этого значения.

10.4.6 Двоично-дополнительное до 2 целое кодирование целого числа в минимальном коли-

честве октетов имеет ширину поля, кратную восьми битам, и удовлетворяет условию, что головные восемь битов поля не равны нулю или единице одновременно.

Примечание — Это является необходимым и достаточным условием для создания единственного кодирования.

10.5 Кодирование ограниченного целого числа

Примечание — На настоящий подраздел ссылаются другие разделы, а сам он ссылается на предыдущие подразделы при создании неотрицательного двоичного целого или двоично-дополнительного до 2 целого кодирования. Для варианта UNALIGNED значение всегда кодируется в минимальном количестве битов, необходимом для представления диапазона (определенного в 10.5.3). Остальная часть настоящего примечания относится к варианту ALIGNED. Когда диапазон меньше или равен 255, значение кодируется в битовом поле минимального для диапазона размера. Когда диапазон равен 256, значение кодируется в единственном октете выровненного по октету битового поля. Когда диапазон от 257 до 64K, значение кодируется в два октета выровненного по октету битового поля. Когда диапазон больше 64K, он игнорируется, и значение кодируется в выровненное по октету битовое поле, которое содержит минимальное для значения количество октетов. В последнем случае дальнейшие процедуры (см. 10.9) кодируют и поле длины (обычно один октет) для указания длины кодирования. В других случаях длина кодирования не зависит от кодируемого значения и не кодируется явно.

10.5.1 Настоящий подраздел специфицирует отображение из ограниченных целых чисел либо в битовое поле, либо в выровненное по октету битовое поле, которое используется в последующих разделах настоящего стандарта.

10.5.2 Процедуры данного подраздела вызываются, если только ограниченное целое число, которое должно быть закодировано, доступно, а значения нижней границы, lb, и верхней границы, ub, были определены из нотации типа (после применения видимых для PER ограничений).

Примечание — Нижняя граница не может быть определена, если MIN является бесконечным числом, а верхняя граница не может быть определена, если MAX является бесконечным числом. Например ни верхняя, ни нижняя границы не могут быть определены для INTEGER (MIN..MAX).

10.5.3 Определим диапазон «range» как целое значение (ub — lb + 1); пусть значение, которое должно быть закодировано, есть *n*.

10.5.4 Если «range» имеет значение 1, то результат кодирования должен быть пустым битовым полем (без битов).

10.5.5 Существует пять других случаев (приводящих к разным кодированиям), в одном из которых применяется вариант UNALIGNED, а в четырех других — вариант ALIGNED.

10.5.6 В случае варианта UNALIGNED значение (*n* — lb) должно быть закодировано как неотрицательное двоичное целое в битовом поле, как определено в 10.3, с минимальным числом битов, необходимым для представления диапазона.

Примечание — Если «range» удовлетворяет неравенству $2^m < \text{«range»} \le 2^{m+1}$, то число битов равно $m + 1$.

10.5.7 В случае варианта ALIGNED кодирование зависит от того, что:

а) «range» меньше или равен 255 (случай битового поля);

б) «range» равен 256 (случай одного октета);

в) «range» больше 256, но меньше или равен 64K (случай двух октетов);

г) «range» больше 64K (случай неопределенной длины).

10.5.7.1 (Случай битового поля). Если диапазон «range» меньше или равен 255, то применение настоящего раздела требует сгенерировать битовое поле с числом битов, указанным ниже в таблице и содержащим значение (*n* — lb) в виде неотрицательного двоичного целого кодирования в битовом поле, как определено в 10.3.

Диапазон «range»	Размер битового поля (в битах)
2	1
3, 4	2
5—8	3
9—16	4
17—32	5
33—64	6
65—128	7
129—255	8

10.5.7.2 (Случай одного октета). Если диапазон «range» имеет значение 256, то значение (*n* — lb) должно быть закодировано в одном октете выровненного по октету битового поля в виде неотрицательного двоичного целого кодирования, как определено в 10.3.

10.5.7.3 (Случай двух октетов). Если диапазон «range» имеет значение большее или равное 257 и меньшее или равное 64K, то значение (*n* — lb) должно быть закодировано в двух октетах выровненного по октету битового поля в виде неотрицательного двоичного целого кодирования, как определено в 10.3.

10.5.7.4 (Случай неопределенной длины). В противном случае значение (*n* — lb) должно быть закодировано в неотрицательном двоичном целом виде в выровненном по октету битовом поле с минимальным количеством октетов, как определено в 10.3, а количество занятых под кодирование октетов len используется другими разделами, ссылающимися на данный раздел, для спецификации кодирования длины.

10.6 Кодирование обычно маленького неотрицательного целого числа

Примечание — Эта процедура используется при кодировании неотрицательного целого числа, которое предполагается маленьким, но размер которого потенциально не ограничивается из-за наличия маркера расширения. Примером является индекс выбора.

10.6.1 Если неотрицательное целое число $n \le 63$, то к списку полей должно быть добавлено однобитовое поле с битом, равным 0, а *n* должно быть закодировано неотрицательным двоичным целым кодированием в 6-битовом поле.

10.6.2 Если $n \ge 64$, то к списку полей должно быть добавлено однобитовое поле с битом, равным 1. Затем значение *n* должно быть закодировано как полуограниченное целое число с lb = 0, и должны быть проведены процедуры 10.9 для его добавления к списку полей с предшествующим детерминантом длины.

10.7 Кодирование полуограниченного целого числа

Примечание — Эта процедура используется, когда может быть идентифицирована нижняя граница, но не верхняя. Процедура кодирования располагает смещение от нижней границы в минимальном количестве октетов в неотрицательном двоичном целом виде и требует явного кодирования длины (обычно один октет), как определено в последующих процедурах.

10.7.1 Данный подраздел специфицирует отображение из полуограниченных целых чисел в выровненное по октету битовое поле, которое используется в последующих разделах настоящего стандарта.

10.7.2 Процедуры настоящего подраздела вызываются, если только полуограниченное целое число (обозначенное *n*), которое должно быть закодировано, доступно, а значение lb было определено из нотации типа (после применения видимых для PER ограничений).

Примечание — Нижняя граница не может быть определена, если вычисление MIN приводит к бесконечному числу. Например нижняя граница не может быть определена для INTEGER (MIN..MAX).

10.7.3 Процедуры настоящего подраздела всегда приводят к случаю неопределенной длины.

10.7.4 (Случай неопределенной длины). Значение (*n* — lb) должно быть закодировано как неотрицательное двоичное целое в выровненном по октету поле битов с минимальным количеством октетов, как определено в 10.3, а количество занятых под кодирование октетов len используется в других разделах, которые ссылаются на настоящий подраздел для спецификации кодирования длины.

10.8 Кодирование неограниченного целого числа

Примечание — Этот случай возникает только при кодировании значения целого типа без нижней границы. Процедура кодирует значение как двоично-дополнительное до 2 целое в минимальном для размещения кода количестве октетов и требует явного кодирования длины (обычно один октет), как определено в последующих процедурах.

10.8.1 Настоящий подраздел специфицирует отображение неограниченного целого числа (обозначенного *n*) в выровненное по октету поле битов, которое используется в последующих разделах настоящего стандарта.

10.8.2 Процедуры настоящего подраздела всегда приводят к случаю неопределенной длины.

10.8.3 (Случай неопределенной длины.) Значение *n* должно быть закодировано как двоично-дополнительное до 2 целое в выровненное по октету поле битов с минимальным количеством

октетов, как определено в 10.3, количество занятых под кодирование октетов len используется в других разделах, которые ссылаются на настоящий подраздел для спецификации кодирования длины.

10.9 Общие правила кодирования детерминанта длины

Примечания

1 Процедуры настоящего подраздела вызываются для некоторой части кодирования, когда необходима явная длина поля независимо от того, ограничен сверху счетчик длины (видимыми для PER ограничениями) или нет. Часть кодирования, к которой относится длина, может быть битовой строкой (со счетчиком длины в битах), строкой октетов (со счетчиком длины в октетах), символьной строкой известной кратности (со счетчиком длины в символах) или списком полей (со счетчиком длины в компонентах последовательности-из или множества-из).

2 В случае варианта ALIGNED, если счетчик длины ограничен верхней границей, которая меньше 64K, для длины используется кодирование ограниченного целого числа. Для достаточно маленьких диапазонов результатом является битовое поле, в противном случае неограниченная длина (обозначенная *n*) кодируется в выровненном по октету поле битов одним из трех способов (в порядке увеличения размера):

а) (*n* меньше 128) единственный октет, содержащий *n* с равным нулю битом 8;

б) (*n* меньше 16K) два октета, содержащие *n* с битом 8 первого октета, равным 1, и битом 7, равным нулю;

в) (большое *n*) единственный октет, содержащий счетчик *m* с 8 и 7 битами, равными 1. Счетчик *m* равен от 1 до 4, а длина указывает, какой фрагмент данных следует дальше (*m*, умноженное на 16K элементов). За определенным значением *m* фрагментом следует другая длина кодирования для оставшихся данных.

3 В случае варианта UNALIGNED, если счетчик длины ограничен верхней границей, которая меньше 64K, для кодирования длины в минимальном количестве битов, необходимых для представления диапазона, используется кодирование ограниченного целого числа. В противном случае неограниченная длина (обозначенная *n*) кодируется в битовое поле способом, описанным в примечании 2.

10.9.1 Процедуры настоящего подраздела не вызываются, если в соответствии со спецификациями последующих разделов значение детерминанта длины *n* фиксировано определением типа (с видимыми для PER ограничениями), и это значение меньше 64K.

10.9.2 Процедуры настоящего подраздела вызываются в дополнение к списку полей, содержащему поля или списки полей, с предшествующим детерминантом длины *n*, который определяет:

а) длину в октетах ассоциированного поля (единицами являются октеты), либо

б) длину в битах ассоциированного поля (единицами являются биты), либо

в) число компонентов, закодированных в ассоциированном списке полей (единицами являются компоненты множества-из или последовательности-из), либо

г) число символов в значении ассоциированного типа символьной строки известной кратности (единицами являются символы).

10.9.3 (Вариант ALIGNED) Процедуры для варианта ALIGNED специфицированы в 10.9.3.1—10.9.3.8.4. (Процедуры для варианта UNALIGNED специфицированы в 10.9.4.)

10.9.3.1 В результате анализа определения типа (установленного в последующих разделах) определяется, будет ли детерминант длины (целое число *n*):

а) обычно маленькой длиной с нижней границей lb = 1; либо

б) ограниченным целым числом с нижней границей lb ≥ 0 и верхней границей ub < 64K, либо

в) полуограниченным целым числом с нижней границей lb ≥ 0 или ограниченным целым числом с нижней границей lb ≥ 0 и верхней границей ub ≥ 64K.

10.9.3.2 Подраздел, вызывающий процедуры настоящего подраздела, должен определить значения для нижней lb (равное нулю, если длина неограничена) и верхней ub границ длины. ub не устанавливается, если нет верхней границы, определимой по видимым для PER ограничениям.

10.9.3.3 Когда детерминант длины является ограниченным целым числом с ub < 64K, список полей будет добавлен к кодированию этого ограниченного целого числа для детерминанта длины, как определено в 10.5. Если *n* не равно нулю, то за ним должно следовать ассоциированное поле или список полей, и процедура завершается. Если *n* равно нулю, то к списку полей ничего не добавляется, и процедура завершается.

Примечания

1 Например:

```
A :: = Foo (SIZE (3..6)) -- Длина кодируется в 2-битовом поле
B :: = Foo (SIZE (40000..40254)) -- Длина кодируется в 8-битовом поле
C :: = Foo (SIZE (0..32000)) -- Длина кодируется в 2 октета, выровненного по октету битового поля
D :: = Foo (SIZE (64000)) -- Длина не кодируется
```

2 Результатом отсутствия дополнения в случае *n*, равного нулю, является отсутствие заполнения до границы октета, когда эти процедуры вызываются для дополнения выровненного по октету битового поля нулевой длины, если иное не требуется 10.5.

10.9.3.4 Когда детерминант длины является обычно маленькой длиной и $n \le 64$, к списку полей должно добавляться однобитовое поле с битом, установленным в 0, а значение $n - 1$ должно быть закодировано как неотрицательное двоичное целое в 6-битовом поле. За этим должно следовать ассоциированное поле, завершающее эти процедуры. Если *n* больше 64, то к списку полей должно добавляться однобитовое поле с битом, установленным в 1, за которым следует кодирование *n* как неограниченная детерминанта длины, с последующим ассоциированным полем, согласно процедурам 10.9.3.5—10.9.3.8.4.

Примечание — Обычно маленькие длины используются только для указания длины битовых отображений, которые являются префиксами значений расширяющих дополнений типов «множество» или «последовательность».

10.9.3.5 В противном случае (неограниченная или большая ub) *n* кодируется и добавляется к списку полей с последующими ассоциированными полями, как указано ниже.

Примечание — Нижняя граница lb не влияет на кодирования длины, определенные в 10.9.3.6—10.9.3.8.4.

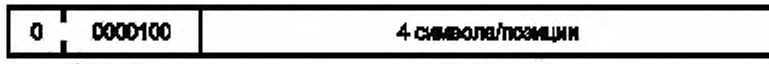
10.9.3.6 Если $n \le 127$, то *n* должно закодироваться как неотрицательное двоичное целое (используя процедуры 10.3) в битах с 7 (старший значащий) до 1 (младший значащий) единственного октета, а бит 8 должен быть установлен в нуль. Результат добавляется к списку полей как выровненное по октету битовое поле со следующим ассоциированным полем или списком полей, что завершает эти процедуры.

Примечание — Например, если следующее значение A имеет длину 4 символа, а значение B имеет длину 4 позиции:

A :: = IA5String

B :: = Foo (SIZE (4..123456))

то оба значения кодируются с длиной, занимающей один октет, и с первым битом, установленным в 0, для указания, что значение длины меньше или равно 127:



10.9.3.7 Если $n > 127$, но меньше 16K, тогда *n* должно быть закодировано как неотрицательное двоичное целое (используя процедуры 10.3) с бита 6 первого октета (старший значащий) до бита 1 второго октета (младший значащий) выровненного по октету битового поля длиной 2 октета с битом 8 первого октета, установленным в 1, и битом 7 первого октета, установленным в нуль. Результат должен быть добавлен к списку полей с последующим ассоциированным полем или списком полей, что завершает эти процедуры.

Примечание — Если в примере 10.9.3.6 значение A имеет длину 130 символов, а значение B имеет длину 130 позиций, то оба значения кодируются с компонентом длины, занимающим 2 октета, с первыми двумя битами, установленными в 10 для указания значения длины, большей 127, но меньшей 16K:

10.9.3.8 Если $n \ge 16K$, то к списку полей должен быть добавлен единственный октет выровненного по октету битового поля с битами 8 и 7, установленными в 1, и с битами с 6 по 1, кодирующими значения 1, 2, 3 или 4 как неотрицательное двоичное целое (используя процедуры 10.8). За этим единственным октетом должна следовать часть ассоциированного поля или списка полей, как установлено ниже.

Примечание — Значение битов с 6 по 1 ограничено значениями 1—4 (вместо теоретических пределов 0—63) для того, чтобы ограничить число элементов, о которых должно быть известно реализации, более управляемым количеством (64K вместо 1024K).

10.9.3.8.1 Для получения счетчика (обозначенного *m*) значение битов с 6 по 1 (1—4) должно быть умножено на 16K. Целое число в битах с 6 по 1 должно выбираться максимально возможным, но таким, чтобы содержимое ассоциированного поля или списка полей было больше или равно *m* октетам, битам, компонентам или символам соответственно.

Примечания

1 Нефрагментированный вид имеет дело с длинами до 16K. Следовательно, фрагментация обеспечивает длины до 64K с детализацией до 16K.

2 Если в примере 10.9.3.6 значение B имеет длину 144K + 1 (то есть 64K + 64K + 16K + 1) элементов, то значение фрагментируется с первыми двумя битами первых трех фрагментов, равными 11, для указания того, что далее следует от одного до четырех блоков по 16K позиций каждый и что еще один компонент длины будет следовать за последним блоком каждого фрагмента:

10.9.3.8.2 Эта часть содержимого, специфицированная *m*, должна затем добавляться к списку полей как:

а) одно выровненное по октету битовое поле из *m* октетов, содержащее первые *m* октетов ассоциированного поля — при измерении длины в октетах, либо

б) одно выровненное по октету битовое поле из *m* битов, содержащее первые *m* битов ассоциированного поля — при измерении длины в битах, либо

в) список полей, кодирующих первые *m* компонентами в ассоциированном списке полей — при измерении длины в компонентах типов «множество-из» или «последовательность-из», либо

г) одно выровненное по октету битовое поле из *m* символов, содержащее первые *m* символов ассоциированного поля — при измерении длины в символах.

10.9.3.8.3 Затем следует повторно использовать процедуры 10.9 для добавления к списку полей оставшейся части ассоциированного поля или списка полей с длиной, которая является полуограниченным целым числом, равным ($n - m$), с нулевой нижней границей.

Примечание — Если последний фрагмент, содержащий часть кодируемого значения, имеет длину, точно кратную 16K, то за ним следует завершающий фрагмент, который состоит из единственного октета компонента длины, равного 0.

10.9.3.8.4 Добавление к списку полей только части ассоциированного (ых) поля (ей) с повторным применением настоящих процедур названо процедурой фрагментации.

10.9.4 (Вариант UNALIGNED) Процедуры для варианта UNALIGNED специфицированы в 10.9.4.1—10.9.4.2 (процедуры для варианта ALIGNED специфицированы в 10.9.3):

10.9.4.1 Если детерминант длины *n*, который должен быть закодирован, является ограниченным целым числом с диапазоном «range» (ub — lb + 1) меньшим 64K, то *n* должен быть закодирован как неотрицательное двоичное целое (как определено в 10.3), используя минимальное количество битов, необходимое для кодирования «range». Если *n* не равен нулю, то за ним должно следовать ассоциированное поле или список полей, и настоящие процедуры завершаются. Если *n* равно нулю, то не должно быть дальнейшего дополнения к списку полей, и настоящие процедуры завершаются.

Примечание — Если «range» удовлетворяет неравенству $2^m < \text{«range»} \le 2^{m+1}$, то число битов в детерминанте длины равно $m + 1$.

10.9.4.2 Если детерминант длины *n*, который должен быть закодирован, является ограниченным целым числом, большим или равным 64K, или полуограниченным целым числом, тогда *n* должен быть закодирован, как определено в 10.9.3.4—10.9.3.8.4.

11 Кодирование булевского типа

11.1 Значение булевского типа должно быть закодировано как битовое поле, состоящее из единственного бита.

11.2 Бит должен быть равен 1 для TRUE и 0 — для FALSE.

11.3 Битовое поле должно быть добавлено к списку полей без детерминанта длины.

12 Кодирование целочисленного типа

Примечания

1 (Вариант ALIGNED) Диапазоны, которые допускают кодирование всех значений в одном октете или менее, попадают в битовое поле минимального размера без счетчика длины. Диапазоны, которые допускают кодирование всех значений в двух октетах, попадают в выровненное по октету битовое поле без счетчика длины. В остальных случаях значение кодируется в минимальном количестве октетов (используя неотрицательное двоичное целое или двоично-дополнительное до 2 целое кодирование) и добавляется детерминант длины. В том случае, когда целочисленное значение может быть закодировано менее чем в 127 октетов (как смещение от некоторой нижней границы, которая может быть определена) и нет конечных верхней и нижней границ, имеется одинооктетный детерминант длины; иначе длина кодируется в наименьшем необходимом количестве битов. Прочие случаи не имеют практического значения, но специфицированы для полноты.

2 (вариант UNALIGNED) Ограниченные целые кодируются в наименьшем необходимом для представления диапазона количестве битов независимо от их размера. Неограниченные целые числа кодируются, как в примечании 1.

12.1 Если в спецификации ограничения целочисленного типа присутствует маркер расширения, то к списку полей в битовом поле длины должен быть добавлен один бит. Бит должен быть равен 1, если значение, которое должно кодироваться, не находится в пределах диапазона корня расширения, и нулю — в противном случае. В первом случае значение должно быть добавлено к списку полей как неограниченное целое значение, как определено в 12.2.4—12.2.6, и процедура завершается. Во втором случае значение должно быть закодировано так, как если бы маркер расширения отсутствовал.

12.2 Если маркер расширения не представлен в спецификации ограничения целочисленного типа, тогда применяется следующая процедура.

12.2.1 Если видимые для PER ограничения допускают для целого единственное значение, то не должно быть никакого дополнения к списку полей, и эта процедура завершается.

12.2.2 Если видимые для PER ограничения допускают ограниченное целое число в качестве целочисленного значения, то оно должно быть преобразовано в поле согласно процедурам 10.5 (кодирование ограниченного целого числа), а затем должны быть применены процедуры 12.2.5—12.2.6.

12.2.3 Если видимые для PER ограничения допускают полуограниченное целое число в качестве целочисленного значения, то оно должно быть преобразовано в поле согласно процедурам 10.7 (кодирование полуограниченного целого числа), а затем должны быть применены процедуры 12.2.6.

12.2.4 Если видимые для PER ограничения допускают целое не только как ограниченное или полуограниченное целое число, то оно должно быть преобразовано в поле согласно процедурам 10.8 (кодирование неограниченного целого числа), а затем должны быть применены процедуры 12.2.6.

12.2.5 Если процедуры, использованные для кодирования целочисленного значения в поле, не привели к случаю неопределенной длины (см. 10.5.7.4 и 10.8.2), то это поле должно быть добавлено в конец списка полей, и эта процедура завершается.

12.2.6 В противном случае (т. е. в случае неопределенной длины) должны быть применены процедуры 10.9 для добавления поля в список полей после одного из следующих элементов:

а) ограниченного детерминанта длины len (как определено 10.5.7.4), если видимое для PER ограничение удерживает тип в пределах верхней и нижней границ и (если тип расширяемый) значение находится в диапазоне корня расширения. Нижняя граница lb, используемая в детерминанте длины, должна быть равна 1, а верхняя граница ub должна быть счетчиком количества октетов, требуемых для хранения диапазона целочисленного значения.

Примечание — Значение «foo INTEGER (256..1234567) :: = 256» было бы закодировано как 00xxxxxx00000000, где каждый 'x' представляет нулевой заполняющий бит, который может присутствовать или отсутствовать в зависимости от того, в каком месте в октете находится длина (например, 00 xxxxxx 00000000, если длина начинается на границе октета, и 00 00000000, если она начинается во втором от конца бите октета);

б) неограниченного детерминанта длины len (как определено в 10.7 и 10.8), если видимые для PER ограничения не удерживают тип в пределах верхней и нижней границ или если тип расширяемый и значение не находится в диапазоне корня расширения.

13 Кодирование перечислимого типа

Примечание — Перечислимый тип без маркера расширения кодируется как ограниченное целое, ограничение подтипа которого не содержит маркер расширения. Это означает, что на практике перечислимый тип будет почти всегда кодироваться как битовое поле с минимальным количеством битов, необходимых для

выражения любого перечисления. В присутствии маркера расширения перечислимый тип кодируется как обычно маленькое неотрицательное целое число, если значение не находится в корне расширения.

13.1 Перечисления в корне должны быть отсортированы в возрастающем порядке их перечислимых значений, а затем им должен быть присвоен индекс перечисления, начинающийся с нуля для первого с шагом единица до последнего перечисления в отсортированном списке. Расширяющим дополнениям (которые всегда определяются в возрастающем порядке) должен быть присвоен индекс перечисления, начинающийся с нуля для первого с шагом единица до последнего перечисления в расширяющих дополнениях.

Примечание — ГОСТ Р ИСО/МЭК 8824-1 требует, чтобы каждое последующее расширяющее дополнение имело значение перечисления большее, чем последнее.

13.2 Если в определении перечислимого типа отсутствует маркер расширения, то должен быть закодирован индекс перечисления. Его кодирование должно быть таким, как если бы это было значение ограниченного целочисленного типа, для которого нет маркера расширения, с равной 0 нижней границей, а верхняя граница равна самому большому индексу перечисления, ассоциированному с типом, и эта процедура завершается.

13.3 Если маркер расширения присутствует, то к списку полей в битовом поле длины должен быть добавлен один бит. Он должен быть равен 1, если значение, которое должно быть закодировано, не находится в корне расширения, и нулю — в противном случае. В первом случае дополнительные перечисления должны быть отсортированы в соответствии с 13.1, а значение должно быть добавлено к списку полей как обычно маленькое неотрицательное целое число, значением которого является индекс перечисления дополнительного перечисления при lb, равной 0, и эта процедура завершается. Во втором случае значение должно быть закодировано так, как если бы маркер расширения отсутствовал, как определено в 13.2.

Примечание — Нет видимых для PER ограничений, которые могут применяться к перечислимому типу.

14 Кодирование действительного типа

Примечание — Для действительного типа используют октеты содержимого CER/DER с предшествующим детерминантом длины, который на практике будет одним октетом.

14.1 Если основанием абстрактного значения является 10, то основанием закодированного значения должно быть 10, а если основанием абстрактного значения является 2, то основанием закодированного значения должно быть 2.

14.2 Должно использоваться кодирование для REAL, определенное для правил канонического и различающего кодирования в ГОСТ Р ИСО/МЭК 8825-1, приводящее к выровненному по октету битовому полю, которые являются октетами содержимого кодирования CER/DER. Содержимое этого кодирования состоит из *n* октетов и размещается в выровненном по октету поле из *n* октетов. Для добавления этого выровненного по октету битового поля из *n* октетов к списку полей с предшествующим неограниченным детерминантом длины, равным *n*, должны применяться процедуры 10.9.

15 Кодирование типа «битовая строка»

Примечание — Битовые строки с ограниченной длиной, меньшей или равной двум октетам, не приводят к выравниванию по октету. Большие битовые строки выравниваются по октету. Если длина фиксирована ограничениями и верхняя граница меньше 64K, то явное кодирование длины отсутствует, в противном случае включается кодирование длины, которое может иметь любую из форм, установленную ранее для кодирования длины, включая фрагментацию для больших битовых строк.

15.1 Видимые для PER ограничения могут относиться только к длине битовой строки bitstring.

15.2 Когда нет видимого для PER ограничения и применяется ГОСТ Р ИСО/МЭК 8824-1, 21.7, значение должно быть закодировано без хвостовых битов 0 (означает, что значение без равных 1 битов всегда кодируется как пустая битовая строка).

15.3 Когда есть видимое для PER ограничение и применяется ГОСТ Р ИСО/МЭК 8824-1, 21.7 (т. е. тип «битовая строка» определяется с конструкцией «NamedBitList»), значение должно быть

закодировано с добавлением или удалением хвостовых битов 0, необходимым для обеспечения того, чтобы размер переданного значения являлся наименьшим, способным передать это значение, и соответствовал ограничению эффективного размера.

15.4 Обозначим максимальное число битов в битовой строке (с учетом видимых для PER ограничений на длину) ub, а минимальное число битов — lb. Если не существует конечного максимума, то говорят, что ub не установлена. Если не существует ограничений на минимум, тогда lb имеет нулевое значение. Фактическую длину кодируемого значения битовой строки обозначим *n*.

15.5 Если в спецификации ограничения размера типа «битовая строка» присутствует маркер расширения, то к списку полей в битовом поле длины должен быть добавлен один бит. Он должен быть равен 1, если длина этого кодирования не находится в пределах диапазона корня расширения, и нулю — в противном случае. В первом случае должны быть выполнены процедуры 15.10 для добавления длины как полуограниченного целого числа к списку полей с последующим значением битовой строки. Во втором случае длина и значение должны быть закодированы, как будто маркер расширения отсутствует.

15.6 Если в спецификации ограничения типа «битовая строка» отсутствует маркер расширения, то применяются процедуры 15.7—15.10.

15.7 Если битовая строка ограничена нулевой длиной (ub = 0), то она не должна кодироваться (нет дополнений к списку полей), и процедуры этого раздела завершаются.

15.8 Если все значения битовой строки ограничены одной длиной (ub = lb) и эта длина меньше или равна 16 битов, то битовая строка должна быть помещена в битовое поле ограниченной длины ub, которое должно быть добавлено к списку полей без детерминанта длины, и процедуры этого раздела завершаются.

15.9 Если все значения битовой строки ограничены одной длиной (ub = lb) и эта длина больше 16 битов, но меньше 64K битов, то битовая строка должна быть помещена в выровненное по октету битовое поле длиной ub (которая не обязательно кратна восьми битам) и добавлена к списку полей без детерминанта длины, и процедуры этого раздела завершаются.

15.10 Если 15.7—15.9 неприменимы, то битовая строка должна быть помещена в выровненное по октету битовое поле длиной *n* битов и должны быть выполнены процедуры 10.9 для добавления этих *n* битов выровненного по октету поля к списку полей с предшествующим детерминантом длины, равным *n* битам, в виде ограниченного целого числа, если ub меньше 64K, или полуограниченного целого числа, если ub больше этого значения или не установлена. lb устанавливается так, как определено выше.

Примечание — Фрагментация применяется для неограниченных или больших ub после 16K, 32K, 48K или 64K битов.

16 Кодирование типа «строка октетов»

Примечание — Строка октетов фиксированной длины, меньшей или равной двум октетам, не выравнивается по октету. Все другие строки октетов выравниваются по октету. Строки октетов фиксированной длины кодируются без октетов длины, если они короче 64K. Для неограниченных строк октетов длина кодируется явно (в случае необходимости, с фрагментацией).

16.1 Видимые для PER ограничения могут относится только к длине строки октетов.

16.2 Обозначим максимальное количество октетов в строке октетов (с учетом видимых для PER ограничений на длину) ub, а минимальное число октетов — lb. Если не существует конечного максимума, то говорят, что ub не установлена. Если не существует ограничения на минимум, то lb имеет нулевое значение. Фактическую длину кодируемого значения строки октетов обозначим *n* (октетов).

16.3 Если есть видимое для PER ограничение размера и в нем присутствует маркер расширения, то к списку полей в поле битов длины должен быть добавлен один бит. Он должен быть равен 1, если длина этого кодирования не находится в пределах диапазона корня расширения, и нулю — в противном случае. В первом случае должны быть выполнены процедуры 16.8 для добавления к списку полей длины в виде полуограниченного целого числа с последующим значением строки октетов. Во втором случае длина и значение должны быть закодированы так, как будто маркер расширения отсутствует.

16.4 Если в спецификации ограничения типа «строка октетов» отсутствует маркер расширения, то применяются 16.5—16.8.

16.5 Если строка октетов ограничена нулевой длиной (ub = 0), то она не должна кодироваться (нет дополнений к списку полей), и процедуры этого раздела завершаются.

16.6 Если все значения строки октетов ограничены одной длиной (ub = lb) и эта длина меньше или равна двум октетам, то строка октетов должна быть помещена в битовое поле с числом битов, равным кратной восьми ограниченной длине ub, которое должно быть добавлено к списку полей без детерминанта длины, и процедуры этого раздела завершаются.

16.7 Если все значения строки октетов ограничены одной длиной (ub = lb) и эта длина больше двух октетов, но меньше 64K, то строка октетов должна быть помещена в выровненное по октету битовое поле с ограниченной длиной ub октетов, которое должно быть добавлено к списку полей без детерминанта длины, и процедуры этого раздела завершаются.

16.8 Если 16.5—16.7 не применимы, то строка октетов должна быть помещена в выровненное по октету поле битов длиной *n* октетов, и должны быть вызваны процедуры 10.9 для добавления этих *n* октетов выровненного по октету битового поля к списку полей с предшествующим детерминантом длины, равным *n* октетам, в виде ограниченного целого числа, если ub установлена, или в виде полуограниченного целого числа, если ub не установлена (lb устанавливается так, как определено выше).

Примечание — Процедуры фрагментация могут применяться после 16K, 32K, 48K или 64K октетов.

17 Кодирование вырожденного типа

Примечание — Вырожденный тип является, по существу, держателем места и имеет практическое значением только как компонент выбора или факультативный компонент множества или последовательности. Идентификация вырожденного типа в выборе или его присутствие в качестве факультативного элемента осуществляется в настоящих правилах кодирования без необходимости иметь октеты, представляющие null. Следовательно, значения null никогда не вносят вклад в октеты кодирования.

Не должно быть дополнения к списку полей для вырожденного значения.

18 Кодирование типа «последовательность»

Примечание — Тип «последовательность» начинается с преамбулы, которая является битовой картой. Если тип «последовательность» не имеет маркера расширения, то в битовой карте, закодированной как битовое поле фиксированной длины, записано только наличие или отсутствие в типе заданных по умолчанию и факультативных компонентов. Если тип «последовательность» имеет маркер расширения, то битовая карта следует за единственным битом, который сообщает, присутствуют ли фактически в кодировании значения расширяющих дополнений. Преамбула кодируется без какого-либо детерминанта длины при условии, что она имеет длину меньше 64K битов, в противном случае детерминант длины кодируется для получения фрагментации. За преамбулой следуют поля, которые по порядку кодируют все компоненты. Если есть расширяющие дополнения, то непосредственно перед первым из них стоит кодирование (как полуограниченного целого числа) счетчика количества расширяющих дополнений в кодируемом типе, за которым следует битовая карта длиной, равной этому счетчику, в которой записано наличие или отсутствие значений каждого расширяющего дополнения. За этим следуют кодирования расширяющих дополнений так, как если бы они были значениями полей открытого типа.

18.1 Если тип «последовательность» имеет маркер расширения, то сначала к списку полей в поле битов длины должен быть добавлен один бит. Он должен быть единицей, если в данном кодировании присутствуют значения расширяющих дополнений, и нулем — в противном случае. (Далее по тексту этот бит называется «бит расширения»). Если маркер расширения отсутствует, то бит расширения не добавляется.

18.2 Если тип «последовательность» имеет *n* компонентов в корне расширения, которые помечены как OPTIONAL или DEFAULT, то для добавления к списку полей должно быть создано одно битовое поле с *n* битами. Биты этого поля, взятые по порядку, должны кодировать наличие или отсутствие кодирования каждого факультативного или заданного по умолчанию компонента в типе «последовательность». Значение бита 1 кодирует присутствие компонента, а значение бита 0 — отсутствие. Головной бит в преамбуле должен кодировать наличие или отсутствие первого факультативного или заданного по умолчанию компонента, а завершающий бит должен кодировать наличие или отсутствие последнего факультативного или заданного по умолчанию компонента.

18.3 Если $n < 64K$, то битовое поле должно быть добавлено к списку полей. Если $n \geq 64K$, то

должны быть использованы процедуры 10.9 для добавления этого битового поля из *n* битов к списку полей с предшествующим детерминантом длины, равному *n* битам, в виде ограниченного целого числа с ub и lb, равными *n*.

Примечание — В этом случае ub и lb будут проигнорированы процедурами для длины. Здесь эти процедуры вызываются для обеспечения фрагментации большой преамбулы. Вероятно, такая ситуация будет возникать очень редко.

18.4 За преамбулой должны по порядку следовать списки полей каждого присутствующего компонента из значения последовательности.

18.5 Для CANONICAL-PER всегда должны отсутствовать кодирования компонентов, отмеченных как DEFAULT, если значение, которое должно быть закодировано, является значением по умолчанию. Для BASIC-PER всегда должны отсутствовать кодирования компонентов, отмеченных как DEFAULT, если значение, которое должно быть закодировано, является значением по умолчанию для простого типа (см. 3.7.25), в противном случае значение должно быть закодировано только тогда, когда оно явно присутствует в абстрактном значении последовательности.

18.6 Этим завершается кодирование, если бит расширения отсутствует или равен нулю. Если бит расширения присутствует и равен единице, то применяются следующие процедуры.

18.7 Обозначим количество расширяющих дополнений в кодируемом типе *n*. Тогда для добавления к списку полей должно быть создано битовое поле с *n* битами. Биты этого поля, взятые по порядку, должны кодировать наличие или отсутствие кодирования каждого расширяющего дополнения в кодируемом типе. Значение бита 1 соответствует присутствию, а значение бита 0 — отсутствию кодирования расширяющего дополнения. Головной бит в этом поле кодирует наличие или отсутствие первого расширяющего дополнения, а завершающий бит — наличие или отсутствие последнего расширяющего дополнения.

18.8 Должны быть осуществлены процедуры 10.9 для добавления этого битового поля из *n* битов к списку полей с предшествующим детерминантом длины, равным *n*, в виде обычно маленькой длины.

Примечание — *n* не может быть нулем, так как эта процедура вызывается только тогда, когда есть по крайней мере одно кодируемое расширяющее дополнение.

18.9 Далее следуют списки полей, содержащие по порядку кодирования каждого присутствующего расширяющего дополнения. Каждое расширяющее дополнение, являющееся «ComponentType» (т. е. не «ExtensionAdditionGroup»), должно быть закодировано так, как если бы оно было значением поля открытого типа, как определено в 10.2.1. Каждое расширяющее дополнение, являющееся «ExtensionAdditionGroup», должно быть закодировано как последовательность в соответствии с 18.2—18.6, которая затем кодируется так, как если бы она была значением поля открытого типа, как определено в 10.2.1. Если все компоненты значений «ExtensionAdditionGroup» отсутствуют, то «ExtensionAdditionGroup» кодируется как отсутствующее расширяющее дополнение (т. е. соответствующий бит в описанном в 18.7 битовом поле должен быть равен 0).

19 Кодирование типа «последовательность-из»

19.1 Видимые для PER ограничения могут относится к числу компонентов типа «последовательность-из».

19.2 Обозначим максимальное (с учетом видимых для PER ограничений) количество компонентов в «последовательности-из» ub, а минимальное — lb. Если не существует конечного максимума или ub ≥ 64K, то говорят, что ub не установлена. Если не существует ограничения на минимум, то lb имеет нулевое значение. Обозначим число компонентов в фактически кодируемом значении «последовательности-из» *n*.

19.3 Кодирование каждого компонента «последовательности-из» будет генерировать ряд полей, которые добавляются к списку полей для типа «последовательность-из».

19.4 Если есть видимое для PER ограничение и в нем присутствует маркер расширения, то к списку полей в битовом поле длины должен быть добавлен один бит. Он должен быть равен 1, если число компонентов в данном кодировании не находится в пределах диапазона корня расширения, и 0 — в противном случае. В первом случае должны использоваться процедуры 10.9 для добавления

длины в виде полуограниченного целого числа к списку полей с последующими значениями компонентов. Во втором случае длина и значение должны быть закодированы так, как если бы маркер расширения отсутствовал.

19.5 Если количество компонентов фиксировано (ub = lb) и ub < 64K, то для «последовательности-из» не должно быть детерминанта длины, а поля каждого компонента должны быть по порядку добавлены к списку полей «последовательности-из».

19.6 В противном случае должны использоваться процедуры 10.9 для добавления списка полей, порожденных *n* компонентами, с предшествующим детерминантом длины, равным *n* компонентам, в виде ограниченного целого числа, если ub установлена, или в виде полуограниченного целого числа, если ub не установлена. lb устанавливается так, как определено выше.

Примечания

1 Процедуры фрагментации могут применяться после 16K, 32K, 48K или 64K компонентов.

2 Точки разрыва для фрагментации находятся между полями. Число битов до точки разрыва необязательно кратно восьми.

20 Кодирование типа «множество»

Тип «множество» должен иметь в «RootComponentTypeList» элементы, отсортированные в каноническом порядке, определенном ГОСТ Р ИСО/МЭК 8824-1, 8.4, и, кроме того, для определения порядка кодирования компонентов, когда один или несколько компонентов являются нетегированными выборочными типами, каждый нетегированный выборочный тип упорядочивается так, как если бы у него был тег, равный наименьшему тегу в «RootAlternativeTypeList» этого выборочного типа или в любом вложенном нетегированном выборочном типе. Элементы множества, которые встречаются в «RootComponentTypeList», должны быть закодированы так, как если бы они были значением типа «последовательность». Элементы множества, которые встречаются в «ExtensionAdditionList», должны быть закодированы так, как если бы они были компонентами типа «последовательность», как определено в 18.9 (т. е. они кодируются в том порядке, в каком были определены).

Пример — В следующем примере, где подразумевается среда тегирования IMPLICIT TAGS:

```
A ::= SET
{
        a [3] INTEGER,
        b [1] CHOICE
        {
            c [2] INTEGER
            d [4] INTEGER
    },
    e CHOICE
    {
        f CHOICE
        {
            g [5] INTEGER,
            h [6] INTEGER
        },
        i CHOICE
        {
            j [0] INTEGER
        }
    }
}
```

компоненты множества всегда будут кодироваться в порядке e, b, a, так как теги сортируются в порядке [0], [1], [3].

21 Кодирование типа «множество-из»

21.1 Для CANONICAL-PER кодирования значений компонентов типа «множество-из» должны появляться в возрастающем порядке; кодирования компонентов сравниваются как битовые строки, заполненные нулевыми битами до границы октета.

Примечание — Заполняющие биты, добавленные для выравнивания по октету при сортировке, не появляются в фактическом кодировании.

21.2 Для BASIC-PER «множество-из» должно быть закодировано так, как если бы это был тип «последовательность-из».

22 Кодирование выборочного типа

Примечание — Выборочный тип кодируется путем кодирования индекса, определяющего выбранную альтернативу. Индекс кодируется как ограниченное целое число (если в выборочном типе нет маркера расширения, когда индекс кодируется как обычно маленькое неотрицательное целое число) и, следовательно, занимает битовое поле фиксированной длины с минимальным количеством битов, необходимых для кодирования индекса. (Хотя, в принципе, оно может быть произвольно большим). Далее следует кодирование выбранной альтернативы с альтернативами, которые являются расширяющими дополнениями, закодированными так, как если бы они были значениями полей открытого типа. Когда выбор имеет только одну альтернативу, индекс не кодируется.

22.1 На кодирование выборочных типов видимые для PER ограничения не влияют.

22.2 Каждый компонент выбора имеет связанный с ним индекс, значение которого равно нулю для первой альтернативы в корне выбора (альтернативы берутся в каноническом порядке, определенном в ГОСТ Р ИСО/МЭК 8824-1, 8.4), равно единичному значению для второй, и так далее до последнего компонента в корне расширения выбора. Аналогично значение индекса присваивается каждому расширяющему дополнению, начиная с нуля, как и для компонентов корня расширения. Обозначим n значение самого большого индекса в корне.

Примечание — ГОСТ Р ИСО/МЭК 8824-1, 28.4, требует, чтобы каждое последующее расширяющее дополнение имело значение тега большее, чем последнее, добавленное к «AdditionalAlternativeTypeList».

22.3 Для канонического упорядочения альтернатив, которые содержат нетегированный выбор, каждый нетегированный тип выбора должен быть упорядочен так, как если бы он имел тег, равный наименьшему тегу в корне расширения данного выборочного типа или любого вложенного нетегированного выборочного типа.

22.4 Если выбор имеет только одну альтернативу в корне расширения, то индекс не должен кодироваться при выборе этой альтернативы.

22.5 Если выборочный тип имеет маркер расширения, то сначала к списку полей в поле битов длины должен быть добавлен один бит. Он равен единице, если в данном кодировании присутствуют значения расширяющих дополнений, и нулю — в противном случае. (Далее по тексту этот бит называется «бит расширения»). Если маркер расширения отсутствует, то бит расширения не должен добавляться.

22.6 Если бит расширения отсутствует, то индекс выбранной альтернативы должен кодироваться в поле по процедурам раздела 12 как значение целочисленного типа (без маркера расширения в ограничении подтипа), заключенное в диапазоне от 0 до n, и это поле должно быть добавлено к списку полей. За этим полем должны следовать поля выбранной альтернативы, и процедуры настоящего раздела завершаются.

22.7 Если бит расширения присутствует и выбранная альтернатива находится в корне расширения, то индекс выбранной альтернативы должен быть закодирован так, как если бы маркер расширения отсутствовал, в соответствии с разделом 12, и процедуры настоящего раздела завершаются.

22.8 Если бит расширения присутствует и выбранная альтернатива не находится в корне расширения, то индекс выбранной альтернативы должен быть закодирован как обычно маленькое неотрицательное число с lb = 0, и это поле должно быть добавлено к списку полей. За этим полем должен следовать список полей, содержащих выбранную альтернативу, закодированную как значение поля открытого типа, как определено в 10.2, и процедуры этого раздела завершаются.

Примечание — Скобки версии в определении расширяющих дополнений выбора не влияют на кодирование «ExtensionAdditionAlternatives».

23 Кодирование типа «идентификатор объекта»

Примечание — Кодирование типа «идентификатор объекта» использует октеты содержимого BER с предшествующим детерминантом длины, который, на практике, будет одним октетом.

Должно применяться кодирование, определенное для BER, дающее выровненное по октету битовое поле, которое является октетами содержимого кодирования BER. Октеты содержимого этого кодирования BER состоят из *n* октетов и помещаются в выровненное по октету битовое поле из *n* октетов. Должны быть использованы процедуры 10.9 для добавления к списку полей этого выровненного по октету битового поля с предшествующим детерминантом длины, равным *n*, в виде полуограниченного целого значения счетчика октетов.

23.1 Кодирование типа «относительный идентификатор объекта»

Примечание — Кодирование типа «относительный идентификатор объекта» использует октеты содержимого BER с предшествующим детерминантом длины, который, на практике, будет одним октетом. Последующий текст идентичен тексту раздела 23.

Должно применяться кодирование, определенное для BER, дающее выровненное по октету битовое поле, которое является октетами содержимого кодирования BER. Октеты содержимого этого кодирования BER состоят из *n* октетов и помещаются в выровненное по октету битовое поле из *n* октетов. Должны быть использованы процедуры 10.9 для добавления к списку полей этого выровненного по октету битового поля с предшествующим детерминантом длины, равным *n*, в виде полуограниченного целого значения счетчика октетов.

24 Кодирование типа «встроенное-здп»

24.1 Установлено два способа, которыми может быть закодирован тип «встроенное-здп»:

а) альтернатива «syntaxes» типа «встроенное-здп» ограничена видимым для PER внутренним ограничением типа единственным значением или «identification» ограничена видимым для PER внутренним ограничением типа альтернативой «fixed». В таком случае должно быть закодировано только значение данных «data-value»; этот случай называется «предопределенным»;

б) внутреннее ограничение типа не используется для ограничения альтернативы «syntaxes» единственным значением или «identification» — альтернативой «fixed», в таком случае должны быть закодированы как «identification», так и «data-value»; этот случай называется «общим».

24.2 В «предопределенном» случае кодированием значения типа «встроенное-здп» должно быть кодирование PER значения типа OCTET STRING. Значение OCTET STRING должно быть октетами, образующими полное кодирование единственного значения данных, указанного в ГОСТ Р ИСО/МЭК 8824-1, 32.3а.

24.3 В общем случае кодированием значения типа «встроенное-здп» должно быть кодирование PER типа, определенного в ГОСТ Р ИСО/МЭК 8824-1, 32.5, с исключенным элементом «descriptor» (а именно, не должно быть битового отображения «OPTIONAL» в заголовке кодирования SEQUENCE). Значение «data-value» OCTET STRING должно быть октетами, образующими полное кодирование единственного значения данных, указанного в ГОСТ Р ИСО/МЭК 8824-1, 32.3а.

25 Кодирование значения внешнего типа

25.1 Кодирование значения внешнего типа должно быть кодированием PER следующего типа «последовательность», для которого принята среда явного тегирования EXPLICIT TAGS, со значением, определенным в последующих подразделах:

```
[UNIVERSAL 8] IMPLICIT SEQUENCE {
    direct-reference        OBJECT IDENTIFIER OPTIONAL,
    indirect-reference      IDENTIFIER OPTIONAL,
    data-value-descriptor ObjectDescriptor OPTIONAL,
    encoding                CHOICE {
        single-ASN1-type            [0] ABSTRACT-SYNTAX.&Type,
        octet-aligned               [1] IMPLICIT OCTET STRING,
        arbitrary                   [2] IMPLICIT BIT STRING}}
```

Примечание — Этот тип «последовательность» является тем же типом, который был определен в ГОСТ Р ИСО/МЭК 8824—93.

25.2 Значения компонентов зависят от передаваемого абстрактного значения, которое является значением типа, определенного в ГОСТ Р ИСО/МЭК 8824-1, 32.5.

25.3 Компонент «data-value-descriptor» должен присутствовать только в том случае, если «data-value-descriptor» присутствует в абстрактном значении, и иметь то же самое значение.

25.4 Значения «direct-reference» и «indirect-reference» должны присутствовать или отсутствовать в соответствии с таблицей 1. Таблица 1 отображает альтернативы «identification» внешнего типа, определенные в ГОСТ Р ИСО/МЭК 8824-1, 32.5, в компоненты внешнего типа «direct-reference» и «indirect-reference», определенные в 25.1.

Таблица 1 — Альтернативные кодирования для «identification»

identification	direct-reference	indirect-reference
syntaxes	***НЕДОПУСТИМО***	***НЕДОПУСТИМО***
syntax	syntax	ОТСУТСТВУЕТ
presentation-context-id	ОТСУТСТВУЕТ	presentation-context-id
context-negotiation	transfer-syntax	presentation-context-id
transfer-syntax	***НЕДОПУСТИМО***	***НЕДОПУСТИМО***
fixed	***НЕДОПУСТИМО***	***НЕДОПУСТИМО***

25.5 Значение данных должно быть закодировано согласно синтаксису передачи, идентифицированному кодированием, и помещено в альтернативу выбора «encoding», как указано ниже.

25.6 Если значение данных является значением единственного типа данных АСН. 1 и правила кодирования для этого значения данных являются теми же самыми, как и для полного типа данных «EXTERNAL», то отправляемая реализация должна быть «single-ASN1-type».

25.7 Если кодированное значение данных, использующее согласованное или договорное кодирование, содержит целое число октетов, то отправляемая реализация должна быть «octet-aligned».

Примечание — Значение данных, которое является последовательностью типов АСН. 1 и для которого синтаксис передачи специфицирует простое сцепление строк октетов, созданных применением базовых правил кодирования АСН. 1 для каждого типа АСН. 1, попадает в эту категорию, а не в 25.6.

25.8 Если кодированное значение данных, использующее согласованное или договорное кодирование, содержит не целое число октетов, то для «encoding» должно быть выбрано «arbitrary».

25.9 Если для «encoding» выбрано «single-ASN1-type», то тип АСН. 1 должен заменить открытый тип со значением, равным значению кодируемых данных.

Примечание — Диапазон значений, которые могут встретиться в открытом типе, определяется регистрацией значения идентификатора объекта, ассоциированного с «direct-reference», и/или значением целого числа, ассоциированного с «indirect-reference».

25.10 Если для «encoding» выбрано «octet-aligned», то значение данных должно кодироваться в соответствии с согласованным или договорным синтаксисом передачи, а получающиеся октеты должны образовывать значение строки октетов.

25.11 Если для «encoding» выбрано «arbitrary», то значение данных должно кодироваться в соответствии с согласованным или договорным синтаксисом передачи, а результат должен образовать значение битовой строки.

26 Кодирование ограниченных типов символьных строк

Примечания

1 (Вариант ALIGNED) Символьные строки фиксированной длины, меньшей или равной двум октетам, не выравниваются по октету. Символьные строки переменной длины, ограниченные максимальной длиной меньше двух октетов, не выравниваются по октету. Все другие символьные строки выравниваются по октету. Символьные строки фиксированной длины кодируются без октетов длины, если они короче 64К символов. Для неограниченных или ограниченных символьных строк длиной свыше 64К-1 длина кодируется явно (при необходимости, с фрагментацией). Каждый символ NumericString, PrintableString, VisibleString (ISO646String), IASString, BMPString и UniversalString кодируется в число битов, которое является минимальной степенью двух, позволяющей разместить все символы, допускаемые эффективным ограничением PermittedAlphabet.

2 (Вариант UNALIGNED) Символьные строки не выравниваются по октету. Если есть только одно возможное значение длины, то она не кодируется, если строки короче 64К символов. Для неограниченных или

ограниченных символьных строк длиной свыше 64К-1 длина кодируется явно (при необходимости, с фрагментацией). Каждый символ NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString и UniversalString кодируется в число битов, которое является минимальной степенью двух, позволяющей разместить все символы, допускаемые эффективным ограничением PermittedAlphabet.

3 (Размер закодированных символов) Кодирование каждого символа зависит от эффективного ограничения PermittedAlphabet (см. 9.3.10), которое определяет используемый для типа алфавит. Пусть этот алфавит состоит из набора символов ALPHA. Для каждого типа символьных строк известной кратности (см. 3.7.16) есть целочисленное значение, связанное с каждым символом, полученное путем ссылки на некоторую таблицу кодов, связанную с ограниченным типом символьных строк. Набор значений BETA, соответствующих набору символов ALPHA, используется для определения кодирования, которое должно применяться, следующим образом: число битов для кодирования каждого символа полностью определяется числом элементов N в наборе BETA (или ALPHA). Для варианта UNALIGNED оно равно наименьшему количеству битов, в которых можно закодировать значение $N-1$ как неотрицательное двоичное целое число. Для варианта ALIGNED оно равно минимальному количеству битов, которое является степенью двух и в которых можно закодировать значение $N-1$. Пусть выбранное количество битов равно B. Тогда, если каждое значение в наборе BETA может быть закодировано (без преобразования) в B битах, то значения в наборе BETA используются для представления соответствующих символов в наборе ALPHA. В противном случае значения в наборе BETA берутся в возрастающем порядке и заменяются значениями 0, 1, 2, и так далее до $N-1$, и эти значения используются для представления соответствующих символов. В результате всегда используется минимум битов (в варианте ALIGNED — округленный до следующей степени двух). Предпочтение отдается использованию значений, обычно связанных с символами, но если какое-то из этих значений не может быть закодировано минимальным количеством битов, то применяется уплотнение.

26.1 Типами символьных строк известной кратности являются следующие ограниченные типы символьных строк: NumericString, PrintableString, VisibleString (ISO646String), IA5String, BMPString и UniversalString. Эффективные ограничения PermittedAlphabet являются видимыми для PER только для этих типов.

26.2 Нотация эффективного ограничения размера может определять верхнюю границу aub длины абстрактной символьной строки. В противном случае aub не устанавливается.

26.3 Нотация эффективного ограничения размера может определять ненулевую нижнюю границу alb длины абстрактной символьной строки. В противном случае alb — нуль.

Примечание — Видимые для PER ограничения применяются только к типам символьных строк известной кратности. Для других ограниченных типов символьных строк aub не устанавливается, и alb равняется нулю.

26.4 Если тип является расширяемым для кодирований PER (см. 9.3.15), то к списку полей должно быть добавлено битовое поле, состоящее из единственного бита. Этот бит должен быть равен нулю, если значение находится в диапазоне корня расширения, и единице — в противном случае. Если значение находится вне пределов диапазона корня расширения, то последующее кодирование должно быть таким, как если бы не было ни эффективного ограничения размера, ни эффективного ограничения PermittedAlphabet.

Примечание — Только типы символьных строк известной кратности могут быть расширяемыми для кодирований PER. Маркеры расширения в других типах символьных строк не влияют на кодирование PER.

26.5 Настоящий подраздел для символьных строк известной кратности. Кодирование других ограниченных типов символьных строк специфицируется в 26.6.

26.5.1 Эффективный допустимый алфавит определяется как допускаемый ограничением PermittedAlphabet, или как весь алфавит встроенного типа, если нет ограничения PermittedAlphabet.

26.5.2 N обозначает число символов в эффективном допустимом алфавите. B обозначает минимальное целое число, такое, что $2^B \geq N$. $B2$ — минимальный показатель степени 2, больший или равный B. Тогда в варианте ALIGNED каждый символ должен кодироваться в $B2$ битов, а в варианте UNALIGNED — в B битов. Число битов, определенных по этому правилу, обозначено b.

26.5.3 С каждым символом связывается числовое значение v ссылкой на ГОСТ Р ИСО/МЭК 8824-1, раздел 38, следующим образом. Для UniversalString это значение, которое используется для определения канонического порядка в ГОСТ Р ИСО/МЭК 8824-1, 38.4 (оно находится в диапазоне от 0 до $2^{31} - 1$). Для BMPString это значение, которое используется для определения канонического порядка в ГОСТ Р ИСО/МЭК 8824-1, 38.5 (оно находится в диапазоне от 0 до $2^{16} - 1$). Для NumericString, PrintableString, VisibleString и IA5String это значение, которое определяется ИСО/МЭК 646 для кодирования соответствующего символа. (Для IA5String это диапазон от 0 до 127, для VisibleString — от 32 до 126, для NumericString — от 32 до 57, для PrintableString —

от 32 до 122. Для IA5String и VisibleString присутствуют все значения диапазона, а для NumericString и PrintableString используются не все значения диапазона).

26.5.4 Наименьшее значение в диапазоне для набора символов допустимого алфавита обозначим lb, а наибольшее — ub. Тогда кодирование символа в *b* битах является неотрицательным двоичным целым кодированием значения *v*, идентифицированного следующим образом;

а) если ub ≤ 2^b — 1, то *v* — указанное выше значение; в противном случае

б) символы размещаются в каноническом порядке, определенном в ГОСТ Р ИСО/МЭК 8824-1, раздел 38. Первому из них присваивается нулевое значение, а следующему в каноническом порядке присваивается значение на единицу больше значения, присвоенного предшествующему символу в каноническом порядке. Это и будут значения *v*.

Примечание — Перечисление а) никогда не применяется к ограниченному или неограниченному символу NumericString, который всегда кодируется в четыре бита или менее, используя б).

26.5.5 Кодирование всей символьной строки будет получено неотрицательным двоичным целым кодированием каждого символа (используя соответствующее значение *v*) в *b* битах, которые должны быть сцеплены для формирования битового поля, кратного *b* битам.

26.5.6 Если aub равно alb и меньше 64K, то битовое поле должно быть добавлено к списку полей как выровненное, если aub более чем в 16 раз превышает *b*, или, в противном случае, как невыровненное по октету. Этим завершаются процедуры настоящего подраздела.

26.5.7 Если aub не равно alb или больше или равно 64K, то должны быть применены процедуры 10.9 для добавления детерминанта длины с *n* в качестве счетчика символов в строке, с нижней границей детерминанта длины alb и верхней границей aub. Затем битовое поле должно быть добавлено к списку полей как выровненное, если aub более чем в 16 раз превышает *b*, или, в противном случае, как невыровненное по октету. Этим завершаются процедуры настоящего подраздела.

26.6 Настоящий подраздел применяется к символьным строкам, которые не являются символьными строками известной кратности. В этом случае ограничения являются невидимыми для PER, а тип не может быть расширяемым для кодирования PER.

26.6.1 Используемый ниже термин «базовое кодирование» означает ОПК для BASIC-PER и CANONICAL-BER для CANONICAL-PER.

26.6.2 К символьной строке должно быть применено «базовое кодирование», дающее поле из *n* октетов.

26.6.3 Должны быть применены процедуры 10.9 для добавления неограниченного детерминанта длины с равным *n* счетчиком в октетах, а поле из *n* октетов должно быть добавлено как выровненное по октету битовое поле, и процедуры этого раздела завершаются.

27 Кодирование неограниченного типа символьных строк

27.1 Установлены два способа, которыми может быть закодирован неограниченный тип символьных строк:

а) альтернатива «syntaxes» неограниченного типа символьных строк ограничена видимым для PER внутренним ограничением типа единственным значением или «identification» ограничен видимым для PER внутренним ограничением типа альтернативой «fixed»; в этом случае должно быть закодировано только значение «string-value»; этот случай называется «предопределенным»;

б) внутреннее ограничение типа не используется для ограничения альтернативы «syntaxes» единственным значением или «identification» — альтернативной «fixed»; в таком случае должны быть закодированы как «identification», так и «string-value»; этот случай называется «общим».

27.2 В «предопределенном» случае кодированием значения типа CHARACTER STRING должно быть кодирование PER значения типа OCTET STRING. Значение OCTET STRING должно быть октетами, образующими полное кодирование значения символьной строки, указанной в ГОСТ Р ИСО/МЭК 8824-1, 39.3а.

27.3 В общем случае кодированием значения типа CHARACTER STRING должно быть кодирование PER типа, определенного в ГОСТ Р ИСО/МЭК 8824-1, 39.5, с исключенным элементом «descriptor» (а именно не должно быть битового отображения «OPTIONAL» в заголовке кодирования SEQUENCE). Значение «string-value» OCTET STRING должно быть октетами, образующими полное кодирование единственного значения данных, указанного в ГОСТ Р ИСО/МЭК 8824-1, 39.3а.

28 Идентификаторы объектов синтаксисов передачи

28.1 Правила кодирования, специфицированные в настоящем стандарте, могут быть указаны и применяться всякий раз, когда есть необходимость специфицировать однозначное представление в виде битовой строки для всех значений одного типа АСН. 1.

28.2 Следующие идентификаторы объектов и значения описателей объектов предназначены для идентификации и описания правил кодирования, определенных в настоящем стандарте:

Для BASIC-PER, вариант ALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) aligned (0)}

«Packed encoding of a single ASN. 1 type (basic aligned)»

Для BASIC-PER, вариант UNALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) unaligned (1)}

«Packed encoding of a single ASN. 1 type (basic unaligned)»

Для CANONICAL-PER, вариант ALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) aligned (0)}

«Packed encoding of a single ASN. 1 type (canonical aligned)»

Для CANONICAL-PER, вариант UNALIGNED:

{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) unaligned (1)}

«Packed encoding of a single ASN. 1 type (canonical unaligned)»

28.3 Когда прикладной стандарт определяет абстрактный синтаксис как множество абстрактных значений, каждое из которых является значением некоторого конкретно названного типа АСН. 1, определенного с использованием нотации АСН. 1, тогда значения идентификаторов объектов, указанные в 28.2, вместе с именем абстрактного синтаксиса могут быть использованы для идентификации того синтаксиса передачи, который является результатом применения определенных в настоящем стандарте правил кодирования к конкретно названному типу АСН. 1, использованному при определении абстрактного синтаксиса.

Примечание — В частности, эти идентификаторы правил кодирования могут появиться в поле «имя синтаксиса передачи» протокола уровня представления (ГОСТ 34.972).

28.4 Имена, определенные в 28.2, не должны использоваться с именем абстрактного синтаксиса для идентификации синтаксиса передачи, если не удовлетворяются условия 28.3 для определения абстрактного синтаксиса.

ПРИЛОЖЕНИЕ А
(справочное)

Пример кодирования

В настоящем приложении иллюстрируется использование правил уплотненного кодирования, специфицированных в настоящем стандарте, на примере представления в октетах (гипотетической) персональной записи, определенной с использованием ACH. 1.

А.1 Запись, которая не использует ограничения подтипа

А.1.1 Описание ACH. 1 структуры записи

Ниже формально описана структура гипотетической персональной записи с использованием ACH. 1, специфицированной в ГОСТ Р ИСО/МЭК 8824-1 для определения типов. Описание идентично примеру, приведенному в ГОСТ Р ИСО/МЭК 8825-1, приложение А.

```
PersonnelRecord :: = [APPLICATION 0] IMPLICIT SET {

    name            Name,
    title           [0] VisibleString,
    number          EmployeeNumber,
    dateOfHire      [1] Date,
    nameOfSpouse    [2] Name,
    children        [3] IMPLICIT
        SEQUENCE OF ChildInformation DEFAULT {} }

ChildInformation :: = SET
    {name               Name,
    dateOfBirth         [0] Date}
Name :: = [APPLICATION 1] IMPLICIT SEQUENCE
    {givenName          VisibleString,
    initial             VisibleString,
    familyName          VisibleString}

EmployeeNumber :: = [APPLICATION 2] IMPLICIT INTEGER

Date :: = [APPLICATION 3] IMPLICIT VisibleString - - YYYYMMDD
```

А.1.2 Описание ACH. 1 значения записи

Ниже приведено формальное описание с использованием ACH. 1 значения персональной записи для Джона Смита (John Smith):

```
{ name {givenName «John», initial «P», familyName «Smith»},
  title          «Director»,
  number         51,
  dateOfHire     «19710917»,
  nameOfSpouse {givenName «Mary», initial «T», familyName «Smith»}, children
    {{name{givenName «Ralph», initial «T», familyName «Smith»}, dateOfBirth «19571111»},
    {name{givenName «Susan», initial «B», familyName «Jones»}, dateOfBirth «19590717»}}}
```

А.1.3 Представление PER (ALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта ALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями.

Длина этого кодирования составляет 94 октета. Для сравнения, то же самое значение PersonnelRecord, закодированное с использованием варианта UNALIGNED PER, занимает 84 октета, BER с определенной формой длины — по крайней мере 136 октетов, а BER с неопределенной формой длины — по крайней мере 161 октет.

А.1.3.1 *Шестнадцатеричное представление*

80044A6F 686E0150 05536D69 74680133 08446972 6563746F 72083139 37313039 3137044D 61727901 5405536D 69746802 0552616C 70680154 05536D69 74680831 39353731 31313105 53757361 6E014205 4A6F6E65 73083139 35393037 3137

А.1.3.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец

строки; для выделения символов в символьной строке использован пробел; 'x' представляет нулевой бит заполнения, который иногда используется для выравнивания полей по границе октета.

```
1xxxxxxx                Битовая карта = 1 указывает, что есть «children»
00000100                                  Длина name.givenName = 4
01001010 01101111 01101000 01101110       name.givenName (имя) = «John»

00000001                Длина name.initial = 1
01010000                name.initial = «P»

00000101                Длина name.familyName = 5
01010011 01101101 01101001 01110100 01101000 name.familyName = «Smith»

00000001                Длина number = 1
00110011                number = 51

00001000                Длина title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010 title = «Director»

00001000                Длина dateOfHire = 8
00110001 00111001 00110111 00110001 00110000 00111001 00110001 00111111 dateOfHire = «19590717»

00000100                Длина nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001 nameOfSpouse.givenName = «Mary»

00000001                Длина nameOfSpouse.initial = 1
01010100                nameOfSpouse.initial = «T»

00000101                Длина nameOfSpouse.familyName = 5
01010011 01101101 01101001 01110100 01101000 nameOfSpouse.familyName = «Smith»

00000010                Количество «children»
00000101                Длина children [0]. givenName = 5
01010010 01100001 01101100 01110000 01101000 children [0]. givenName = «Ralph»

00000001                Длина children [0]. initial = 1
01010100                children [0]. initial = «T»

00000101                Длина children [0]. familyName = 5
01010011 01101101 01101001 01110100 01101000 children [0]. familyName = «Smith»

00001000                Длина children [0]. dateOfBirth = 8
00110001 00111001 00110101 00110111 00110001 00110001 00110001 00110001 children [0]. dateOfBirth = «19571111»

00000101                Длина children [1]. givenName = 5
01010011 01110101 01110011 01100001 01101110 children [1]. givenName = «Susan»

00000001                Длина children [1]. initial = 1
01000010                children [1]. initial = «B»

00000101                Длина children [1]. familyName = 5
01001010 01101111 01101110 01100101 01110011 children [1]. familyName = «Jones»

00001000                Длина children [1]. dateOfBirth = 8
00110001 00111001 00110101 00111001 00110000 00110111 00110001 00110111 children [1]. dateOfBirth = «19590717»
```

А.1.4 Представление PER (UNALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта UNALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями. Биты заполнения не встречаются в варианте UNALIGNED, а символы кодируются в минимально возможное число битов.

Длина этого кодирования составляет 84 октета. Для сравнения, то же самое значение PersonnelRecord, закодированное с использованием варианта ALIGNED PER, занимает 94 октета, BER с определенной формой длины — по крайней мере 136 октетов, а BER с неопределенной формой длины — по крайней мере 161 октет.

А.1.4.1 *Шестнадцатеричное представление*

```
824ADFA3 700D005A 7B74F4D0 02661113 4F2CB8FA 6FE410C5 CB762C1C B16E0937 0F2F2035 0169EDD3
D340102D 2C3B3868 01A80B4F 6E9E9A02 18B96ADD 8B162C41 69F5E787 700C2059 5BF765E6 10C5CB57
2C1BB16E
```

A.1.4.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец строки; для выделения символов в символьной строке использован пробел; точка (.) отмечает границу октета; 'x' представляет нулевой бит, использованный для заполнения последнего октета до границы октета.

```
1                       Битовая карта = 1 указывает, что есть «children»

0000010.0               Длина name.givenName = 4
1001010.1101111 1.101000 11.01110 name.givenName = «John»

000.00001               Длина name.initial = 1
101.0000                name.initial = «P»

0000.0101               Длина name.familyName = 5
1010.011 11011.01 110100.1 1110100 .1101000 name.familyName = «Smith»

0.0000001               Длина number = 1
0.0110011               number = 51

0.0001000               Длина title = 8
1.000100 11.01001 111.0010 1100.101 11000.11 111010.0 1101111 .1110010 title = «Director»

0.0001000               Длина dateOfHire = 8
0.110001 01.11001 011.0111 0110.001 01100.00 011100.1 0110001 .0111111 dateOfHire = «19590717»

0.0000100               Длина nameOfSpouse.givenName = 4
1.001101 11.00001 111.0010 1111.001 nameOfSpouse.givenName = «Mary»

00000.001               Длина nameOfSpouse.initial = 1
10101.00                nameOfSpouse.initial = «T»

000001.01               Длина name.OfSpousefamilyName = 5
101001.1 1101101 .1101001 1.110100 11.01000 nameOfSpouse.familyName = «Smith»

000.00010               Количество «children»
000.00101               Длина children [0]. givenName = 5
101.0010 1100.001 11011.00 111000.0 1101000 children [0]. givenName = «Ralph»

.00000001               Длина children [0]. initial = 1
.1010100                children [0]. initial = «T»

0.0000101               Длина children [0]. familyName = 5
1.010011 11.01101 110.1001 1110.100 11010.00 children [0]. familyName = «Smith»

000010.00               Длина children [0]. dateOfBirth = 8
011000.1 0111001 .0110101 0.110111 01.10001 011.0001 0110.001 01100.01 children [0]. dateOfBirth = «19571111»

000001.01               Длина children [1]. givenName = 5
101001.1 1110101 .1110011 1.100001 11.01110 children [1].givenName = «Susan»

000.00001               Длина children [1]. initial = 1
100.0010                children [1]. initial = «B»

0000.0101               Длина children [1]. familyName = 5
1001.100 11011.11 110111.0 1100101 .1110011 children [1]. familyName = «Jones»

0.0001000               Длина children [1]. dateOfBirth = 8
0.110001 01.11001 011.0101 0111.001 01100.00 011011.1 0110001 .0110111x  children [1]. dateOfBirth = «19590717»
```

A.2 Запись, которая использует ограничения подтипа

Это тот же самый пример, который показан в разделе A.1, за исключением того, что в нем использована нотация подтипа для наложения ограничений на некоторые элементы.

A.2.1 Описание АСН.1 структуры записи

Ниже формально описана структура гипотетической персональной записи с использованием АСН. 1, специфицированной в ГОСТ Р ИСО/МЭК 8824-1 для определения типов.

```
PersonnelRecord :: = [APPLICATION 0] IMPLICIT SET {
        name                Name,
        title               [0] VisibleString,
        number              EmployeeNumber,
```

```
        dateOfHire          [1] Date,
        nameOfSpouse        [2] Name,
        children            [3] IMPLICIT
            SEQUENCE OF ChildInformation DEFAULT {} }

    ChildInformation :: = SET
        {name               Name,
        dateOfBirth         [0] Date}

    Name :: = [APPLICATION 1] IMPLICIT SEQUENCE
        {givenName          VisibleString,
        initial             VisibleString (SIZE(1)),
        familyName          VisibleString}

    EmployeeNumber :: = [APPLICATION 2] IMPLICIT INTEGER

    Date :: = [APPLICATION 3] IMPLICIT VisibleStirng (FROM(«0»..«9») ^ SIZE (8)) - - YYYYMMDD

    NameString :: = VisibleString (FROM(«a»..«z»|«A»..«Z»|«-.») ^ SIZE (1..64))
```

А.2.2 Описание ACH. 1 значения записи

Далее формально, с использованием ACH. 1, описано значение персональной записи для Джона Смита (John Smith).

```
{ name {givenName «John», initial «P», familyName «Smith»},
  title          «Director»,
  number         51,
  dateOfHire     «19710917»,
  nameOfSpouse {givenName «Mary», initial «T», familyName «Smith»}, children
    {{name{givenName «Ralph», initial «T», familyName «Smith»}, dateOfBirth «19571111»},
    {name{givenName «Susan», initial «B», familyName «Jones»}, dateOfBirth «19590717»}}}
```

А.2.3 Представление PER (ALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта ALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями.

Длина этого кодирования составляет 74 октета. Для сравнения, то же самое значение PersonnelRecord, закодированное с использованием варианта UNALIGNED PER, занимает 61 октет, BER с определенной формой длины — по крайней мере 136 октетов, а BER с неопределенной формой длины — по крайней мере 161 октет.

А.2.3.1 *Шестнадцатеричное представление*

```
864A6F68 6E501053 6D697468 01330844 69726563 746F7219 7109170C 4D617279 5410536D 69746802 1052616C
70685410 536D6974 68195711 11105375 73616E42 104A6F6E 65731959 0717
```

А.2.3.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец строки; для выделения символов в символьной строке использован пробел; 'x' представляет нулевой бит заполнения, который иногда используется для выравнивания полей по границе октета.

```
1                   Битовая карта = 1 указывает, что есть «children»

000011x             Длина name.givenName = 4
01001010 01101111 01101000 01101110 name.givenName = «John»

01010000            name.initial = «P»

000100xx            Длина name.familyName = 5
01010011 01101101 01101001 01110100 01101000 name.familyName = «Smith»

00000001            Длина number = 1
00110011            number = 51

00001000            Длина title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010 title = «Director»

0001 1001 0111 0001 0000 1001 0001 0111 dateOfHire = «19590717»

000011xx Длина nameOfSpouse. givenName = 4
```

01001101 01100001 01110010 01111001 nameOfSpouse.givenName = «Mary»

01010100 nameOfSpouse. initial = «T»

000100xx Длина nameOfSpouse. familyName = 5
01010011 01101101 01101001 01110100 01101000 nameOfSpouse. familyName = «Smith»

00000010 Количество «children»

000100xx Длина children [0]. givenName = 5
01010010 01100001 01101100 01110000 01101000 children [0]. givenName = «Ralph»

01010100 children [0]. initial = «T»

000100xx Длина children [0]. familyName = 5
01010011 01101101 01101001 01110100 01101000 children [0]. familyName = «Smith»

0001 1001 0101 0111 0001 0001 0001 0001 children [0]. dateOfBirth = «19571111»

000100xx Длина children [1]. givenName = 5
01010011 01110101 01110011 01100001 01101110 children [1]. givenName = «Susan»

01000010 children [1]. initial = «B»

000100xx Длина children [1]. familyName = 5
01001010 01101111 01101110 01100101 01110011 children [1]. familyName = «Jones»

0001 1001 0101 1001 0000 0111 0001 0111 children [1]. dataOfBirth = «19590717»

А.2.4 Представление PER (UNALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта UNALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями. Биты заполнения не встречаются в варианте UNALIGNED, а символы кодируются в минимально возможное число битов.

Длина этого кодирования составляет 61 октет. Для сравнения, то же самое значение PersonnelRecord, закодированное с использованием варианта ALIGNED PER, занимает 74 октета, BER с определенной формой длины — по крайней мере 136 октетов, а BER с неопределенной формой длины — по крайней мере 161 октет.

А.2.4.1 *Шестнадцатеричное представление*

865D51D2 888A5125 F1809984 44D3CB2E 3E9BF90C B8848B86 7396E8A8 8A5125F1 81089B93 D71AA229 4497C632 AE222222 985CE521 885D54C1 70CAC838 B8

А.2.4.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец строки; для выделения символов в символьной строке использован пробел; точка (.) отмечает границу октета; 'x' представляет нулевой бит, использованный для заполнения последнего октета до границы октета.

1 Битовая карта = 1 указывает, что есть «children»

000011 Длина name. givenName = 4
0.01011 101.010 10001.1 101001 name. givenName = «John»

0.10001 name. initial = «P»

000.100 длина name. familyName = 5
01010.0 101000 1.00100 101.111 10001.1 name.familyName = «Smith»

0000000.1 Длина number = 1
0011001.1 number = 51

0000100.0 Длина title = 8
1000100 .1101001 1.110010 11.00101 110.0011 1110.100 11011.11 111001.0 title = «Director»

0001 100.1 0111 000.1 0000 100.1 0001 011.1 dateOfHire = «19590717»

000011 Длина nameOfSpouse. givenName = 4
0.01110 011.100 10110.1 110100 nameOfSpouse. givenName = «Mary»

0.10101 nameOfSpouse. initial = «T»

000.100 Длина nameOfSpouse. familyName = 5

01010.0 101000 1.00100 101.111 10001.1 nameOfSpouse. familyName = «Smith»

0000001.0 Количество «children»

000100 Длина children [0]. givenName = 5
0.10011 011.100 10011.1 101011 1.00011 children [0]. givenName = «Ralph»

010.101 children [0]. initial = «T»

00010.0 Длина children [0]. familyName = 5
010100 1.01000 100.100 10111.1 100011 children [0]. familyName = «Smith»

0.001 1001 0.101 0111 0.001 0001 0.001 0001 children [0]. dataOfBirth = «19571111»

0.00100 Длина children [1]. givenName = 5
010.100 11000.0 101110 0.11100 101.001 children [1]. givenName = «Susan»

00001.1 children [1]. initial = «B»

000100 Длина children [1]. familyName = 5
0.01011 101.010 10100.1 100000 1.01110 children [1]. familyName = «Jones»

000.1 1001 010.1 1001 000.0 0111 00.1 0111xxx children [1]. dateOfBirth = «19590717»

A.3 Запись, использующая маркеры расширения

A.3.1 Описание ACH. 1 структуры записи

Ниже формально описана структура гипотетической персональной записи с использованием ACH. 1, описанной в ГОСТ Р ИСО/МЭК 8824-1 для определения типов.

```
PersonnelRecord :: = [APPLICATION 0] IMPLICIT SET {
        name            Name,
        title           [0] VisibleString,
        number          EmployeeNumber,
        dateOfHire      [1] Date,
        nameOfSpouse    [2] Name,
        children        [3] IMPLICIT
            SEQUENCE (SIZE(2, . . .)) OF ChildInformation OPTIONAL, . . .}

ChildInformation :: = SET
        {name           Name,
        dateOfBirth     [0] Date}
        . . . .
        sex             [1] IMPLICIT ENUMERATED {male (1), female (2), unknown (3)} OPTIONAL
        }

Name :: = [APPLICATION 1] IMPLICIT SEQUENCE
        {givenName      VisibleStrnng,
        initial         VisibleString (SIZE (1)),
        familyName      VisibleString,
        . . . }

EmployeeNumber :: = [APPLICATION 2] IMPLICIT INTEGER (0..9999, . . .)

Date :: = [APPLICATION 3] IMPLICIT VisibleString (FROM(«0»..«9») ^ SIZE (8, . . . , 9..20)) - - YYYYMMDD

NameString :: = VisibleString (FROM («a»..«z»|«A»..«Z»|«-. ») ^ SIZE (1..64, . . . ))
```

A.3.2 Описание ACH. 1 значения записи

Далее формально, с использованием ACH. 1, описано значение персональной записи для Джона Смита (John Smith).

```
{name {givenName «John», initial «P», familyName «Smith»},
 title          «Director»,
 number         51,
 dateOfHire     «19710917»,
 nameOfSpouse {givenName «Mary», initial «T», familyName «Smith»}, children
   {{name{givenName «Ralph», initial «T», familyName «Smith»}, dateOfBirth «19571111»},
   {name{givenName «Susan, initial «B», familyName «Jones»}, dateOfBirth «19590717», sex female}}}
```

А.3.3 Представление PER (ALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта ALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями.

Длина этого кодирования составляет 83 октета. Для сравнения, то же самое значение PersonnelRecord, закодированное с использованием варианта UNALIGNED PER, занимает 65 октетов, BER с определенной формой длины — по крайней мере 139 октетов, а BER с неопределенной формой длины — по крайней мере 164 октета.

А.3.3.1 *Шестнадцатеричное представление*

40C04A6F 686E5008 536D6974 68000033 08446972 6563746F 72001971 0917034D 61727954 08536D69 74680100 52616C70 68540853 6D697468 00195711 11820053 7573616E 42084A6F 6E657300 19590717 010140

А.3.3.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец строки; для выделения символов в символьной строке использован пробел; 'x' представляет нулевой бит заполнения, который иногда используется для выравнивания полей по границе октета.

```
0                       В PersonnelRecord нет значений расширения
1                       Битовая карта = 1 указывает, что есть «children»

0                       В «name» нет значений расширения

0                       Длина находится в диапазоне корня расширения
0000 11xxxxxx           Длина name. givenName = 4
01001010 01101111 01101000 01101110 name. givenName = «John»

01010000                name. initial = «P»

0                       Длина находится в диапазоне корня расширения
000100x                 длина name. familyName = 5
01010011 01101101 01101001 01110100 01101000 name. familyName = «Smith»

0xxxxxxx                Значение находится в диапазоне корня расширения
00000000 00110011       number = 51

00001000                Длина title = 8
01000100 01101001 01110010 01100101 01100011 01110100 01101111 01110010 title = «Director»

0xxxxxxx                Длина находится в диапазоне корня расширения
0001 1001 0111 0001 0000 1001 0001 0111 dateOfHire = «19590717»

0                       В nameOfSpouse нет значений расширения

0                       Длина находится в диапазоне корня расширения
000011                  Длина nameOfSpouse.givenName = 4
01001101 01100001 01110010 01111001 nameOfSpouse.givenName = «Mary»

01010100                nameOfSpouse. initial = «T»

0                       Длина находится в диапазоне корня расширения
000100x                 Длина nameOfSpouse. familyName = 5
01010011 01101101 01101001 01110100 01101000 nameOfSpouse. familyName = «Smith»

0                       Количество «children» находится в диапазоне корня расширения

0                       В children [0] нет значений расширения

0                       В children [0]. name нет значений расширения

0                       Длина находится в диапазоне корня расширения
000100xx xxxx           Длина children [0]. givenName = 5
01010010 01100001 01101100 01110000 01101000 children [0]. givenName = «Ralph»

01010100                children [0]. initial = «T»

0                       Длина находится в диапазоне корня расширения
000100x                 Длина children [0]. familyName = 5
01010011 01101101 01101001 01110100 01101000 children [0]. familyName = «Smith»

0xxxxxxx                Длина находится в диапазоне корня расширения
0001 1001 0101 0111 0001 0001 0001 0001 children [0]. dateOfBirth = «19571111»
```

1 В children [1] есть значение (я) расширения

0 В children [1]. name нет значений расширения

0 Длина находится в диапазоне корня расширения
00010 0xxxxxxx Длина children [1]. givenName = 5
01010011 01110101 01110011 01100001 01101110 children [1]. givenName = «Susan»

01000010 children [1]. initial = «B»

0 Длина находится в диапазоне корня расширения
000100x Длина children [1]. familyName = 5
01001010 01101111 01101110 01100101 01110011 children [1]. familyName = «Jones»

0xxxxxxx Длина находится в диапазоне корня расширения
0001 1001 0101 1001 0000 0111 0001 0111 children [1]. dateOfBirth = «19590717»

0000000 Длина битовой карты расширяющего дополнения для children [1] = 1

1 Указывает, что есть значение расширения для «sex»

00000001 Длина полного кодирования «sex»
01xxxxxx Полное кодирование «sex» = female

А.3.4 Представление PER (UNALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта UNALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями. Биты заполнения не встречаются в варианте UNALIGNED, а символы кодируются в минимально возможное число битов.

Длина этого кодирования составляет 65 октетов. Для сравнения, то же самое значение PersonnelRecord, закодированное с использованием варианта ALIGNED PER, занимает 83 октета, BER с определенной формой длины — по крайней мере 139 октетов, а BER с неопределенной формой длины — по крайней мере 164 октета.

А.3.4.1 *Шестнадцатеричное представление*

40CBAA3A 5108A512 5F180330 889A7965 C7D37F20 CB8848B8 19CE5BA2 A114A24B E3011372 7AE35422 94497C61 95711118 22985CE5 21842EAA 60B832B2 0E2E0202 80

А.3.4.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец строки; для выделения символов в символьной строке использован пробел; точка (.) отмечает границу октета; 'x' представляет нулевой бит, использованный для заполнения последнего октета до границы октета.

0 В PersonnelRecord нет значений расширения
1 Битовая карта = 1 указывает, что есть «children»

0 В «name» нет значений расширения

0 Длина находится в диапазоне корня расширения
0000.11 Длина name. givenName = 4
001011 .101010 10.0011 1010.01 name. givenName = «John»

010001 name.initial = «P»

.0 Длина находится в диапазоне корня расширения
000100 длина name. familyName = 5
0.10100 101.000 10010.0 101111 1.00011 name. familyName = «Smith»

0 Значение находится в диапазоне корня расширения
00. 00000011.0011 number = 51

0000.1000 Длина title = 8
1000.100 11010.01 111001.0 1100101 1100011 1.110100 11.01111 111.0010 title = «Director»

0 Длина находится в диапазоне корня расширения
000.1 1001 011.1 0001 000.0 1001 000.1 0111 dateOfHire = «19590717»

0 В nameOfSpouse нет значений расширения

0 Длина находится в диапазоне корня расширения
0.00011 Длина nameOfSpouse. givenName = 4

```
001.110 01110.0 101101 1.10100 nameOfSpouse. givenName = «Mary»
010.101                        nameOfSpouse. initial = «T»
0                              Длина находится в диапазоне корня расширения
0001.00                        Длина nameOfSpouse. familyName = 5
010100 .101000 10.0100 1011.11 100011 nameOfSpouse. familyName = «Smith»
.0                             Количество «children» находится в диапазоне корня расширения
0                              В children [0] нет значений расширения
0                              В children [0]. name нет значений расширения
0                              Длина находится в диапазоне корня расширения
0001.00                        Длина children [0]. givenName = 5
010011 .011100 10.0111 1010.11 100011 children [0]. givenName = «Ralph»
.010101                        children [0]. initial = «T»
0                              Длина находится в диапазоне корня расширения
0.00100                        Длина children [0]. familyName = 5
010.100 10100.0 100100 1.01111 100.011 children [0]. familyName = «Smith»
0                              Длина находится в диапазоне корня расширения
0001 .1001 0101 .0111 0001 .0001 0001 .0001 children [0]. dateOfBirth = «19571111»
1                              В children [1] есть значение (я) расширения
0                              В children [1]. name нет значений расширения
0                              Длина находится в диапазоне корня расширения
0.00100                        Длина children [1]. givenName = 5
010.100 11000.0 101110 0.11100 101.001 children [1]. givenName = «Susan»
00001.1                        children [1]. initial = «B»
0                              Длина находится в диапазоне корня расширения
000100                         Длина children [1]. familyName = 5
.001011 10.1010 1010.01 100000 .101110 children [1]. familyName = «Jones»
0                              Длина находится в диапазоне корня расширения
0.001 1001 0.101 1001 0.000 0111 0.001 0111 children [1]. dateOfBirth = «19590717»
0.000000                       Длина битовой карты расширяющего дополнения для children [1] = 1
1                              Указывает, что есть значение расширения для «sex»
0.0000001                      Длина полного кодирования «sex»
0.1xxxxxx                      Полное кодирование «sex» = female
x                              Бит заполнения для создания полного кодирования PersonnelRecord
```

А.4 Запись, которая использует расширяющие дополнительные группы

А.4.1 Описание АСН. 1 структуры записи

Ниже формально описана структура гипотетической записи покупателя с использованием АСН. 1. специфицированной в ГОСТ Р ИСО/МЭК 8824-1 для определения типов. Принята среда AUTOMATIC TAGS.

```
Ax ::= SEQUENCE {
    a INTEGER (250..253),
    b BOOLEAN,
    c CHOICE {
      d INTEGER,
      ...,
      [[
        e BOOLEAN,
        f IA5String
      ]],
      ...
    },
    ....
```

```
        ||
          g NumericString (SIZE (3)),
          h BOOLEAN OPTIONAL
        ]],
        ...,
          i BMPString OPTIONAL,
          j PrintableString OPTIONAL
}
```

А.4.2 Описание АСН. 1 значения записи

Ниже значение Ax формально описано с использованием АСН. 1:

{a 253, b TRUE, c e: TRUE, g «123», h FALSE}

А.4.3 Представление PER (ALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта ALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями. В двоичном виде 'x' используется для представления битов заполнения, которые кодируются как нулевые; они используются для выравнивания полей.

Длина этого кодирования составляет 8 октетов. Для сравнения, то же самое значение, закодированное с использованием варианта UNALIGNED PER, занимает 8 октетов, BER с определенной формой длины — по крайней мере 22 октета, а BER с неопределенной формой длины — по крайней мере 26 октет.

А.4.3.1 *Шестнадцатеричное представление*

9E000180 010291A4

А.4.3.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец строки; для выделения символов в символьной строке использован пробел; 'x' представляет нулевой бит заполнения, который иногда используется для выравнивания полей по границе октета.

1	В Ax есть дополнительные расширяющие значения
00	Битовая карта = 0 указывает, что факультативные поля (i, j) отсутствуют
11	a = 253
1	b = TRUE
1	Выбор значения c является дополнительным расширяющим значением
0000000 xx	Выбрано c. e
00000001	Длина c. e
1xxxxxxx	c. e = TRUE
0000000	Количество расширяющих дополнений, определенных в Ax, = 1
1	Первое расширяющее дополнение присутствует
00000010	Длина кодирования расширяющего дополнения = 2
1	Битовая карта = 1 указывает, что h присутствует
0010 0011 0100	g = «123»
1xx	h = TRUE

А.4.4 Представление PER (UNALIGNED) данного значения записи

Далее показано представление приведенного выше значения записи (после применения варианта UNALIGNED установленных в настоящем стандарте правил уплотненного кодирования). Кодирование представлено в шестнадцатеричном виде и сопровождается двоичным видом с описательными комментариями. Биты заполнения не встречаются в варианте UNALIGNED, за исключением, возможно, конца кодирования самого внешнего значения.

Длина этого кодирования составляет 8 октетов. Для сравнения, то же самое значение, закодированное с использованием варианта ALIGNED PER, занимает 8 октетов, BER с определенной формой длины — по крайней мере 22 октета, а BER с неопределенной формой длины — по крайней мере 26 октет.

А.4.4.1 *Шестнадцатеричное представление*

9E000600 040A4690

А.4.4.2 *Двоичное представление*

Для облегчения чтения данных в двоичном представлении использованы пустые строки для группировки логически связанных полей (обычно это пары длина/значение); для разделения полей использован конец

строки; для выделения символов в символьной строке использован пробел; точка (.) отмечает границу октета; 'x' представляет нулевой бит, использованный для заполнения последнего октета до границы октета.

1	В Ax есть дополнительные расширяющие значения
00	Битовая карта = 0 указывает, что факультативные поля (i, j) отсутствуют
11	a = 253
1	b = TRUE
1	Выбор значения c является дополнительным расширяющим значением
0.000000	Выбрано c. e
00.000001	Длина c. e
1x.xxxxxx	c. e = TRUE
00.00000	Количество расширяющих дополнений, определенных в Ax, = 1
1	Первое расширяющее дополнение присутствует
00.000010	Длина кодирования расширяющего дополнения = 2
1	Битовая карта = 1 указывает, что h присутствует
0.010 0011 0.100	g = «123»
1xxxx	h = TRUE

ПРИЛОЖЕНИЕ В
(справочное)

Объединения видимых для PER ограничений

Некоторые свойства могут оставаться видимыми при объединении элементов подтипа, каждый из которых по отдельности может быть видимым для PER. Ниже приведены примеры таких свойств:

B.1 Эффективным ограничением размера для

A :: = IA5String (SIZE (1..4) | SIZE (9..10))

является

A :: = IA5String (SIZE (1..4) | (9..10))

B.2 Когда ограничение PermittedAlphabet объединяется в одну спецификацию подтипа с другими ограничениями PermittedAlphabet, нет эффективного ограничения PermittedAlphabet, содержащего меньше, чем все символы в неограниченном типе, если нет единственной спецификации PermittedAlphabet, которая является супермножеством всех других спецификаций PermittedAlphabet в этой спецификации подтипа. Кроме того, если в эти спецификации ограничений включены ограничения размера, то супермножество спецификации PermittedAlphabet должно иметь эффективное ограничение размера, объединенное с ним операцией INTERSECTION, которое является супермножеством всех других ограничений размера, наложенных на тип. Например,

```
B :: = IA5String (FROM («AB») ^ SIZE (1..2) |
                  FROM («DE») ^ SIZE (3) |
                  FROM («ABCDE») ^ SIZE (1..5))
```

имеет эффективное ограничение размера и эффективное ограничение алфавита PermittedAlphabet

B :: = IA5String (FROM («ABCDE») ^ (SIZE (1..5))

так как это является супермножеством более сложного выражения, приведенного выше, таким образом оно становится видимым для PER. С другой стороны, в следующем примере эффективное ограничение PermittedAlphabet является полным набором символов, допустимых для IA5String, так как нет единственного эквивалента ограничения PermittedAlphabet. Таким образом, следующее ограничение не является видимым для PER:

C :: = IA5String (FROM («AB») | (FROM («CD»)) -- не эквивалентно (FROM («ABCD»))

B.3 Ограничения размера могут произвольно объединяться, если нет ограничения PermittedAlphabet. Например,

E :: = IA5String (SIZE (1..4) | (SIZE (5..10) ^ FROM («ABCD») | SIZE (6..10))

является невидимым для PER ограничением размера (так как размер 5 допустим не для всех возможных символов), тогда как

E :: = IA5String (SIZE (1..4) | (SIZE (6..10) ^ FROM («ABCD») | SIZE (6..10))

является видимым для PER ограничением размера SIZE (1..4) | (6..10), так как эффективное ограничение PermittedAlphabet является набором всех символов IA5String. В этом случае FROM («ABCD») не является видимым для PER ограничением, так как оно неприменимо ко всем возможным значениям E (например, если длина строки равна 1, то символ не ограничивается одним из «ABCD»).

ПРИЛОЖЕНИЕ С
(справочное)

Поддержка алгоритмов PER

Прикладной стандарт или функциональный профиль могут определить, какие из правил уплотненного кодирования должны поддерживаться, а соответствующие синтаксисы передачи должны предлагаться или приниматься при согласовании.

Когда есть требования использовать надежно передающее и/или каноническое кодирование в EMBEDDED PDV (или в EXTERNAL) либо в CHARACTER STRING, то они должны быть явно сформулированы.

Ниже дано руководство по разработке нормативных текстов.

С.1 Каноническое кодирование предназначено для использования, когда возможности защиты применяются к кодированию (см. ГОСТ Р ИСО/МЭК 8825-1, приложение D). Использование CANONICAL-PER может повлечь за собой существенные дополнительные издержки использования центрального процессора, когда кодируемое значение содержит тип «множество-из», и, вообще говоря, не рекомендуется для протоколов, если возможности защиты не требуются.

С.2 Когда значение абстрактного синтаксиса содержит вложенные данные, которые закодированы с использованием синтаксиса передачи или абстрактного синтаксиса, отличного от связанного со значением абстрактного синтаксиса, настоятельно рекомендуется, чтобы для вложенных данных использовалось надежно передающее кодирование. Если существенны возможности защиты, то потребуются канонические правила кодирования. В данном контексте особое внимание следует уделить уровню ИСО/МЭК 10646-1, который должен использоваться для типа BMPString или UniversalString, так как только реализация уровня 1 ИСО/МЭК 10646-1 будет гарантированно канонической.

С.3 Когда контекст представления устанавливается для одностороннего потока данных, настоятельно рекомендуется, чтобы данные были закодированы надежно передающим способом.

С.4 Когда контекст представления устанавливается для двустороннего потока данных, применение BASIC-PER может дать существенные преимущества гибкости и экономии.

С.5 Настоятельно рекомендуется, чтобы все реализации, поддерживающие декодирование какого-либо варианта ALIGNED PER синтаксиса передачи, поддерживали декодирование BASIC-PER, вариант ALIGNED (и, следовательно, CANONICAL-PER, вариант ALIGNED) и принимали контексты представления, идентифицированные с любым из этих двух правил кодирования, при условии, что контексты для получения данных устанавливаются этой реализацией. Аналогичная рекомендация справедлива для варианта UNALIGNED.

С.6 В интересах обеспечения взаимодействия рекомендуется, чтобы все реализации PER поддерживали как вариант ALIGNED, так и вариант UNALIGNED (дополнительное усложнение реализации невелико). Какой из них (или оба) предоставляется для конкретного сеанса связи, является локальным вопросом управления, как и вопрос о том, какой из них принимается, если предоставлены оба. Если предложен только один вариант, то он должен быть принят.

С.7 Следование настоящим рекомендациям важно, в частности, для поставщиков средств общего назначения. Когда реализация является специфической для некоторого конкретного приложения, то поддержка единственного синтаксиса передачи PER (возможно, определенного прикладным разработчиком) может быть вполне допустимой.

ПРИЛОЖЕНИЕ D
(справочное)

Поддержка правил расширения ACH. 1

D.1 Настоящие правила уплотненного кодирования зависят от полного определения типа, к которому они применяются. В общем случае, любые изменения в определении типа, отличные от чисто синтаксических, будут влиять на кодирование всех значений, использующих соответствующую часть спецификации. В частности, добавление новых факультативных компонентов к последовательности, преобразование компонента в тип CHOICE из компонента этого и некоторого другого типа, ослабление или ужесточение ограничений на некоторый компонент, вероятно, изменят кодирования значений типа.

D.2 Однако настоящие правила кодирования гарантируют, что требования к правилам кодирования, установленные в модели расширения типа ACH. 1 (см. ГОСТ Р ИСО/МЭК 8824-1), удовлетворяются.

D.3 Когда тип не является частью последовательности расширения (нет маркера расширения), то, в соответствии со сказанным выше, PER не обеспечивают поддержку расширения этого типа. Когда тип «последовательность» или «множество» имеет маркер расширения, но расширяющих дополнений нет, есть один дополнительный бит (который в вариантах ALIGNED может стать одним октетом из-за заполнения) по сравнению с тем же самым типом без маркера расширения. Когда в типе присутствуют дополнения и они фактически передаются в сеансе связи, есть дополнительные издержки приблизительно в один октет, плюс дополнительное поле длины для каждого передаваемого расширяющего дополнения по сравнению с тем же самым типом без маркера расширения.

D.4 Важно иметь в виду, что добавление или удаление маркера расширения изменяет биты в строки и, в общем случае, потребует изменения номера версии протокола.

D.5 Кодирование не изменяется из-за включения маркера расширения в набор информационных объектов из-за добавления или удаления спецификаций исключений, но они могут потребовать изменений в поведении реализации и изменения номера версии протокола.

ПРИЛОЖЕНИЕ E
(справочное)

Руководство по сцеплению кодирований PER

E.1 Кодирования PER являются саморазграниченными при знании правил и типа кодирования. Полные кодирования для вариантов ALIGNED и UNALIGNED всегда кратны 8 битам.

E.2 Для передачи кодирований PER в протоколе уровня представления ВОС, кодирования вариантов ALIGNED и UNALIGNED могут быть сцеплены в строку октетов.

ПРИЛОЖЕНИЕ F
(справочное)

Присвоенные значения идентификаторов объектов

В настоящем стандарте присвоены следующие значения идентификаторов и описателей объектов:

Для BASIC-PER, вариант ALIGNED:
{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) aligned (0)}
«Packed encoding of a single ASN. 1 type (basic aligned)»

Для BASIC-PER, вариант UNALIGNED:
{joint-iso-itu-t asn1 (1) packed-encoding (3) basic (0) unaligned (1)}
«Packed encoding of a single ASN. 1 type (basic unaligned)»

Для CANONICAL-PER, вариант ALIGNED:
{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) aligned (0)}
«Packed encoding of a single ASN. 1 type (canonical aligned)»

Для CANONICAL-PER, вариант UNALIGNED:
{joint-iso-itu-t asn1 (1) packed-encoding (3) canonical (1) unaligned (1)}
«Packed encoding of a single ASN. 1 type (canonical unaligned)»

УДК 681.324:006.354 ОКС 35.100.60 П85 ОКСТУ 4002

Ключевые слова: обработка данных, информационный обмен, сетевое взаимодействие, взаимосвязь открытых систем, коммуникационная процедура, преобразование данных, кодирование, правила (инструкции).

Редактор *Р.Г. Говердовская*
Технический редактор *О.Н. Власова*
Корректор *В.И. Кануркина*
Компьютерная верстка *И.А. Налейкиной*

Изд. лиц. № 02354 от 14.07.2000. Сдано в набор 22.05.2003. Подписано в печать 27.06.2003. Усл. печ.л. 5,12. Уч.-изд.л. 5,10. Тираж 262 экз. С 10992. Зак. 544.

ИПК Издательство стандартов, 107076 Москва, Колодезный пер., 14.
http://www.standards.ru e-mail: info@standards.ru
Набрано в Издательстве на ПЭВМ
Филиал ИПК Издательство стандартов — тип. «Московский печатник», 105062 Москва, Лялин пер., 6.
Плр № 080102