ФЕДЕРАЛЬНОЕ АГЕНТСТВО
ПО ТЕХНИЧЕСКОМУ РЕГУЛИРОВАНИЮ И МЕТРОЛОГИИ

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГОСТ Р ИСО/ТС 10303-26— 2015

Системы автоматизации производства и их интеграция

ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ

Часть 26
Методы реализации. Двоичное представление данных, определенных на языке EXPRESS

ISO/TS 10303-26:2011
Industrial automation systems and integration – Product data representation and exchange – Part 26: Implementation methods: Binary representation of EXPRESS-driven data
(IDT)

Издание официальное

Москва
Стандартинформ
2016

Предисловие

1 ПОДГОТОВЛЕН Федеральным государственным автономным научным учреждением «Центральный научно-исследовательский и опытно-конструкторский институт робототехники и технической кибернетики» (ЦНИИ РТК) на основе собственного аутентичного перевода на русский язык международного документа, указанного в пункте 4

2 ВНЕСЕН Техническим комитетом по стандартизации ТК 459 «Информационная поддержка жизненного цикла изделий»

3 УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Приказом Федерального агентства по техническому регулированию и метрологии от 21 июля 2015 г. № 926-ст

4 Настоящий стандарт идентичен международному документу ИСО/ТС 10303-26:2011 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 26. Методы реализации. Двоичное представление данных, определенных на языке EXPRESS» (ISO/TS 10303-26:2011 «Industrial automation systems and integration – Product data representation and exchange – Part 26: Implementation methods: Binary representation of EXPRESS-driven data»).

При применении настоящего стандарта рекомендуется использовать вместо ссылочных международных стандартов соответствующие им национальные стандарты Российской Федерации, сведения о которых приведены в дополнительном приложении ДА

5 ВВЕДЕН ВПЕРВЫЕ

Правила применения настоящего стандарта установлены в ГОСТ Р 1.0—2012 (раздел 8). Информация об изменениях к настоящему стандарту публикуется в ежегодном (по состоянию на 1 января текущего года) информационном указателе «Национальные стандарты», а официальный текст изменений и поправок – в ежемесячном информационном указателе «Национальные стандарты». В случае пересмотра (замены) или отмены настоящего стандарта соответствующее уведомление будет опубликовано в ближайшем выпуске информационного указателя «Национальные стандарты». Соответствующая информация, уведомление и тексты размещаются также в информационной системе общего пользования – на официальном сайте Федерального агентства по техническому регулированию и метрологии в сети Интернет (www.gost.ru)

Содержание

Введение

Стандарты комплекса ИСО 10303 распространяются на компьютерное представление информации об изделиях и обмен данными об изделиях. Их целью является обеспечение нейтрального механизма, способного описывать изделия на всем протяжении их жизненного цикла. Этот механизм применим не только для обмена файлами в нейтральном формате, но является также основой для реализации и совместного доступа к базам данных об изделиях и организации архивирования.

Стандарты комплекса ИСО 10303 представляют собой набор отдельно издаваемых стандартов (частей). Стандарты данного комплекса относятся к одной из следующих тематических групп: «Методы описания», «Методы реализации», «Методология и основы аттестационного тестирования», «Интегрированные обобщенные ресурсы», «Интегрированные прикладные ресурсы», «Прикладные протоколы», «Комплекты абстрактных тестов», «Прикладные интерпретированные конструкции» и «Прикладные модули». Полный перечень стандартов комплекса ИСО 10303 представлен на сайте http://www.tc184-sc4.org/titles/STEP_Titles.htm. Настоящий стандарт входит в тематическую группу «Методы реализации». Он подготовлен подкомитетом SC4 «Производственные данные» Технического комитета 184 ИСО «Системы автоматизации производства и их интеграция».

Настоящий стандарт определяет связь языка EXPRESS с Иерархическим форматом данных, версия 5 (HDF5). HDF5 первоначально был разработан Национальным центром суперкомпьютерных приложений Иллинойского университета, а в настоящее время поддерживается некоммерческой организацией Группа HDF. Программное обеспечение HDF включает библиотеки ввода-вывода и средства анализа, визуализации и конвертирования научных данных.

Настоящий стандарт определяет отображение экземпляров данных, определенных на языке EXPRESS, на формат файла HDF5. Однако настоящий стандарт не отображает все конструкции языка моделирования EXPRESS на HDF5. HDF5 не является языком моделирования, но файл HDF5 содержит в себе типы данных, определяющие включенные в данный файл экземпляры данных.

Пользователи настоящего стандарта должны обладать детальными знаниями понятий языка EXPRESS, определенного в ИСО 10303-11, и HDF5. В приведенных примерах программ использован прикладной программный интерфейс HDF5.

НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Системы автоматизации производства и их интеграция
ПРЕДСТАВЛЕНИЕ ДАННЫХ ОБ ИЗДЕЛИИ И ОБМЕН ЭТИМИ ДАННЫМИ
Часть 26
Методы реализации.
Двоичное представление данных, определенных на языке EXPRESS

Industrial automation systems and integration. Product data representation and exchange. Part 26. Implementation methods. Binary representation of EXPRESS-driven data

Дата введения — 2016—10—01

1 Область применения

Настоящий стандарт определяет отображение конструкций данных, определенных на языке EXPRESS, на формат двоичного файла, соответствующего Иерархическому формату данных, версия 5 (HDF5).

Требования настоящего стандарта не распространяются на:

- отображение ограничений и правил, определенных на языке EXPRESS, на файл HDF5;
- отображение конкретных прикладных программных интерфейсов EXPRESS-схем на данные, определенные на языке EXPRESS.

2 Нормативные ссылки

В настоящем стандарте использованы ссылки на следующие международные стандарты и документы (для датированных ссылок следует использовать только указанное издание, для недатированных ссылок – последнее издание указанного документа, включая все поправки к нему):

ИСО 639-2 Коды для представления названий языков. Часть 2. Код Alpha-3 (ISO 639-2, Codes for the representation of names of languages – Part 2: Alpha-3 code)

ИСО 8601 Элементы данных и форматы обмена. Обмен информацией. Представление дат и времени (ISO 8601, Data elements and interchange formats – Information interchange – Representation of dates and times)

ИСО 10303-1 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы (ISO 10303-1, Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles)

ИСО 10303-11 Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS (ISO 10303-11, Industrial automation systems and integration – Product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual)

Спецификация формата файлов HDF5. Группа HDF. Доступна в Интернете по адресу: http://www.hdfgroup.org/HDF5/doc/H5.format.html (HDF5 File Format Specification. HDF Group. Available from Workd Wide Web: http://www.hdfgroup.org/HDF5/doc/H5.format.html)

3 Термины и определения

В настоящем стандарте применены следующие термины с соответствующими определениями:

3.1 **прикладной протокол** (application protocol): Часть настоящего комплекса стандартов, которая определяет прикладную интерпретированную модель, удовлетворяющую области применения и информационным требованиям к конкретному приложению.

[ИСО 10303-1:1994, статья 3.2.7]

3.2 **данные** (data): Представление информации в формальном виде, пригодном для передачи, интерпретации или обработки людьми или компьютерами.

[ИСО 10303-1:1994, статья 3.2.14]

3.3 **язык определения данных** (data specification language): Набор правил для определения данных и их взаимосвязей, пригодный для компьютерной передачи, интерпретации или обработки.

Издание официальное

[ИСО 10303-1:1994, статья 3.2.16]

3.4 **данные, определенные на языке EXPRESS** (EXPRESS-driven data): Набор экземпляров объектов, определенных в EXPRESS-схеме.

3.5 **совокупность** (population): Множество экземпляров объектного типа данных.

[ИСО 10303-11:2004, статья 3.3.16]

4 Сокращения

В настоящем стандарте применены следующие сокращения:

ПП – прикладной протокол (application protocol; AP);

ТБ – тепловой баланс (thermal balance; TB);

ЯОД – язык определения данных (data definition language; DDL);

API – интерфейс прикладного программирования (application programming interface);

HDF5 – Иерархический формат данных, версия 5 (Hierarchical Data Format Version 5);

I/O – ввод и вывод (input and output).

5 Соответствие

Отображение данных, определенных на языке EXPRESS, на формат файла HDF5 не зависит от интерфейса прикладного программирования, используемого при работе с файлом HDF5.

Примечание — Редакции 1.6.5 и 1.8.0 интерфейса прикладного программирования HDF5 поддерживаются данным отображением. Редакция 1.8.0 содержит новые конструкции, которые не используются в данном отображении.

Файл HDF5 соответствует настоящему стандарту в том случае, если:

- он соответствует определению формата файла HDF5;
- он основан на допустимой EXPRESS-схеме, которая может иметь интерфейсы с другими допустимыми схемами;
- данные, определенные на языке EXPRESS, представлены в файле HDF5 так, как это определено в настоящем стандарте.

Соответствие препроцессоров и постпроцессоров обработки данных находится вне области применения настоящего стандарта.

6 Отображение данных, определенных на языке EXPRESS, на формат файла HDF5

6.1 Общие положения

В данном разделе определены отображения данных, определенных на языке EXPRESS, на Иерархический формат данных, версия 5 (HDF5). В данном разделе использовано соглашение, что заданы отображаемые понятия языка EXPRESS и определены соответствующие понятия HDF5. Отображение не определено относительно какого-либо конкретного интерфейса прикладного программирования (API). Реализации HDF5 доступны на многих языках программирования.

Примечание — В настоящем стандарте в примерах использован API HDF5, разработанный в Группе HDF5.

На рисунке 1 показаны условные обозначения, использованные в настоящем стандарте на диаграммах, для того чтобы пояснять понятия HDF5, встречающиеся в соответствующем файле HDF5.

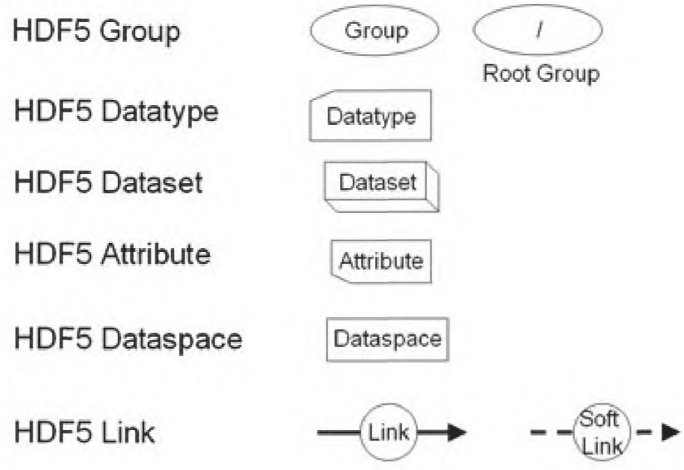

Рисунок 1 — Условные обозначения на диаграммах HDF5

Пример — На рисунке 2 показан пример обозначений для представления понятий HDF5, использованных в настоящем стандарте.

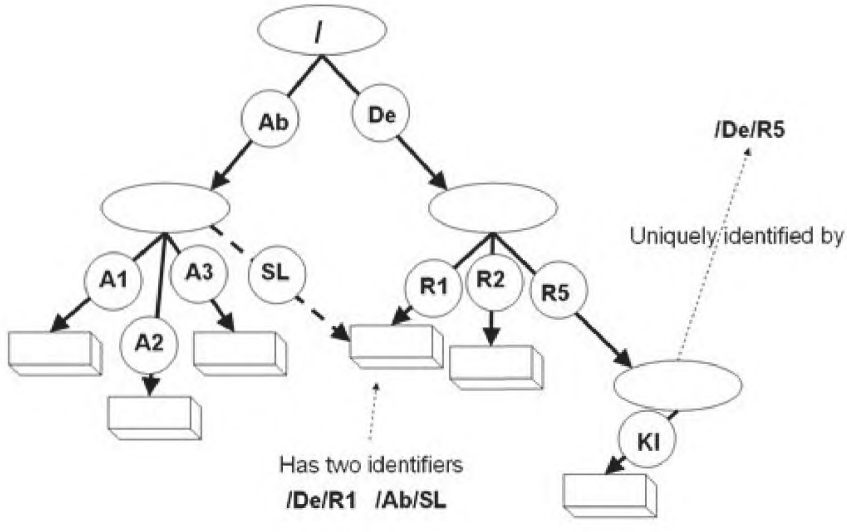

Рисунок 2 – Условные обозначения на диаграммах HDF5

6.2 Неотображаемые понятия языка EXPRESS

Следующие понятия языка EXPRESS в настоящем стандарте не отображаются на HDF5, и поэтому их представление не должно присутствовать в файле HDF5:

- объявления RULE;
- правила определения областей в объявлениях ENTITY и TYPE;
- правила UNIQUE в объявлениях ENTITY;

- объявления SUPERTYPE;
- объявления FUNCTION;
- объявления PROCEDURE;
- объявления CONSTANT, за исключением объявлений CONSTANT для экземпляров объектов;
- производные атрибуты и явно определенные атрибуты, переобъявленные как производные атрибуты;
- инверсные атрибуты;
- спецификации интерфейсов на языке EXPRESS;
- комментарии.

Примечание — Алгоритмические ограничения и значения атрибутов языка EXPRESS, являющиеся результатом оценочных вычислений, не включены в файл HDF5, отображающий экземпляры объектов языка EXPRESS. Поэтому проверка на основе EXPRESS-схемы достоверности файла HDF5 на соответствие настоящему стандарту потребует доступа к EXPRESS-схеме.

6.3 Общие требования к отображению и использованный подход

6.3.1 Использованный подход

Язык EXPRESS является языком определения данных. Схема в языке EXPRESS задает область определения, а также раздел, к которому относятся объявления. Поэтому в настоящем стандарте для определения контекста данной реализации должна быть выбрана единственная схема. Однако это не означает, что контекстные схемы, содержащие спецификации интерфейсов не поддерживаются данным отображением. Спецификации интерфейсов на языке EXPRESS поддерживаются до тех пор, пока они не отображены непосредственно на файл HDF5. Что касается определения данных, то набор схем, видимых для данной контекстной схемы, определяет полный набор допустимых типов данных. В настоящем стандарте принято, что EXPRESS-схемы определены в целях ограничения применимости совокупностей данных. Множество совокупностей данных, основанных на одной и той же EXPRESS-схеме, может существовать и быть включено в один файл HDF5. Так как HDF5 поддерживает разные способы представления значений данных, определенных на языке EXPRESS, на основе одной и той же схемы, то для любой EXPRESS-схемы не требуется существование единственного представления. Один и тот же файл HDF5 может содержать совокупности данных EXPRESS-схемы, использующие разные ее представления. Информация о конструкциях HDF5, использованных для кодирования данных, определенных на языке EXPRESS, хранится в файле HDF5 вместе с самими данными, поэтому постпроцессоры могут запросить данный файл для нахождения способов декодирования. Подход, использованный в настоящем стандарте для представления данных, определенных на языке EXPRESS, с помощью HDF5, основан на следующих предположениях:

- для обеспечения соответствия требованиям к производительности и размеру файла необходимо максимально использовать структуры HDF5, реализующие его оптимизационные возможности;
- определены только типы данных, используемые для записи данных на диск, но ничего не задано относительно представления этих данных в памяти;
- в тех случаях, когда это возможно, предпочтительно использовать предопределенные типы данных HDF5;
- должна поддерживаться межплатформенная функциональная совместимость.

Общий подход к представлению данных, определенных на языке EXPRESS, заключается в трактовке экземпляров объектных типов данных языка EXPRESS и экземпляров агрегированного типа данных языка EXPRESS одинаковым способом. Множество экземпляров объектного типа данных языка EXPRESS вместе с любыми неагрегированными значениями атрибутов трактуется как набор данных, основанный на составном типе данных HDF5. Агрегированные значения атрибутов языка EXPRESS представлены с помощью отдельного набора данных HDF5 для каждого агрегированного экземпляра. Кроме того, для экземпляров с небольшими агрегированными значениями определена возможность их непосредственного встраивания в набор данных HDF5, содержащий экземпляры объектного типа данных языка EXPRESS.

6.3.2 Имена связей HDF5

Группы, наборы данных и именованные типы данных, использованные в данном отображении, идентифицированы, или поименованы, с помощью связи HDF5. Имя, начинающееся с символа «косая черта» («/»), является абсолютным именем, доступ к которому начинается с корневой группы файла; все остальные имена являются относительными, а доступ к поименованному объекту начинается с указанной группы. Особым случаем является имя «/», которое относится к корневой группе.

Для конструкций HDF5, являющихся непосредственным отображением конструкций языка EXPRESS, идентификаторы языка EXPRESS используются как часть имен связей HDF5. Все идента-

фикаторы языка EXPRESS должны быть представлены символами верхнего регистра (прописными буквами). Символ «косая черта» («/») используется для того, чтобы отделить имена HDF5, являющиеся частью данной связи.

Примечание — Символ «точка» («.»), используемый как разделитель в идентификаторах языка EXPRESS, является зарезервированным символом в HDF5 и поэтому не используется для этой цели в файле HDF5.

Важно отметить, что в соответствии с определением HDF5 так же, как и в файловой системе UNIX, объекты HDF5 не имеют имен, так как имена ассоциированы со связями. Объект имеет идентификатор объекта, являющийся уникальным в данном файле, но один объект может иметь много имен, потому что может существовать много связей для данного объекта. Объект может быть снабжен альтернативным именем или перемещен в другую группу с помощью добавления или удаления связей. Но в данном случае сам объект никуда не перемещается, а его членство в группе не имеет никакого отношения к физическому размещению объекта в памяти.

Настоящий стандарт не накладывает никаких ограничений на дополнительные связи (т. е. имена), определяемые в файле для любого объекта HDF5, определенного на языке EXPRESS. Если данные совместно используются разными представлениями, определенными на языке EXPRESS, то настоящий стандарт не препятствует тому, чтобы один объект HDF5 имел связи, отображенные из нескольких EXPRESS-схем.

Пример — Следующий фрагмент на языке EXPRESS, отображенный на группу HDF5 с именем "pets_encoding", может в результате стать частью имени связи HDF5, относящейся к набору данных HDF5, – "pets_encoding/Dogs":

```
SCHEMA pets;
  ENTITY Dogs;
  END_ENTITY;
END_SCHEMA;
```

6.3.3 Представление совокупностей данных, определенных на языке EXPRESS, в HDF5

Каждая совокупность данных из EXPRESS-схемы представлена группой HDF5 (см. пример в С.2). Разные совокупности данных из одной EXPRESS-схемы могут появиться в одном файле HDF5, но при этом они будут принадлежать к разным группам HDF5.

Пример – На рисунке 3 показаны три группы HDF5, содержащие данные в соответствии с настоящим стандартом.

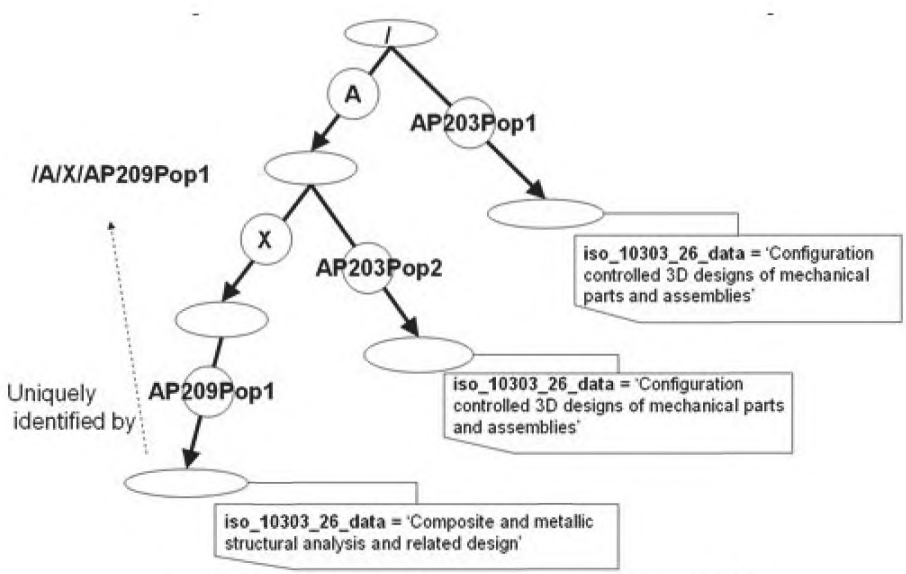

Рисунок 3 – Совокупности данных в группах HDF5

Данные в файле HDF5 базируются на связанных с ними типах данных HDF5. Совокупности данных, определенных на языке EXPRESS, в файле HDF5 базируются на типах данных HDF5, которые получены, по крайней мере частично, из лежащей в их основе EXPRESS-схемы. Эти типы данных присутствуют в группе HDF5, как это определено в 6.5.

Группа HDF5, представляющая совокупность данных из EXPRESS-схемы, должна иметь два связанных с ней атрибута HDF5 со следующими именами и описаниями:

- iso_10303-26_data – имеет значение <schema_id> и тип данных HDF5 STRING;
- iso_10303_26_data_set_names – содержит имена наборов данных родительских групп. В действительности данный атрибут представляет массив, содержащий имена объектов, соответствующие именам наборов данных. Реальные имена наборов данных получаются с использованием соглашений об именах, представленных в 6.10.

Атрибут HDF5 с именем “iso_10303-26_data” является индикатором для программного приложения, указывающим, что группа HDF5 содержит данные, закодированные в соответствии с настоящим стандартом.

Настоящий стандарт определяет также несколько необязательных атрибутов HDF5, предназначенных для дополнительного описания групп HDF5, содержащих совокупности данных, определенных на языке EXPRESS. Данные атрибуты относятся к типу данных HDF5 STRING и имеют следующие имена и описания:

- iso_10303-26_description – имеет значение <user_defined_description>;
- iso_10303-26_timestamp – имеет значение, соответствующее расширенному формату полной календарной даты, определенному в ИСО 8601, объединенному с расширенным форматом времени дня, также определенному в ИСО 8601. Дата и время должны быть разделены прописной буквой “T”, как это определено в ИСО 8601, в котором также определены альтернативные форматы, допускающие факультативное включение указателя часового пояса;
- iso_10303-26_author – имеет значение <user>;
- iso_10303-26_organization – имеет значение <user_organization>;
- iso_10303-26_originating_system – имеет значение <software_system_name>;
- iso_10303-26_preprocessor_version – имеет значение <software_application_and_version>;
- iso_10303-26_context – имеет значение <контекст, которому соответствуют данные>;
- iso_10303-26_language – имеет значение <язык, используемый по умолчанию для строковых значений>, где название языка должно быть закодировано с использованием библиографического кода Alpha-3, определенного в ИСО 639-2.

Настоящий стандарт не накладывает никаких ограничений на необязательные атрибуты HDF5, связанные с объектами HDF5 в файлах HDF5.

6.4 Отображение простых типов данных языка EXPRESS

Формат данных HDF5 поддерживает кодирование некоторых простых типов данных разными способами. Настоящий стандарт не требует определения конкретного способа кодирования для типов данных INTEGER, REAL или NUMBER языка EXPRESS.

Способ кодирования может быть запрошен из файла HDF5 постпроцессором. В таблице 1 определено представление значений простых и перечисляемых типов данных языка EXPRESS в HDF5 (см. раздел «Наборы данных HDF5» в [3]).

Таблица 1 – Отображение типов данных языка EXPRESS на HDF5

Тип данных языка EXPRESS	Представление в HDF5	Пример – Исходные типы данных API HDF5 (справочно)
INTEGER	HDF5 Standard 8, 16, 32 или 64 bits, Big- или Little-Endian	H5T_NATIVE_INT, H5T_NATIVE_LONG, H5T_NATIVE_LLONG
REAL	HDF5 IEEE Floating Point 32 или 64 bits, Big- или Little-Endian	H5T_NATIVE_FLOAT, H5T_NATIVE_DOUBLE, H5T_NATIVE_LDOUBLE
BOOLEAN	HDF5 ENUM из пар <string>:<integer> BOOLEAN-TRUE:1, BOOLEAN-FALSE:0	–
LOGICAL	HDF5 ENUM из пар <string>:<integer> LOGICAL-TRUE:1, LOGICAL-FALSE:0 и LOGICAL-UNKNOWN: –1	
STRING	H5T_STRING переменной длины	
BINARY	HDF5 OPAQUE, если задано FIXED или разрядность, иначе – HDF5 VARYING OPAQUE	
NUMBER	Аналогично представлению типа данных REAL языка EXPRESS	H5T_NATIVE_FLOAT, H5T_NATIVE_DOUBLE, H5T_NATIVE_LDOUBLE
ENUMERATION	HDF5 ENUM из пар <string>:<integer>, где string это <schema_group_name> + “/” + <enum_name> + “/” + <enum_literal>	–

Такое же кодирование для указанного типа данных языка EXPRESS может быть использовано во всей совокупности данных, содержащихся в файле HDF5. В этом случае в совокупности данных может быть использован необязательный атрибут для задания способа кодирования, который следует использовать для указанного типа данных языка EXPRESS. В зависимости от типа данных этот атрибут должен иметь следующее имя:

iso_10303_26_+ <data_type> + encoding

Значением данного атрибута должен быть тип данных HDF5, используемый для кодирования типа данных языка EXPRESS.

Пример – Атрибут iso_10303_26_real_encoding, имеющий значение “H5T_IEEE_F64LE”, указывает, что все значения типа данных REAL языка EXPRESS из данной совокупности должны кодироваться как Eight-byte, little-endian, IEEE floating point (восьмибайтовый код с плавающей запятой по IEEE, начиная с младшего байта).

Настоящий стандарт не требует использования какого-либо конкретного способа кодирования. Для простых случаев способ кодирования, используемый по умолчанию, определяется следующим образом:

- для представления типа данных INTEGER языка EXPRESS используется способ кодирования HDF5 32 bits, Signed, Little-Endian (например, H5T_NATIVE_LONG в API HDF5);
- для представления типов данных REAL и NUMBER языка EXPRESS используется способ кодирования HDF5 IEEE Floating Point 64 bits, Little-Endian (например, H5T_ NATIVE_DOUBLE в API HDF5);
- для представления типа данных ENUMERATION языка EXPRESS используются целые числа 1, 2, 3 и т.д.

6.5 Отображение объявлений и спецификаций интерфейсов из EXPRESS-схемы

Конкретная EXPRESS-схема является контекстом для любого отображения и называется далее контекстной схемой. Только объявления в контекстной схеме и объявления, видимые для данной контекстной схемы через спецификации интерфейсов на языке EXPRESS, должны быть основой для данных, определенных на языке EXPRESS.

Спецификации интерфейсов, объявленные в контекстной схеме, не отображаются на файл HDF5. Единственным исключением является случай, когда существует конфликт имен и имя внешней схемы требуется для разрешения данного конфликта. Для всех данных, записываемых в файл HDF5, требуется, чтобы соответствующий им тип (или типы) данных также был записан в файл HDF5. Поэтому для того, чтобы точно определить средства HDF5, используемые при кодировании совокупности данных на языке EXPRESS, все параметры EXPRESS-схемы также записываются в файл HDF5. Сама EXPRESS-схема и настоящий стандарт не требуют, чтобы закодированное представление всей

EXPRESS-схемы было включено в файл HDF5.

Примечание — Текстовое или любое другое представление самой EXPRESS-схемы может факультативно быть включено в файл HDF5 так же, как и многие другие типы данных (как атрибуты HDF5).

Параметры EXPRESS-схемы, закодированные в файле HDF5, должны быть определены в группе HDF5. Группа HDF5, представляющая EXPRESS-схему, должна быть расположена непосредственно под корневой группой (“/”). Имя группы, представляющей EXPRESS-схему, должно быть представлено следующим образом: <schema_name> + _encoding. Группа HDF5, содержащая схемы, может совместно использоваться совокупностями данных EXPRESS-схемы. Группа HDF5, содержащая представление необходимых параметров EXPRESS-схемы, должна иметь связанный с ней атрибут HDF5 с именем “iso_10303_26_schema” и значением <schema_id>. Атрибут HDF5 с именем “iso_10303_26_express_text” может дополнительно быть связан с группой HDF5 для обмена самим текстом на языке EXPRESS.

Пример — На рисунке 4 показана группа HDF5, содержащая именованные типы данных HDF5, которые относятся к EXPRESS-схеме, содержащей данную совокупность данных (т. е. группа HDF5 “Geometry_encoding” содержит определения типов данных для данных, закодированных на основании схемы “geometry”).

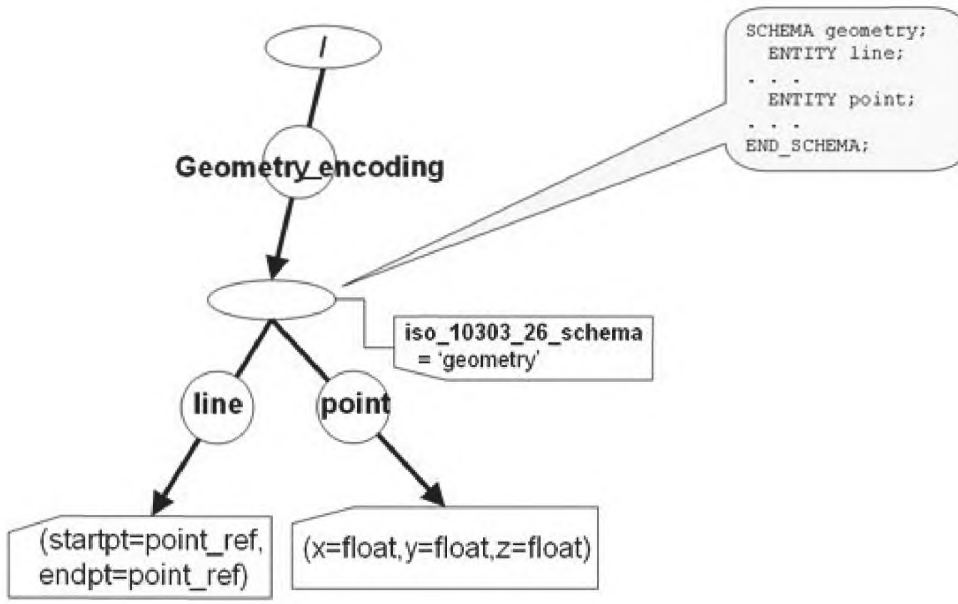

Рисунок 4 – Группа HDF5 с информацией о EXPRESS-схеме

Дополнительные примеры приведены в С.1 приложения С

6.6 Отображение объектных типов данных языка EXPRESS

В конкретном файле HDF5 должны быть отображены только те объектные типы данных языка EXPRESS, которые представляют данные, определенные на языке EXPRESS, содержащиеся в данном файле HDF5. Для каждого такого объектного типа данных языка EXPRESS в группе HDF5, представляющей данную EXPRESS-схему, должен присутствовать именованный тип данных HDF5, относящийся к составному типу данных HDF5. Относительное имя именованного типа данных HDF5 должно быть представлено следующим образом: <schema_group_name> + “/” + <entity_id>, либо следующим образом (см. 6.7):

```
<schema_group_name> + "/" + <entity_id> + ( "+" + <entity_id> )
```

В составном типе данных HDF5, представляющем объектный тип данных языка EXPRESS, должно присутствовать в качестве второго члена поле HDF5, основанное на типе данных INTEGER HDF5. Данное поле должно иметь имя “Entity-Instance-Identifier”. Данное поле должно содержать уни-

кальное целое число, идентифицирующее экземпляр объекта языка EXPRESS в рамках набора данных, содержащего полную совокупность экземпляров объектов языка EXPRESS, относящихся к конкретному объектному типу данных. Данное целое число не должно изменяться на протяжении всего существования данного экземпляра объекта языка EXPRESS.

Информация о представляемом объектном типе данных языка EXPRESS должна содержать информацию обо всех явно заданных атрибутах, включая все унаследованные явно заданные атрибуты. Для каждого явно заданного атрибута объектного типа данных языка EXPRESS, включая все унаследованные явно заданные атрибуты, должно быть создано поле (или объект) HDF5, относящееся к составному типу данных HDF5, которое представляет этот объектный тип данных.

Именем любого члена составного типа данных HDF5, представляющего атрибут языка EXPRESS, должно быть имя явно заданного атрибута языка EXPRESS, представленное символами верхнего регистра.

Типом данных любого члена составного типа данных HDF5 является тип данных HDF5, соответствующий области определения атрибута языка EXPRESS, как это установлено в 6.8.2 для простых типов данных, в 6.10.4 для идентификаторов экземпляров объектов, в 6.8.3 для типов данных ARRAY и в 6.8.4 для агрегированных типов данных.

Пример — Следующие объектные типы данных языка EXPRESS, относящиеся к представлению схемы "s_encoding":

```
SCHEMA s;
  ENTITY x;
    name : STRING;
  END_ENTITY;
  ENTITY y SUBTYPE OF (x);
    age : INTEGER;
  END_ENTITY;
END_SCHEMA;
```

приведут к следующим определениям HDF5:

- именованный тип данных HDF5 "s_encoding/x" определен одним полем HDF5 с именем "s_encoding/x/name";

- именованный тип данных HDF5 "s_encoding/y" определен двумя полями HDF5 с именами "s_encoding/y/name" и "s_encoding/y/age".

Настоящий стандарт не определяет отображение для производных или инверсных атрибутов, следовательно, они не должны присутствовать в соответствующем файле HDF5.

Примечание — Хотя в настоящем стандарте не определено представление производных или инверсных атрибутов, это не препятствует им присутствовать в файле HDF5. Кроме того, можно сохранить и даже транслировать на язык программирования в файле HDF5 выражение, определяющее значение атрибута, и вычислить данное значение, когда это потребуется.

В языке EXPRESS унаследованный атрибут может быть переобъявлен несколькими способами. Обработка этих способов при отображении осуществляется следующим образом:

- заданные в явном виде атрибуты языка EXPRESS, которые переобъявляют унаследованные атрибуты, отображаются на основе области определения атрибута в переобъявляющем подтипе данных;

- производные атрибуты языка EXPRESS, являющиеся переобъявлениями унаследованных заданных в явном виде атрибутов, не присутствуют в составном типе данных HDF5, представляющем подтип данных, в котором они переобъявлены;

- заданные в явном виде атрибуты языка EXPRESS, которые переименовывают переобъявленные унаследованные атрибуты, отображаются с использованием нового имени, определенного в переобъявляющем подтипе данных.

Так как в HDF5 нет понятия отсутствующего значения данных, то для этого в составной тип данных HDF5 вводится дополнительный член. Данный вспомогательный член является битовым образом, полученным из соответствующего объектного типа данных языка EXPRESS, который определяет состояние «1» или «0» (присутствует или отсутствует) для всех остальных членов составного типа данных, за исключением поля номер 2, которое является обязательным полем. Данное поле должно иметь тип данных H5T_INTEGER. Вспомогательный член должен быть первым в своем составном типе данных и иметь имя "set_unset_bitmap". Самый младший бит представляет первый атрибут языка EXPRESS (второй член в составном типе данных, начиная с нулевого). Следующий бит представ-

ляет второй атрибут языка EXPRESS и т. д.

Пример приведен в приложении С.3.

6.7 Отображение объявлений подтипов данных языка EXPRESS

В языке EXPRESS взаимосвязь между подтипами объектного типа данных по умолчанию заключается в том, что они не являются взаимно исключающими, чему соответствует оператор ANDOR языка EXPRESS. Экземпляры объектов языка EXPRESS, основанные на комбинации объектных типов данных, называются сложными объектными типами данных. В данном случае отображение заключается в том, что для каждой комбинации, присутствующей в данном экземпляре, определены составные типы данных HDF5. Эти составные типы данных трактуются так, как если бы они были отображением объектных типов данных, непосредственно присутствующих в EXPRESS-схеме. Если какой-либо объектный тип данных в комбинации является супертипом любого другого объектного типа данных из той же комбинации, то идентификатор данного супертипа не присутствует в имени именованного типа данных HDF5.

Примечание — Сказанное выше означает, что в имени именованного типа данных HDF5 присутствуют только имена объектных типов данных самого нижнего уровня.

Относительное имя именованного типа данных HDF5 должно выглядеть следующим образом:

```
<schema_group_name> + "/" + <entity_id> + ( "+" + <entity_id> ),
```

где имена входящих в комбинацию объектных типов данных (entity_id) расположены в алфавитном порядке, а символ "+" является разделителем.

Атрибуты языка EXPRESS трактуются так, как если бы сложный тип данных являлся подтипом всех объектных типов данных, входящих в комбинацию. В случае если унаследованные атрибуты языка EXPRESS в подтипе данных имеют одинаковые имена, то либо с помощью обычного наследования, либо с помощью переобъявления, переименовывающего данный атрибут, представление в HDF5 данного атрибута в данном подтипе получает следующее имя:

```
<schema_group_name> + "/" + <entity_id> + ( "+" + <entity_id> ) + "/" +
<entity_id> + "." + <attribute_id> .
```

Пример – Следующая EXPRESS-схема:

```
SCHEMA test;

  ENTITY a;
    name : STRING;
  END_ENTITY;

  ENTITY b SUBTYPE OF a;
    age : INTEGER;
    x : REAL;
  END_ENTITY;

  ENTITY c SUBTYPE OF a;
    height : REAL;
    x: BOOLEAN;
  END_ENTITY;

END_SCHEMA;
```

приведет к появлению следующих имен в файле HDF5:

```
test/a
test/a/name

test/b
test/b/name
test/b/age
test/b/x
```

```
test/c
test/c/name
test/c/height
test/c/x

test/b+c
test/b+c/name
test/b+c/age
test/b+c/height
test/b+c/b.x
test/b+c/c.x
```

6.8 Отображение атрибутов языка EXPRESS

6.8.1 Общие положения

Конкретные детали отображения явных атрибутов языка EXPRESS зависят от области определения атрибута и представлены в данном подразделе для всех областей определения.

6.8.2 Области определения простых типов данных

Для каждого явного атрибута языка EXPRESS, областью определения которого является простой, перечисляемый или выбираемый тип данных языка EXPRESS, либо определенный тип данных, который сводится к простому или перечисляемому типу данных, определено поле HDF5 с единственным значением, относящимся к типу данных HDF5 в соответствии с 6.4, 6.9.4, 6.9.2 и 6.9.3.2.

6.8.3 Представление значений типа данных ARRAY языка EXPRESS в HDF5

В настоящем стандарте определены два подхода к отображению атрибутов языка EXPRESS, имеющих агрегированные значения:

a) в случае больших агрегированных экземпляров для представления данных используется отдельный набор данных HDF5;

b) в случае небольших агрегированных экземпляров для представления данных может быть использован тот же самый набор данных HDF5, который содержит диапазон объектов языка EXPRESS (т.е. агрегированные экземпляры встраиваются в составной тип данных, представляющий экземпляр родительского объекта языка EXPRESS).

Создание отдельного набора данных HDF5 с целью оптимизации доступа для небольших агрегированных экземпляров не является необходимым. Встраивание агрегированного экземпляра в представление родительского экземпляра позволяет избежать переполнения, вызванного многочисленными небольшими наборами данных HDF5. Решение о том, какой способ использовать (определить, является агрегированный экземпляр большим или нет), не определено в настоящем стандарте, а оставлено на усмотрение пользователей.

Примечание — Сказанное выше означает, что программа, обрабатывающая экземпляры объектов языка EXPRESS, имеющих атрибуты с агрегированными значениями, должна проверять определение типа данных HDF5, представляющего экземпляр объекта языка EXPRESS, для того чтобы установить, определены ли агрегированные данные в составном типе данных HDF5 как объектные ссылки HDF5 на наборы данных HDF5, либо как встроенный массив HDF5 или тип данных VLEN.

Значение атрибута языка EXPRESS, которое классифицировано как большое агрегированное значение, представляется как набор данных HDF5, в котором присутствует массив HDF5 или структура данных VLEN, содержащий реальные данные. В HDF5 все элементы массива могут иметь один и тот же тип данных HDF5. Кроме того, должны быть установлены число вложений массива (в терминах HDF5 это называется «рангом» массива) и размерность для каждого вложенного массива. Ранг массива в HDF5 должен быть представлен целым числом, определяющим число вложений массива, а размерности в HDF5 соответствуют числу элементов на каждом уровне. Массивы языка EXPRESS хранятся как отдельные наборы данных, доступ к которым осуществляется с помощью ссылок на наборы данных HDF5. Ссылка на набор данных HDF5 является значением, хранящимся в памяти для данного значения атрибута языка EXPRESS, и с нее должна быть снята косвенность для того, чтобы прочитать элементы массива.

Массивы могут содержать элементы, представляющие сброс или возврат в исходное состояние. Поэтому элементы массива хранятся как небольшие составные типы данных, состоящие из двух элементов. Первый элемент относится к типу данных H5T_BITFIELD и имеет имя "`set_unset_array_element`". Второй элемент имеет имя "value", и он может относиться к любому из типов данных, перечисленных в таблице 1, так же как и ссылки на экземпляры (см. 6.10.4) и составные типы данных, представляющие выбираемые типы данных. Если элементом массива является возврат в исходное состояние, то первый элемент составного типа данных должен иметь значение 0, а второй элемент должен быть пустым. Если

второй элемент с именем "value" не пустой, то первый элемент должен иметь значение 1.

Примечания

1 Приложение, читающее и записывающее данные, закодированные в HDF5, само должно определять отличие индекса массива HDF5 от индекса массива языка EXPRESS.

2 В языке программирования С элементы массивов упорядочены. Таким же образом индексы экземпляра массива интерпретируются в HDF5.

Пример — Следующее объявление на языке EXPRESS:

```
ARRAY[1:2] OF ARRAY[1:3} OF INTEGER
```

будет представлено в HDF5 как массив ранга 2 с размерностями 2 и 3.

В таблице 2 определено представление значений элементов N-мерного массива, имеющих одинаковый тип данных языка EXPRESS.

Таблица 2 — Отображение массива языка EXPRESS на HDF5

Тип данных значений элементов N-мерного массива языка EXPRESS	Представление в HDF5
INTEGER	H5T_ARRAY базового типа данных для типа данных INTEGER языка EXPRESS
REAL	H5T_ARRAY базового типа данных для типа данных REAL языка EXPRESS
BOOLEAN	То же, что и для типа данных INTEGER
LOGICAL	То же, что и для типа данных INTEGER
STRING	H5T_ARRAY базового типа данных для типа данных STRING языка EXPRESS
BINARY	H5T_ARRAY базового типа данных для типа данных STRING языка EXPRESS
NUMBER	то же, что и для типа данных REAL
ENUMERATION	то же, что и для типа данных INTEGER

6.8.4 Представление значений агрегированных типов данных языка EXPRESS в HDF5

Так как размерности типов данных SET, LIST и BAG определяются их содержимым, а не схемой, то размерности типов данных HDF5 не могут быть определены на основе определения типов данных языка EXPRESS. Эти размерности должны быть установлены на основе самих данных.

Пример — Следующее объявление на языке EXPRESS:

```
LIST[2:?] OF LIST[2:?} OF INTEGER
```

будет иметь в HDF5 ранг 2, но его размерности не могут быть установлены исходя из информации, имеющейся в данной схеме. Если данные помещены в два списка, в каждом из которых содержится три целых числа, то размерности в HDF5 будут представлены значениями 2 и 3.

Поэтому N-мерные типы данных LIST, BAG и SET языка EXPRESS отображаются на данные VLEN: Variable Length, Variable Length {,Variable Length} <HDF5 Datatype representation>.

Пример – Следующее объявление на языке EXPRESS:

```
LIST[0:?] OF LIST[0:?} OF point
```

будет представлено в HDF5 данными VLEN, содержащими данные VLEN, которые представляют идентификаторы экземпляров объектов (см. 6.10.4), которые ссылаются на представление объектного типа данных языка EXPRESS "point".

Дополнительные примеры приведены в С.7.

6.8.5 Динамическое связывание хранилища агрегированных данных

Считают, что вложенный массив принадлежит к «чистому» типу данных ARRAY, если его агрегированные данные на всех уровнях вложенности принадлежат к типу данных ARRAY.

Для чистого массива его оценка, как большого или небольшого массива, может быть легко сделана на основе схемы, в которой верхние и нижние индексы представлены постоянными значениями. Для того чтобы сделать такую же оценку для агрегированных структур, не являющихся чистыми, должен быть использован метод динамического связывания. Данный метод использует дескриптор агрегированной структуры, представленный составным типом данных, так, что данная оценка может быть отложена до периода выполнения программы.

Первый элемент составного типа данных относится к типу данных H5T_BITFIELD и имеет имя

"obj_ref_or_vlen". Второй элемент относится к типу данных H5T_STD_REF_OBJ и имеет имя "object_reference". Данный элемент не пуст в том случае, когда значением "obj_ref_or_vlen" является 0, и содержит ссылку на отдельный набор данных, в котором хранится агрегированная структура. Третий элемент относится к типу данных H5T_VLEN и имеет имя "vlen_array". Данный элемент не пуст в том случае, когда значением "obj_ref_or_vlen" является 1. При этом данные типа H5T_VLEN встроены в родительский составной тип данных.

6.9 Отображение определенных типов данных языка EXPRESS

6.9.1 Общие положения

Именованные типы данных HDF5, которые представляют определенные типы данных языка EXPRESS, определены так, что они имеют родителя, являющегося представлением EXPRESS-схемы, как определено в 6.5.

6.9.2 Отображение перечисляемых типов данных языка EXPRESS на HDF5

Перечисляемые типы данных языка EXPRESS отображаются так же, как и другие простые определенные типы данных языка EXPRESS, которые отображаются на именованные типы данных HDF5. Однако соответствующий тип данных HDF5 определен непосредственно как HDF5 ENUM, представленный парой <string>:<unsigned integer>, где <string> представляет символическое имя, которому должно быть задано значение в виде <enum_name> + "/" + <enum_literal>. Во второй редакции языка EXPRESS был определен наращиваемый перечисляемый тип данных, следствием чего явилось отсутствие упорядочения, которое могло бы быть связано со значениями элементов перечисляемого типа данных. Поэтому ссылки должны делаться с использованием символического имени.

Пример — Отображение представленного ниже перечисляемого типа данных языка EXPRESS, связанного с представлением схемы "s_encoding", даст в результате именованный тип данных HDF5 "s_encoding/ahead_or_behind" и пары значений ("s_encoding/ahead_or_behind/ ahead", 0), ("s_encoding/ahead_or_behind/exact", 1) и ("s_encoding/ahead_or_behind/behind", 2), представляющие элементы перечисляемого типа данных языка EXPRESS, помня при этом, что целые числа 0, 1 и 2 не должны использоваться в качестве ссылок.

```
SCHEMA s;
  TYPE ahead_or_behind = ENUMERATION OF
    (ahead, exact, behind);
  END_TYPE;
END_SCHEMA;
```

Для каждого наращиваемого перечисляемого типа данных языка EXPRESS любой возможный элемент перечисления должен отображаться так, как если бы он был определен локально.

Пример — Следующие перечисляемые типы данных языка EXPRESS, связанные с представлением схемы "s_encoding":

```
SCHEMA s;
  TYPE x = EXTENSIBLE ENUMERATION OF (a,b);
  END_TYPE;
  TYPE y = ENUMERATION BASED_ON x WITH(c,d);
  END_TYPE;
  TYPE z = ENUMERATION BASED_ON x WITH(e,f);
  END_TYPE;
END_SCHEMA;
```

в результате отображения дадут следующие определения в HDF5:

- ***именованный тип данных HDF5 "s_encoding/x", заданный шестью парами ("s_encoding/x/a", 0), ("s_encoding/x/b", 1), ("s_encoding/x/c", 2), ("s_encoding/x/d", 3), ("s_encoding/x/e", 4) и ("s_encoding/ x/f", 5), представляющими элементы перечисляемого типа данных "x" языка EXPRESS;***
- ***именованный тип данных HDF5 "s_encoding/y", заданный четырьмя парами ("s_encoding/y/a", 0), ("s_encoding/y/b", 1), ("s_encoding/y/c", 2) и ("s_encoding/y/d", 3), представляющими элементы перечисляемого типа данных "y" языка EXPRESS;***
- ***именованный тип данных HDF5 "s_encoding/z", заданный четырьмя парами ("s_encoding/z/a", 0), ("s_encoding/z/b", 1), ("s_encoding/z/e", 2) и ("s_encoding/y/f", 3), представляющими элементы перечисляемого типа данных "z" языка EXPRESS.***

Дополнительный пример приведен в С.4.

6.9.3 Отображение выбираемых типов данных языка EXPRESS на HDF5

6.9.3.1 Общие положения

В настоящем стандарте определены следующие категории выбираемых типов данных языка EXPRESS:

- выбираемый тип данных, в списке выбора которого присутствуют только элементы определенных типов данных, которые сводятся к одному лежащему в их основе простому или агрегированному типу данных;
- выбираемый тип данных, в списке выбора которого присутствуют только элементы объектного типа данных и других выбираемых типов данных, содержащих элементы только объектного типа данных;
- выбираемый тип данных, в списке выбора которого присутствуют элементы разных типов данных.

6.9.3.2 Отображение выбираемого типа данных языка EXPRESS, содержащего элементы простых типов данных, на HDF5

Существуют два возможных варианта:

1) Граф выбора является однозначным. В этом случае отображение аналогично отображению простых определенных типов данных, определенному в 6.9.4.

2) Граф выбора является неоднозначным. В этом случае необходимо знать путь на графе к этому типу данных, и соответствующее отображение определено в 6.9.3.4.

6.9.3.3 Отображение выбираемого типа данных языка EXPRESS, содержащего элементы объектных типов данных, на HDF5

В случае, когда все элементы, содержащиеся в списке выбора выбираемого типа данных языка EXPRESS, принадлежат к объектному типу данных, выбираемый тип данных языка EXPRESS не отображается на файл HDF5. Вместо этого все атрибуты языка EXPRESS, для которых в качестве области определения задан выбираемый тип данных, определяются так, что в качестве их типа данных задается ссылка на соответствующий экземпляр объекта языка EXPRESS (см. 6.10.4).

Пример — Следующий выбираемый тип данных языка EXPRESS, связанный с представлением схемы "s_encoding":

```
SCHEMA s;
  ENTITY x;
    a: REAL;
  END_ENTITY;
  ENTITY y;
    b : INTEGER;
  END_ENTITY;
  TYPE z = SELECT( x, y);
  END_TYPE;
END_SCHEMA;
```

приведет к отображению на два объектных типа данных для "x" и "y", определенному в другом месте, но для "z" не будет создано никакого нового типа данных.

6.9.3.4 Отображение выбираемого типа данных языка EXPRESS, содержащего элементы разных типов данных, на HDF5

В случае, когда элементы, содержащиеся в списке выбора выбираемого типа данных языка EXPRESS, не принадлежат к одному объектному типу данных, представлением в HDF5 значений атрибутов языка EXPRESS, основанных на этом выбираемом типе данных, является составной тип данных HDF5, который имеет одно поле HDF5, определенное для каждого возможного простого типа данных.

Составной тип данных HDF5, представляющий выбираемый тип данных, должен иметь имя <schema_group_name> + "/" + <type_id>, а его поля должны иметь следующее имя, основанное на лежащих в основе основных типах данных: <underlying_type_id> + "-value".

Пример

```
integer-value      (H5T_INTEGER)
real-value         (H5T_FLOAT)
string-value       (H5T_STRING)
instance-value     (see 6.10.4)
boolean-value      (H5T_ENUM)
logical-value      (H5T_ENUM)
binary-value       (H5T_OPAQUE)
```

```
<enumeration-name> (H5T_ENUM)
<name_of_typed_aggregate> (дескриптор агрегированной структуры - по 6.8.5)
```

Если значения данных представлены набором, то только одно из полей должно содержать элементы. В HDF5 отсутствует понятие недостающих данных. Поэтому подобно случаю отображения объектного типа данных (см. 6.6), в составном типе данных HDF5 из списка выбора первым должен быть элемент с именем "select_bitmap". Второй элемент должен иметь имя "type_path" и принадлежать к типу данных H5T_VLEN из H5T_STRING, представляя возможный путь для этого типа данных. В некоторых случаях подобный путь должен иметь составляющие элементы, так как значение выбираемого типа данных в противном случае будет неоднозначным. Такие выбираемые значения обычно называ-ются типизированными данными.

Пример — Следующий выбираемый тип данных языка EXPRESS, связанный с представлением схемы "s_encoding":

```
SCHEMA s;
 TYPE r = REAL;
 END_TYPE;
 TYPE i = INTEGER;
 END_TYPE;
 TYPE n = SELECT( r, i);
 END_TYPE;
 ENTITY e;
    a : n;
 END_ENTITY;
END_SCHEMA;
```

приведет к отображению атрибута "e.a" языка EXPRESS, которое будет представлено составным типом данных HDF5 с именем "s_encoding/n" и полями HDF5 "select_bitmap", "real-value" и "integer-value".

Дополнительные примеры приведены в С.5.

6.9.4 Отображение простых определенных типов данных языка EXPRESS

Атрибуты объектов языка EXPRESS, базирующиеся на простых типах данных, отображаются непосредственно на основные типы данных HDF5, как определено в 6.4.

Существует дополнительная возможность отображать простые определенные типы данных языка EXPRESS на HDF5. В этом случае отображение осуществляется описанным ниже способом.

Для определенных типов данных языка EXPRESS, являющихся конкретизациями простых типов данных, должен быть создан именованный тип данных HDF5 в группе HDF5, представляющей EXPRESS-схему, для конкретных элементов отображаемого типа данных. Этот именованный тип данных должен иметь следующее относительное имя:

```
<schema_ group_name> + "/" + <type_id>.
```

Тип данных HDF5 именованного типа данных HDF5 является представлением базового типа определенного типа данных языка EXPRESS, как определено в 6.4.

Пример — Следующий определенный тип данных языка EXPRESS из EXPRESS-схемы "s" с соответствующей группой HDF5 с именем "s_encoding":

```
SCHEMA s;
  TYPE x = REAL;
  END_TYPE;
END_SCHEMA;
```

приведет к отображению на именованный тип данных HDF5 с именем "s_encoding/x" и базовым типом данных HDF5 H5T_FLOAT.

Отображение определенных типов данных языка EXPRESS, конкретизирующих другие определенные типы данных языка EXPRESS, осуществляется аналогичным образом. Новый именованный тип данных HDF5 определяется с базовым типом данных HDF5 другого именованного типа данных HDF5.

Пример – Пусть задана следующая EXPRESS-схема:

```
SCHEMA s;
  TYPE positive_integer = INTEGER;
  END_TYPE;
  TYPE big_positive_integer = positive_integer;
  END_TYPE;
END_SCHEMA;
```

Объявление первого типа данных отобразится на именованный тип данных HDF5 с именем "s_encoding/positive_integer" и базовым типом данных HDF5 H5T_INTEGER. Объявление второго типа данных отобразится на именованный тип данных HDF5 с именем "s_encoding/big_positive_ integer" и базовым типом данных HDF5, соответствующим именованному типу данных "s_encoding/positive_integer".

Дополнительный пример приведен в С.5.

6.9.5 Отображение определенных типов данных ARRAY языка EXPRESS

Отображение всех агрегированных типов данных языка EXPRESS представляется с использованием массива HDF5 или типа данных VLEN. Для типа данных ARRAY языка EXPRESS с фиксированными в схеме границами может быть непосредственно определен тип данных HDF5 Array. Однако для других агрегированных типов данных языка EXPRESS размерности агрегированных структур зависят от реального содержимого данных, и поэтому не выводятся непосредственно из EXPRESS-схемы. Именованный тип данных HDF5 должен иметь следующее относительное имя:

```
<schema_group_name> + "/" + <type_id>.
```

Пример — Следующий определенный тип данных языка EXPRESS из EXPRESS-схемы "s" с группой HDF5 с именем "s_encoding", соответствующей данной схеме:

```
SCHEMA s;
  TYPE context_dependent_measure = REAL;
  END_TYPE;
  TYPE anisotropic_symmetric_tensor2_2d =
    ARRAY [1:3] OF context_dependent_measure;
  END_TYPE;
END_SCHEMA;
```

приведет к отображению на именованный тип данных HDF5 с именем "s_encoding/anisotropic_ symmetric_tensor2_2d" и базовым типом данных HDF5 H5T_ARRAY или VLEN, содержащим данные типа H5T_FLOAT, потому что тип данных "s_encoding/context_dependent_measure" должен быть отображен в соответствии с 6.9.4.

6.9.6 Отображение определенных типов данных BAG, LIST и SET языка EXPRESS

Для каждого экземпляра агрегированного определенного типа данных языка EXPRESS, в определении которого встречаются типы данных BAG, SET или LIST, используется тип данных HDF5 VLEN для сохранения каждого экземпляра типа данных BAG, SET или LIST языка EXPRESS. Представляющий данный экземпляр именованный тип данных HDF5, в соответствии с требованием для определенных типов данных ARRAY языка EXPRESS, не должен быть включен в файл HDF5.

Пример — Следующий определенный тип данных языка EXPRESS из EXPRESS-схемы "s" с группой HDF5 с именем "s_encoding", соответствующей данной схеме:

```
SCHEMA s;
  TYPE real_list = LIST [1:?] OF REAL;
  END_TYPE;
  ENTITY x;
    x_real_list : real_list;
  END_ENTITY;
END_SCHEMA;
```

приведет к отображению на тип данных VLEN, не обязательно имеющий имя, хотя это и возможно. Базовым типом данных HDF5 должен быть HDF5 IEEE Floating Point 32 или 64 bits, Big- или Little-Endian. Настоящий стандарт не требует, чтобы этот тип данных был определен в группе "s_encoding".

6.10 Отображение экземпляров объектов языка EXPRESS

6.10.1 Общие положения

Экземпляры из пространства объектов языка EXPRESS представляются с использованием по крайней мере одного набора данных HDF5. Может потребоваться множество наборов данных HDF5 в том случае, если у экземпляров объектов присутствуют атрибуты с агрегированными значениями. Экземпляры объекта типа данных CONSTANT языка EXPRESS представляются с использованием того же отображения, что и для экземпляров обычных объектов языка EXPRESS.

Примечание — На экземпляры объекта типа данных CONSTANT языка EXPRESS не могут быть сделаны прямые ссылки с помощью обычных ссылок на экземпляры объектов. Цель их включения в отображение заключается в том, что их необходимо учитывать при проверке достоверности набора данных относительно EXPRESS-схемы.

6.10.2 Представление экземпляров простых объектов языка EXPRESS в HDF5

В рамках содержимого EXPRESS-схемы полный набор экземпляров объектов языка EXPRESS одного объектного типа данных, часто называемый пространством объектов, представляется группой HDF5. Группа HDF5, содержащая экземпляры объектов языка EXPRESS, имеет следующее относительное имя:

```
<entity_id> + "_objects".
```

Группа HDF5, связанная с объектным типом данных языка EXPRESS, содержит набор данных HDF5, в состав которого входят все экземпляры объектов языка EXPRESS конкретного объектного типа данных. Набор данных HDF5, содержащий экземпляры объектов языка EXPRESS, должен иметь следующее относительное имя:

```
<entity_id> + "_objects" + "/" + <entity_id> + "_instances".
```

Пример — На рисунке 5 показано общее представление, не содержащее всех деталей, демонстрирующее два набора данных HDF5, один из которых содержит экземпляры объектов языка EXPRESS, принадлежащих к объектному типу данных "product", а второй содержит экземпляры объектов языка EXPRESS, принадлежащих к объектному типу данных "product_definition formation".

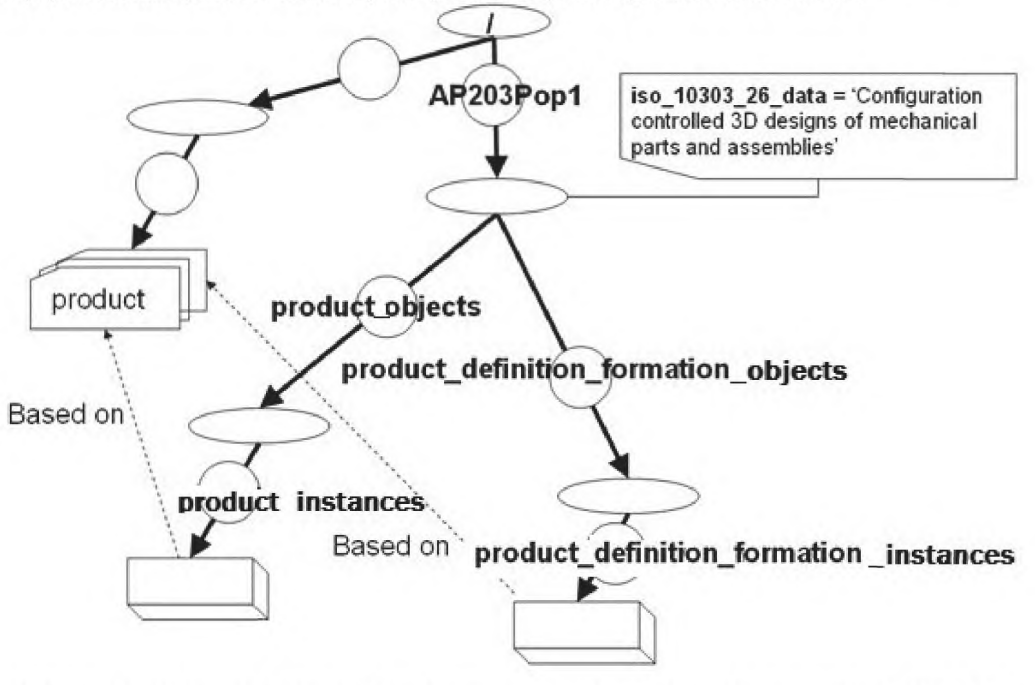

Рисунок 5 – Наборы данных HDF5 с экземплярами простых объектов языка EXPRESS

Набор данных HDF5, содержащий пространство объектов языка EXPRESS, базируется на именованном типе данных HDF5, представляющем объектный тип данных языка EXPRESS в соответствии с 6.6. Набор данных HDF5 должен также иметь связанное с ним пространство данных HDF5 для того, чтобы определить ранг и размерность данного набора. Ранг HDF5 должен быть равен 1, а размерность HDF5 зависит от числа экземпляров в пространстве объектов языка EXPRESS. Следова-

тельно, размерность не известна до начала реализации процесса отображения.

В настоящем стандарте определены два способа отображения атрибутов языка EXPRESS, имеющих агрегированные значения:

a) для больших агрегированных экземпляров для представления данных используется отдельный набор данных HDF5;

b) для небольших агрегированных экземпляров для представления агрегированных структур используется один набор данных HDF5, содержащий пространство объектов языка EXPRESS (т. е. агрегированный экземпляр встраивается в свой родительский составной тип данных).

Решение о том, какой способ использовать (определить, является агрегированный экземпляр большим или нет) субъективно для каждой реализации, и поэтому не определено в настоящем стандарте.

Примечание — Сказанное выше означает, что программа, обрабатывающая экземпляры объектов языка EXPRESS, имеющих атрибуты с агрегированными значениями, должна проверять определение типа данных HDF5, представляющего экземпляр объекта языка EXPRESS, для того чтобы установить, определены ли агрегированные данные в составном типе данных HDF5 как объектные ссылки HDF5 (на наборы данных HDF5 в соответствии с 6.10.3), либо как встроенный массив HDF5 или тип данных VLEN. Для объектных типов данных языка EXPRESS, у которых нет атрибутов с агрегированными значениями, представленных в отдельных наборах данных HDF5, никаких дополнительных структур не требуется.

6.10.3 Экземпляры объектов языка EXPRESS со связанными с ними наборами экземпляров агрегированных типов данных

Как было определено выше, группа HDF5, связанная с объектным типом данных языка EXPRESS, может содержать наборы данных HDF5, представляющие атрибуты с агрегированными значениями (см. 6.8.3 и 6.8.4). Относительное имя набора данных HDF5, содержащего агрегированные экземпляры языка EXPRESS, должно иметь следующий вид:

```
<schema_group_name> + "/" + <entity_id> + "_objects" + "/" + "Aggr_" +
<attribute_id> + "_" + <aggr_unique_id>.
```

Пример — На рисунке 6 показаны три набора данных HDF5, связанных с одним объектным типом данных языка EXPRESS, один из которых представляет диапазон объектов языка EXPRESS, соответствующий объектному типу данных с именем "representation", а два других представляют агрегированные экземпляры языка EXPRESS, являющиеся значениями атрибута "representation.items".

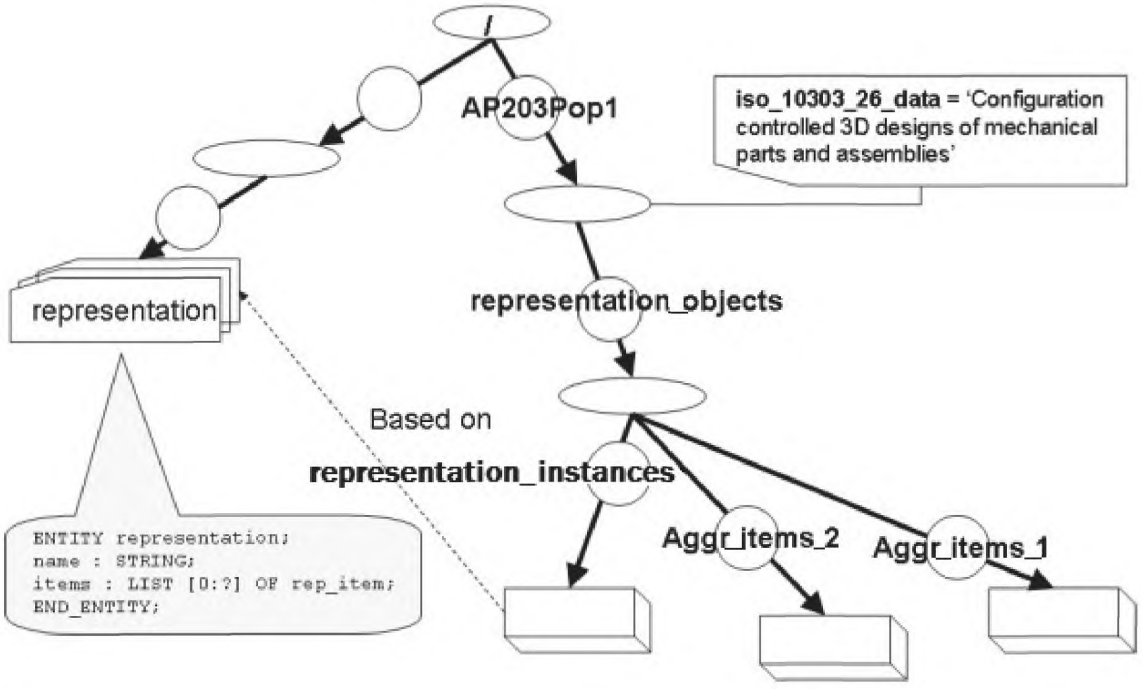

Рисунок 6 — Наборы данных HDF5, связанные с экземпляром объекта и агрегированными экземплярами языка EXPRESS

Так как каждый набор данных HDF5 представляет единственный экземпляр агрегированного типа данных языка EXPRESS, то идентификатор набора данных HDF5 сам по себе достаточен для представления идентификатора агрегированного типа данных языка EXPRESS.

6.10.4 Ссылки на экземпляры объектов языка EXPRESS

Идентификатор экземпляра объекта языка EXPRESS включен в качестве члена в составной тип данных HDF5, который представляет соответствующий экземпляр объекта. Данный член должен иметь имя Entity-Instance-Identifier и тип данных H5T_INTEGER. Если на объект языка EXPRESS имеется ссылка от другого объекта языка EXPRESS, то ссылка на него должна быть создана и включена в качестве члена в составной тип данных HDF5, который представляет ссылающийся на него объектный тип данных. Имя данного члена должно совпадать с именем соответствующего атрибута объекта языка EXPRESS. Данный член должен относиться к составному типу данных HDF5 и иметь имя _HDF_ INSTANCE_REFERENCE_HANDLE_. Этот составной тип данных должен содержать два индекса. Первый индекс должен иметь имя _HDF5_dataset_index_ и представлять индекс в массиве имен наборов данных, хранящийся как атрибут (iso_10303_26_data_set_ names) родительской группы. В действительности данный атрибут является массивом, содержащим имена объектов, соответствующие именам наборов данных. Реальные имена наборов данных формируются с помощью соглашений об именах, определенных в 6.10. Второй индекс должен иметь имя _HDF5_instance_index_ и указывать на реальный экземпляр, хранящийся в наборе данных, на который ссылается индекс набора данных.

Пример приведен в С.6.

Приложение А
(обязательное)

Обозначение документа

Для однозначного обозначения информационного объекта в открытой системе настоящему стандарту присвоен следующий идентификатор объекта:

{ iso standard 10303 part(26) version(1) }

Смысл данного обозначения установлен в ИСО/МЭК 8824-1 и описан в ИСО 10303-1.

Приложение В
(справочное)

Техническое обсуждение

В.1 Введение

Данное приложение содержит обсуждение, касающееся взаимосвязи между языком EXPRESS и двоичным представлением данных, определенных на языке EXPRESS. Кроме того, описаны примеры сценариев, поддерживаемых настоящим стандартом.

В.2 Обзор HDF5

Пользователям, которые мало знакомы с HDF5, следует начать с игнорирования разных сложностей, и сконцентрироваться на простом использовании основных архитектурных особенностей HDF5, заключающихся в следующем:

- файл HDF5 может содержать данные и объекты многих типов (логические группировки данных, определения типов данных, битовые образы, большие массивы данных и т.д.);
- группа HDF5 (HDF5 Group) является логической структурой в HDF5. Группы HDF5 содержат другие группы HDF5 и наборы данных HDF5 (HDF5 Datasets). Группа HDF5 подобна понятию каталога или папки в файловой системе компьютера;
- группа HDF5 самого верхнего уровня называется корневой группой (Root Group), и в каждом файле HDF5 существует только одна корневая группа;
- набор данных HDF5 содержит данные из файла. Набор данных HDF5 содержит определение типов данных HDF5 (HDF5 Datatypes), определение размерностей массива, который в нем содержится, называемых пространством данных HDF5 (HDF5 Dataspace), и сам массив данных;
- тип данных HDF5 может быть простым или составным, подобно типу данных struct в языке программирования С, и может иметь имя;
- пространство данных HDF5 определяет размерности массива, а также содержит информацию о том, как данный массив хранится;
- группы HDF5 и наборы данных HDF5 могут иметь прикрепленные к ним атрибуты HDF5 (HDF5 Attributes), предназначенные для задания некоторых характеристик группы или набора данных.

Формат файлов HDF5 был выбран в качестве базовой технологии для настоящего стандарта после исследования текущего состояния существующих технологий обработки двоичных данных. Существующие технологии имеют две основные области применения:

-технологии, предназначенные для перемещения небольших, но многочисленных пакетов информации по сети, используются в основном в телекоммуникационной сфере;

-технологии, предназначенные для поддержки очень больших наборов данных с эффективным доступом к их частям, используются в основном в научной сфере.

Учитывая, что сообщество, объединяемое подкомитетом SC4 «Производственные данные» Технического комитета 184 ИСО «Системы автоматизации производства и их интеграция», относится к производственным и техническим отраслям экономики, наивысшие приоритеты былы отданы требованиям поддержки больших объемов данных. Эти требования хорошо согласуются с возможностями и целями HDF5, который и был выбран в качестве базовой технологии для настоящего стандарта. Однако это не исключает того, что в будущих редакциях настоящего стандарта будут использованы возможности представления двоичных данных, основанные на других базовых технологиях.

Основным источником для понимания технологии HDF5 является «Руководство пользователя HDF5» [2], глава 1 «Модель данных HDF5 и структура файлов». Пользователям настоящего стандарта рекомендуется начать со знакомства с этим документом. Еще одним важным источником является «Учебное пособие по HDF5» [4].

В.3 Сценарий использования: Внешние файлы HDF5

Наборы данных HDF5 могут представлять данные, определенные в EXPRESS-схеме, как это определено в настоящем стандарте. Одним из преимуществ данного метода является то, что не требуется никакой модификации исходной EXPRESS-схемы.

В дополнение к методу отображения данных, определенных на языке EXPRESS, непосредственно на файлы HDF5 можно использовать файлы HDF5 как часть обменного набора данных, в котором на файл HDF5 дается ссылка из представления данных, определенных на языке EXPRESS, основанного на других методах реализации комплекса ИСО 10303.

Преимущество второго метода заключается в том, что HDF5 используется только для больших объемов данных в таблице, для которой требуется эффективная обработка. Данные в таблице могут быть созданы приложением, которое ничего не знает о языке EXPRESS. Данные, определенные на языке EXPRESS, представляют собой упакованный массив, который добавляет семантику к набору данных HDF5.

В данном сценарии использования файл HDF5 может рассматриваться как таблица с данными. Таблица с данными может также моделироваться как объект в EXPRESS-схеме. Примером такого объекта является explicit_table_function, определенный в ИСО 10303-50. Объект, являющийся таблицей с данными, может также быть экземпляром объекта externally_defined_item, определенного в ИСО 10303-41, который для своего определения ссылается на набор данных HDF5.

Способ использования таблицы с данными для представления изменения характеристики относительно переменной описан в ИСО 10303-51. Кроме того, в ИСО 10303-51 используется объект representation_context, опреде-

ленный в ИСО 10303-43.

В.4 Сценарий использования: Контроль и проверка теплового баланса

Контроль и проверка теплового баланса является обычным действием в процессе создания изделия. Конкретным примером данного действия в аэрокосмической промышленности является тестирование теплового баланса (ТБ), которое выполняется для того, чтобы проверить, что реальное изделие функционирует правильно.

Во время тестирования ТБ космический аппарат помещается в граничные условия эксплуатации и работает в них в условиях, максимально приближенных к условиям реального космического полета. Эти условия работы поддерживаются достаточно продолжительное время для того, чтобы оборудование достигло состояния теплового равновесия. После достижения устойчивого состояния, полученные данные используют для сравнения результатов тестирования с аналитическими расчетами для данных условий. Обычно некоторые поправки должны быть внесены в имитационную модель, для того чтобы получить удовлетворительное соответствие между реальным поведением космического аппарата и его поведением, предсказанным на основе модели. Теперь имитационная модель считается «коррелированной» и может быть использована для предсказаний поведения космического аппарата во время реального полета, которое считается максимально приближенным к реальности.

При обычном тестировании ТБ космического аппарата число датчиков (обычно это датчики температуры) может находиться в диапазоне от менее сотни до нескольких сотен. Кроме того, в процессе тестирования используются полетные датчики, число которых превышает 50. Все эти датчики опрашиваются с большой частотой, для того чтобы как можно раньше предупредить о возникновении нежелательных состояний и выдавать управляющие сигналы с нужной частотой. Показания от большинства из этих датчиков записываются не реже одного раза в минуту.

Тестирование ТБ может длиться от нескольких дней до нескольких недель, в результате чего записываются миллионы показаний датчиков, а размер файлов с данными достигает гигабайтов. Не является исключением то, что некоторые из этих данных поступают из разных мест от субподрядчиков, которые используют разные программные средства.

Все сказанное выше приводит в необходимости выполнения двух утомительных и чреватых ошибками действий:

- интеграция данных;
- обеспечение доступности тестовых данных для средств имитационного моделирования.

Использование двоичного представления данных, определенных на языке EXPRESS, значительно сокращает время, необходимое для выполнения указанных действий, и, кроме того, дополнительно обеспечивает возможность долговременного сохранения и извлечения данных. Это достигается благодаря тому, что обеспечивается возможность использования модели данных, которая может быть разработана под заказчика и представлена на ряде языков, например UML или EXPRESS, при наличии высокоэффективного механизма архивирования и извлечения данных (HDF5).

На рисунке В.1 представлена таксономия информации, относящейся к тестированию ТБ. Она может быть использована для разработки заказной модели данных или отображена на существующую модель данных, такую как STEP-NRF. Если преследуется цель реальной демонстрации использования двоичного представления данных, определенных на языке EXPRESS, то данные для реализации любой из этих моделей могут быть получены.

В.5 Произвольный доступ и навигация

В ходе разработки требований, относящихся к настоящему стандарту, была выявлена необходимость поддержки произвольного доступа и эффективной навигации для очень больших наборов данных. Технический анализ существующих возможностей управления очень большими двоичными наборами данных показал, что данное требование не может быть выполнено при помощи какой-либо из доступных технологий. Исследования по поддержке данного требования продолжаются, и остается надеяться, что будущая редакция настоящего стандарта будет учитывать данное требование, если будет доступна соответствующая технология.

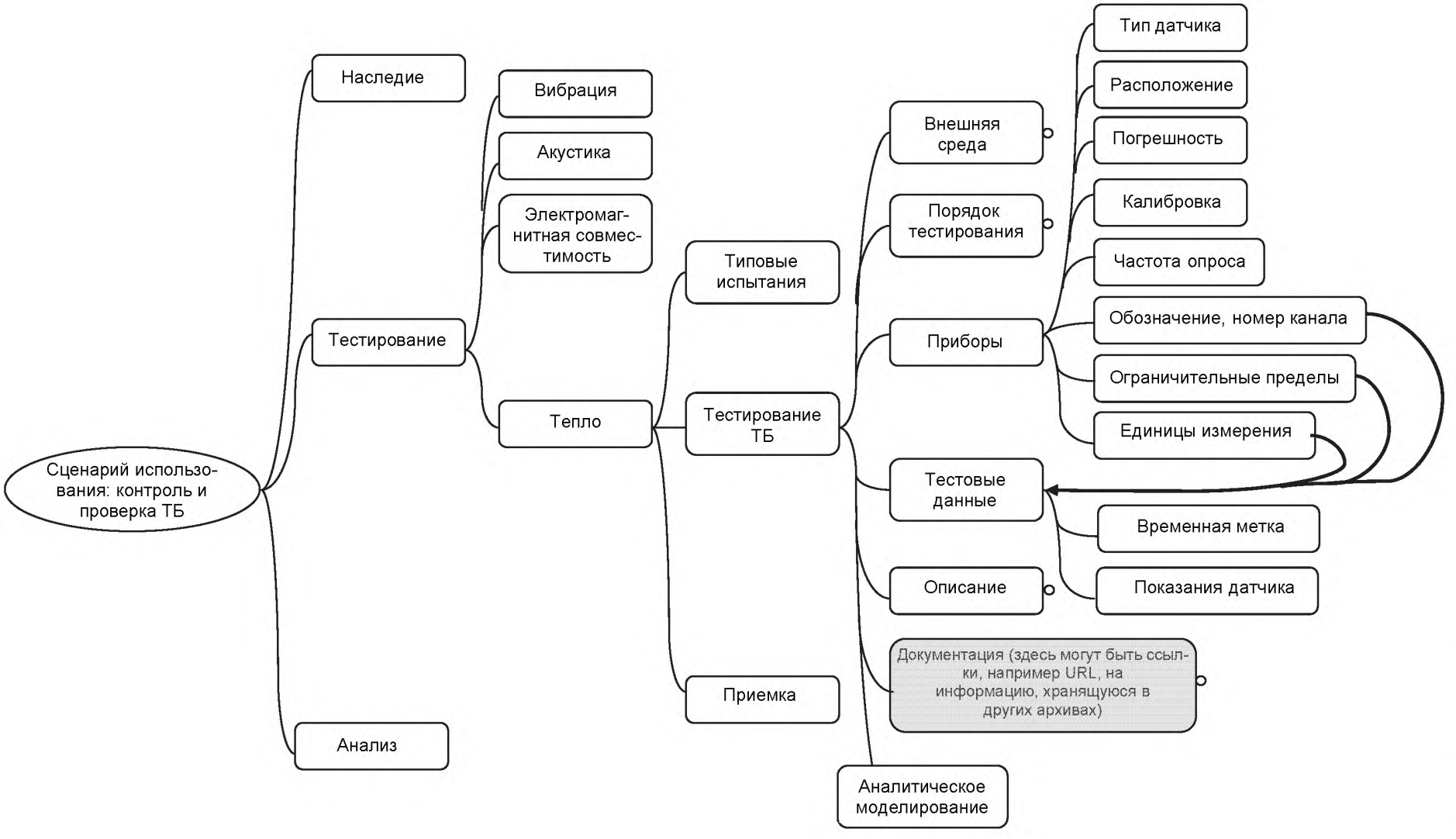

Рисунок В.1 – Таксономия для сценария контроля и проверки теплового баланса

Приложение С
(справочное)

Примеры

С.1 Введение

Данное приложение содержит несколько примеров того, как применяются правила отображения, определенные в настоящем стандарте. В примерах использован API HDF5, разработанный Группой HDF5.

В приведенных примерах использованы функции и типы данных, объявленные в следующем фрагменте кода:

```
#ifndef EXAMPLES
#define EXAMPLES

#include "hdf5.h"

#define NUMBER_OF_ENTITIES 4
#define NO_OF_PPOINTS      4
#define NO_OF_MANY_POINTS 25

hid_t file_id;
hid_t instance_reference_tid;
long  instance_reference_tid_size;
char *defined_entity_names[NUMBER_OF_ENTITIES];
char *entity_names[NUMBER_OF_ENTITIES];

herr_t define_instance_reference(hid_t group_id);
herr_t define_compound_point(hid_t group_id, hid_t  *pp_tid);
herr_t define_compound_line(hid_t group_id, hid_t col_tid,
hid_t *ll_tid);
herr_t define_colour_enumeration(hid_t group_id, hid_t *eenum_col_tid);
herr_t define_compound_colour(hid_t group_id, hid_t eenum_col_tid,
hid_t  *cc_tid);
herr_t define_aggr_reference(hid_t group_id, hid_t aggr_tid,
char *aggr_id, hid_t *aggr_ref_tid);
herr_t create_schema_group(hid_t file_id, char *group_name,
char *schema_name, hid_t *ggroup_id);
herr_t create_pop_group(hid_t file_id, char *group_name,
char *schema_name, hid_t *ggroup_id);
herr_t create_ppoints(hid_t group_id, hid_t p_tid);
herr_t create_llines(hid_t group_id, hid_t l_tid);
hid_t define_nested_real_aggr(void);
herr_t create_nested_real_aggr(hid_t group_id,
hid_t  vlen_of_vlen_double_tid, char **buffer);
hid_t define_nested_point_aggr(hid_t p_tid);
herr_t create_nested_point_aggr(hid_t group_id,
hid_t  vlen_of_vlen_p_tid);
hid_t define_nested_ref_aggr(void);
herr_t create_many_points(hid_t group_id, hid_t p_tid);
herr_t define_compound_land_survey(hid_t group_id, hid_t *lls_tid);
herr_t create_survey(hid_t pop_id, hid_t ls_tid,hid_t p_tid);
herr_t create_nested_real_array(hid_t group_id);
long get_dataset_index(char *entity_name);

   enum colors_t {vvoid,red, green, blue, white, black};

   typedef struct inst_ref_t {
     long dataset_index;
     long instance_index;
   } inst_ref_t;

   typedef struct point_t {
     long   set_unset_bitmap;
```

```
    long   id;
    double x;
    double y;
  } point_t;

  typedef struct colour_t {
    long          set_unset_bitmap;
    char          *s_colour;
    enum colors_t e_colour;
  } colour_t;

  typedef struct lline_t {
    long              set_unset_bitmap;
    long              id;
    struct inst_ref_t startp;
    struct inst_ref_t endp;
    colour_t          colour;
  } lline_t;

#endif /* EXAMPLES */
```

С.2 Отображение объявлений EXPRESS-схемы

Следующая EXPRESS-схема используется в качестве контекстной схемы во всех примерах, приведенных в данном приложении.

```
SCHEMA geometry;
  TYPE Colour = ENUMERATION OF
    (
      VVOID,
      RED,
      GREEN,
      BLUE,
      WHITE,
      BLACK
    );
  END_TYPE;

  TYPE SSTRING = STRING;
  END_TYPE;

  TYPE CColour = SELECT(Colour,SSTRING);
  END_TYPE;

  ENTITY Line;
    colour : CColour;
    startp : POINT;
    endp   : POINT;
  END_ENTITY;

  ENTITY Point;
    x : REAL;
    y : REAL;
  END_ENTITY;

  ENTITY Land_survey;
    country    : STRING;
    properties : LIST OF LIST OF POLYGON;
    altitudes  : LIST OF LIST OF LIST OF ALTITUDE;
  END_ENTITY;

END_SCHEMA;
```

Следующая функция показывает, как отобразить EXPRESS-схему на группу HDF5. Входными параметрами данной функции являются: идентификатор файла HDF5; имя создаваемой группы HDF5; имя EXPRESS-схемы. Последний параметр — "ggroup_id" является выходным параметром, который содержит идентификатор новой создаваемой группы HDF5. Правила, использованные для реализации данного отображения, описаны в 6.5.

```
#include <string.h>
#include "hdf5.h"
#include "examples.h"

herr_t create_schema_group(hid_t file_id, char *group_name,
char *schema_name, hid_t *ggroup_id)
{
  herr_t rstat;
  hid_t  group_id;
  hid_t  attr_id;
  hid_t  string_tid;
  hid_t  scalar_tid;

  rstat = -1;

  *ggroup_id = -1;
  group_id = H5Gcreate(file_id, group_name, H5P_DEFAULT, H5P_DEFAULT,
  H5P_DEFAULT);
  if(group_id > 0) {
    *ggroup_id = group_id;
    string_tid   = H5Tcopy(H5T_C_S1);
    H5Tset_size(string_tid, strlen(schema_name)+1);
    scalar_tid   = H5Screate(H5S_SCALAR);
    attr_id = H5Acreate (group_id, "iso_10303_26_schema", string_tid,
    scalar_tid, H5P_DEFAULT, H5P_DEFAULT);
    rstat = H5Awrite(attr_id,string_tid,schema_name);
  }

  return(rstat);
}
```

Следующий фрагмент текста представляет распечатку результирующего файла HDF5, полученного после выполнения представленной выше функции. Результирующий текстовый файл создан утилитой распечатки HDF5 (HDF5 dump). Он структурирован в соответствии с синтаксисом языка определения данных HDF5 (DDL).

```
HDF5 "example.h5" {
GROUP "/" {
   GROUP "Geometry_encoding" {
      ATTRIBUTE "iso_10303_26_schema" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
    }
}
}
```

С.3 Представление содержимого данных, определенных на языке EXPRESS, в HDF5

Следующая функция демонстрирует создание группы HDF5, содержащей данные, определенные на языке EXPRESS, из приведенной выше контекстной схемы. Правила, применяемые для выполнения данного отображения, описаны в 6.3.3.

Необходимо учесть, что глобальная переменная “defined_entity_names” инициализирована в главной про-

грамме (см. С.10).

```
#include <string.h>
#include "hdf5.h"
#include "examples.h"

herr_t create_pop_group(hid_t file_id, char *group_name,
char *schema_name, hid_t *ggroup_id)
{
  herr_t  rstat;
  hid_t   group_id;
  hid_t   attr_id;
  hid_t   string_tid;
  hid_t   scalar_tid;
  hid_t   string_array_tid;
  hsize_t array_dim[1];

  rstat = -1;

  *ggroup_id = -1;
  group_id = H5Gcreate(file_id, group_name, H5P_DEFAULT, H5P_DEFAULT,
  H5P_DEFAULT);
  if(group_id > 0) {
    *ggroup_id = group_id;
    string_tid   = H5Tcopy(H5T_C_S1);
    H5Tset_size(string_tid, strlen(schema_name)+1);
    scalar_tid   = H5Screate(H5S_SCALAR);
    attr_id = H5Acreate (group_id, "iso_10303_26_data", string_tid,
    scalar_tid, H5P_DEFAULT, H5P_DEFAULT);
    rstat = H5Awrite(attr_id,string_tid,schema_name);

    array_dim[0] = 4;
    rstat = H5Tset_size(string_tid, H5T_VARIABLE);
    string_array_tid = H5Tarray_create2(string_tid, 1, array_dim);
    attr_id = H5Acreate(group_id, "_10303_26_data_set_names",
    string_array_tid, scalar_tid, H5P_DEFAULT, H5P_DEFAULT);
    rstat = H5Awrite(attr_id,string_array_tid,defined_entity_names);
  }

  return(rstat);
}
```

Следующий фрагмент текста представляет распечатку результирующего файла HDF5, полученного в результате выполнения приведенной выше функции.

```
HDF5 "example.h5" {
GROUP "/" {
   GROUP "Geometry_population" {
      ATTRIBUTE "_10303_26_data_set_names" {
         DATATYPE  H5T_ARRAY { [4] H5T_STRING {
               STRSIZE H5T_VARIABLE;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            } }
         DATASPACE  SCALAR
         DATA {
         (0): [ "Point", "Many_Point", "Line", "Land_survey" ]
         }
      }
      ATTRIBUTE "iso_10303_26_data" {
         DATATYPE  H5T_STRING {
```

```
                STRSIZE 16;
                STRPAD H5T_STR_NULLTERM;
                CSET H5T_CSET_ASCII;
                CTYPE H5T_C_S1;
             }
          DATASPACE  SCALAR
          DATA {
          (0): "Geometry_schema"
          }
       }
    }
}
}
```

С.4 Отображение объектных типов данных языка EXPRESS

Следующая функция демонстрирует отображение объектного типа данных языка EXPRESS на составной тип данных HDF5 с использованием API HDF5. В частности, показано как объект “Point” языка EXPRESS, определенный в приведенной выше контекстной схеме, отображается на HDF5. Правила, применяемые для выполнения данного отображения, описаны в 6.6.

```
//-----------------------------------------------------------------
//  Define Point:
//
//  Only simple types are encoded is this case
//-----------------------------------------------------------------
herr_t define_compound_point(hid_t group_id, hid_t  *pp_tid)
{
   herr_t rstat;
   hid_t  p_tid;

   *pp_tid = -1;
//
// Create named compound type for POINT
//
   p_tid = H5Tcreate (H5T_COMPOUND, sizeof(point_t));
   if(rstat = H5Tinsert(p_tid, "set_unset_bitmap", HOFFSET(point_t,
   set_unset_bitmap), H5T_NATIVE_LONG)) goto err;
   if(rstat = H5Tinsert(p_tid, "Entity-Instance-Identifier",
   HOFFSET(point_t, id), H5T_NATIVE_LONG)) goto err;
   if(rstat = H5Tinsert(p_tid, "x", HOFFSET(point_t, x),
   H5T_NATIVE_DOUBLE)) goto err;
   if(rstat = H5Tinsert(p_tid, "y", HOFFSET(point_t, y),
   H5T_NATIVE_DOUBLE)) goto err;
   if(rstat = H5Tcommit2(group_id, "Point", p_tid,H5P_DEFAULT,
   H5P_DEFAULT,H5P_DEFAULT)) goto err;
   *pp_tid = p_tid;

err:
   return(rstat);
}
```

Следующий фрагмент кода используется для наполнения файла HDF5 с помощью создания экземпляров составного типа данных HDF5, представляющего объект “Point”, как это определено функцией “define_compound_point”.

```
#include "hdf5.h"
#include "examples.h"

#define NO_OF_PPOINTS 4

//-----------------------------------------------------------------
//  Create Points:
```

```
//
//  Exemplifies how the Point Compound type, defined by means of the
//  define_compound_point function, is used to populate Points on
//  an HDF5 dataset.
//  Geometrically the Points make up corners of a square(0,0),(100,0),
//  (100,100),(0,100)
//-----------------------------------------------------------------
herr_t create_ppoints(hid_t group_id, hid_t p_tid)
{
  herr_t   rstat;
  hid_t    point_dataset;
  hid_t    myspace;
  hid_t    obj_group;
  hsize_t  mydim[1];
  point_t  ppoints[NO_OF_PPOINTS];
//
// Populate the points in a memory buffer
//
    ppoints[0].set_unset_bitmap = 7;
    ppoints[0].id = 0;
    ppoints[0].x = 0.;
    ppoints[0].y = 0.;

    ppoints[1].set_unset_bitmap = 7;
    ppoints[1].id = 1;
    ppoints[1].x = 100.;
    ppoints[1].y = 0.;

    ppoints[2].set_unset_bitmap = 7;
    ppoints[2].id = 2;
    ppoints[2].x = 100.;
    ppoints[2].y = 100.;

    ppoints[3].set_unset_bitmap = 7;
    ppoints[3].id = 3;
    ppoints[3].x = 0.;
    ppoints[3].y = 100.;

//
//  Create the group that shall contain the dataset
//
   obj_group = H5Gcreate(group_id, "Point_objects", H5P_DEFAULT,
   H5P_DEFAULT, H5P_DEFAULT);
//
//  Create the dataset that shall contain the Points
//
   mydim[0] = NO_OF_PPOINTS;
   myspace = H5Screate_simple(1, mydim, NULL);
   point_dataset = H5Dcreate1(obj_group, "Point_instances", p_tid,
   myspace, H5P_DEFAULT);

//
// Write the Points to the dataset
//
   rstat = H5Dwrite(point_dataset, p_tid, H5S_ALL, H5S_ALL,
   H5P_DEFAULT, ppoints);

   return(rstat);
}
```

Следующий фрагмент текста представляет распечатку результирующего файла HDF5, полученного в результате выполнения приведенных выше двух функций. Определение HDF5 объекта “Point” языка EXPRESS, созданное с помощью функции “define_compound_point”, представлено составным типом данных “Point”. Набор данных “Point_ instances” содержит множество экземпляров, созданных с помощью функции “create_ ppoints”.

```
HDF5 "example.h5" { GROUP "/" {    GROUP "Geometry_encoding" {
      ATTRIBUTE "iso_10303_26_schema" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
      DATATYPE "Point" H5T_COMPOUND {
         H5T_STD_I32LE "set_unset_bitmap";
         H5T_STD_I32LE "Entity-Instance-Identifier";
         H5T_IEEE_F64LE "x";
         H5T_IEEE_F64LE "y";
      }
   }
   GROUP "Geometry_population" {
      ATTRIBUTE "_10303_26_data_set_names" {
         DATATYPE  H5T_ARRAY { [4] H5T_STRING {
               STRSIZE H5T_VARIABLE;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            } }
         DATASPACE  SCALAR
         DATA {
         (0): [ "Point", "Many_Point", "Line", "Land_survey" ]
         }
      }
      ATTRIBUTE "iso_10303_26_data" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
      GROUP "Point_objects" {
         DATASET "Point_instances" {
            DATATYPE  "/Geometry_encoding/Point"
            DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
            DATA {
            (0): {
                  7,
                  0,
                  0,
                  0
               },
            (1): {
```

```
                7,
                1,
                100,
                0
            },
        (2): {
                7,
                2,
                100,
                100
            },
        (3): {
                7,
                3,
                0,
                100
            }
        }
    }
  }
}
}
```

С.5 Отображение перечисляемых типов данных языка EXPRESS на HDF5

Следующая функция демонстрирует отображение перечисляемого типа данных языка EXPRESS на перечисляемый тип данных HDF5 с использованием API HDF5. В частности, показано, как перечисляемый тип данных “Colour” языка EXPRESS, определенный в приведенной выше контекстной схеме, отображается на HDF5. Правила, применяемые для выполнения данного отображения, описаны в 5.8.1.

```
//-----------------------------------------------------------------
//  Define enumeration:
//
//  Exemplifies how enumeration types are encoded
//-----------------------------------------------------------------
herr_t define_colour_enumeration(hid_t group_id, hid_t *eenum_col_tid)
{
   herr_t rstat;
   short  enum_val;
   hid_t  enum_col_tid;

   *eenum_col_tid = -1;
   enum_col_tid = H5Tcreate (H5T_ENUM, sizeof(short));
   if(rstat = H5Tenum_insert(enum_col_tid, "VVOID",(enum_val=0,
   &enum_val))) goto err;
   if(rstat = H5Tenum_insert(enum_col_tid, "RED"  ,(enum_val=1,
   &enum_val))) goto err;
   if(rstat = H5Tenum_insert(enum_col_tid, "GREEN",(enum_val=2,
   &enum_val))) goto err;
   if(rstat = H5Tenum_insert(enum_col_tid, "BLUE" ,(enum_val=3,
   &enum_val))) goto err;
   if(rstat = H5Tenum_insert(enum_col_tid, "WHITE",(enum_val=4,
   &enum_val))) goto err;
   if(rstat = H5Tenum_insert(enum_col_tid, "BLACK",(enum_val=5,
   &enum_val))) goto err;
   if(rstat = H5Tcommit2(group_id, "Colour", enum_col_tid,H5P_DEFAULT,
   H5P_DEFAULT,H5P_DEFAULT)) goto err;
   *eenum_col_tid = enum_col_tid;
err:
   return(rstat);
}
```

Следующий фрагмент текста представляет распечатку результирующего файла HDF5, полученного в результате выполнения функции "define_colour_enumeration". Определение HDF5 перечисляемого типа данных “Colour” языка EXPRESS представлено перечисляемым типом данных “Colour” HDF5. В данном примере ни один из экземпляров не был наполнен данными, поэтому соответствующая конструкция HDF5, представляющая данные, пуста.

```
HDF5 "example.h5" {
GROUP "/" {
   GROUP "Geometry_encoding" {
      ATTRIBUTE "iso_10303_26_schema" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
      DATATYPE "Colour" H5T_ENUM {
            H5T_STD_I16LE;
            "VVOID"           0;
            "RED"             1;
            "GREEN"           2;
            "BLUE"            3;
            "WHITE"           4;
            "BLACK"           5;
         };

   }
   GROUP "Geometry_population" {
      ATTRIBUTE "_10303_26_data_set_names" {
         DATATYPE  H5T_ARRAY { [4] H5T_STRING {
               STRSIZE H5T_VARIABLE;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            } }
         DATASPACE  SCALAR
         DATA {
         (0): [ "Point", "Many_Point", "Line", "Land_survey" ]
         }
      }
      ATTRIBUTE "iso_10303_26_data" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
   }
}
}
```

С.6 Представление выбираемого типа данных языка EXPRESS, содержащего разные типы данных, в HDF5

Следующая функция демонстрирует отображение выбираемого типа данных языка EXPRESS на составной тип данных HDF5 с использованием API HDF5. В частности, показано, как выбираемый тип данных “Ccolour” языка EXPRESS, определенный в приведенной выше контекстной схеме, отображается на HDF5. Правила, применяемые для выполнения данного отображения, описаны в 5.8.2.3.

```
//-----------------------------------------------------------------
//  Define Colour:
//
//  Exemplifies how select types are encoded
//-----------------------------------------------------------------
herr_t define_compound_colour(hid_t group_id, hid_t eenum_col_tid,
hid_t  *cc_tid)
{
   herr_t rstat;
   hid_t  c_tid;
   hid_t  string_tid;

   *cc_tid = -1;
   string_tid   = H5Tcopy(H5T_C_S1);
   if(rstat = H5Tset_size(string_tid, H5T_VARIABLE)) goto err;
//
// Create named compound type for COLOUR
//
   c_tid = H5Tcreate (H5T_COMPOUND, sizeof(colour_t));
   if(rstat = H5Tinsert(c_tid, "select_bitmap", HOFFSET(colour_t,
   select_bitmap), H5T_NATIVE_LONG)) goto err;
   if(rstat = H5Tinsert(c_tid, "s_colour", HOFFSET(colour_t,
   s_colour), string_tid)) goto err;
   if(rstat = H5Tinsert(c_tid, "e_colour", HOFFSET(colour_t,
   e_colour), eenum_col_tid)) goto err;
   if(rstat = H5Tcommit2(group_id, "CColour", c_tid,H5P_DEFAULT,
   H5P_DEFAULT,H5P_DEFAULT)) goto err;
   *cc_tid = c_tid;

err:
   return(rstat);
}
```

Следующий фрагмент текста представляет распечатку файла HDF5, показывающую, как реально выбираемый тип данных “Ccolour” языка EXPRESS кодируется в HDF5.

```
HDF5 "example.h5" {
GROUP "/" {
   GROUP "Geometry_encoding" {
      ATTRIBUTE "iso_10303_26_schema" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
      DATATYPE "CColour" H5T_COMPOUND {
         H5T_STD_I32LE "select_bitmap";
         H5T_STRING {
```

```
            STRSIZE H5T_VARIABLE;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         } "s_colour";
         H5T_ENUM {
            H5T_STD_I16LE;
            "VVOID"            0;
            "RED"              1;
            "GREEN"            2;
            "BLUE"             3;
            "WHITE"            4;
            "BLACK"            5;
         } "e_colour";
      }
      DATATYPE "Colour" H5T_ENUM {
            H5T_STD_I16LE;
            "VVOID"            0;
            "RED"              1;
            "GREEN"            2;
            "BLUE"             3;
            "WHITE"            4;
            "BLACK"            5;
         };

   }
   GROUP "Geometry_population" {
      ATTRIBUTE "_10303_26_data_set_names" {
         DATATYPE  H5T_ARRAY { [4] H5T_STRING {
               STRSIZE H5T_VARIABLE;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            } }
         DATASPACE  SCALAR
         DATA {
         (0): [ "Point", "Many_Point", "Line", "Land_survey" ]
         }
      }
      ATTRIBUTE "iso_10303_26_data" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
   }
}
}
```

С.7 Ссылки на экземпляры объектов языка EXPRESS

Следующая функция демонстрирует отображение объекта “Line” языка EXPRESS, определенного в приведенной выше контекстной схеме, на соответствующий составной тип данных HDF5 с использованием API HDF5. Кроме того, данный пример показывает, как ссылки на экземпляры и атрибуты смешанных выбираемых типов данных отображаются с применением правил, описанных в 5.9.3. Объект “Line” языка EXPRESS ссылается на два экземпляра объекта “Point” (“startp” и “endp”) и имеет атрибут “Ccolour”, относящийся к выбираемому типу данных языка EXPRESS, определенному в приведенной выше контекстной схеме.

```
//_____________________________________________________________________
//  Define Line:
//
//  In addition to simple type encoding, this functions exemplifies
//  how to encode references to entity instances.
//  Moreover it is exemplified how select types , also being defined
//  as Compound types, are embedded.
//_____________________________________________________________________
herr_t define_compound_line(hid_t group_id, hid_t col_tid,
hid_t *ll_tid)
{
   herr_t rstat;
   hid_t  l_tid;

  *ll_tid = _1;
//
// Create named compound type for LINE
//
   l_tid = H5Tcreate (H5T_COMPOUND, sizeof(lline_t));
   if(rstat = H5Tinsert(l_tid, "set_unset_bitmap", HOFFSET(lline_t,
   set_unset_bitmap), H5T_NATIVE_LONG)) goto err;
   if(rstat = H5Tinsert(l_tid, "Entity_Instance_Identifier",
   HOFFSET(lline_t, id), H5T_NATIVE_LONG)) goto err;
   if(rstat = H5Tinsert(l_tid, "startp", HOFFSET(lline_t, startp),
   instance_reference_tid )) goto err;
   if(rstat = H5Tinsert(l_tid, "endp", HOFFSET(lline_t, endp),
   instance_reference_tid )) goto err;
   if(rstat = H5Tinsert(l_tid, "colour", HOFFSET(lline_t, colour),
   col_tid )) goto err;
   if(rstat = H5Tcommit2(group_id, "Line", l_tid,H5P_DEFAULT,
   H5P_DEFAULT,H5P_DEFAULT)) goto err;
   *ll_tid = l_tid;

err:
   return(rstat);
}
```

Следующий фрагмент кода используется для наполнения файла HDF5 в соответствии с полной контекстной схемой.

```
#include <stdlib.h>
#include <string.h>

#include "hdf5.h"
#include "examples.h"

#define NO_OF_LLINES 4

//-------------------------------------------------------------------
//  Create Lines:
//
//  Lines are created by letting the startp and endp
//  attributes (fields) refer to Points in the dataset
//  /Geometry_population/Point_objects/Point_instances
//
//  (000,100)------------------(100,100)
//           .                 .
//           .                 .
//           .                 .
//           .                 .
//  (000,000)-------------------(000,100)
```

```
//
//-----------------------------------------------------------------
herr_t create_llines(hid_t group_id, hid_t l_tid)
{
   herr_t          rstat;
   hid_t           line_dataset;
   hid_t           myspace;
   hid_t           obj_group;
   hsize_t         mydim[1];
   inst_ref_t      inst_ref;
   enum colors_t   line_colour;
   lline_t         lines[NO_OF_LLINES];

//
// Populate the four Lines in a memory buffer
//
   mydim[0] = NO_OF_LLINES;
   myspace = H5Screate_simple(1, mydim, NULL);
//
// Lower Line (0,0) to (100,0)
//
   lines[0].set_unset_bitmap = 7;
   lines[0].id = 4;
   inst_ref.dataset_index = get_dataset_index("Point");
   inst_ref.instance_index = 0;
   memcpy(&lines[0].startp,&inst_ref,sizeof(inst_ref_t));
   inst_ref.instance_index = 1;
   memcpy(&lines[0].endp,&inst_ref,sizeof(inst_ref_t));
   lines[0].colour.select_bitmap = 2;
   line_colour = red;
   lines[0].colour.e_colour = line_colour;
   lines[0].colour.s_colour = "";
//
// Rightmost Line (100,0) to (100,100)
//
   lines[1].set_unset_bitmap = 7;
   lines[1].id = 5;
   inst_ref.instance_index = 1;
   memcpy(&lines[1].startp,&inst_ref,sizeof(inst_ref_t));
   inst_ref.instance_index = 2;
   memcpy(&lines[1].endp,&inst_ref,sizeof(inst_ref_t));
   lines[1].colour.select_bitmap = 2;
   line_colour = blue;
   lines[1].colour.e_colour = line_colour;
   lines[1].colour.s_colour = "";
//
// Upper Line (100,100) to (0,100)
//
   lines[2].set_unset_bitmap = 7;
   lines[2].id = 6;
   inst_ref.instance_index = 2;
   memcpy(&lines[2].startp,&inst_ref,sizeof(inst_ref_t));
   inst_ref.instance_index = 3;
   memcpy(&lines[2].endp,&inst_ref,sizeof(inst_ref_t));
   lines[2].colour.select_bitmap = 1;
   line_colour = vvoid;
   lines[2].colour.e_colour = line_colour;
   lines[2].colour.s_colour = "Red line";
//
// Leftmost Line (0,100) to (0,0)
//
```

```
   lines[3].set_unset_bitmap = 7;
   lines[3].id = 7;
   inst_ref.instance_index = 3;
   memcpy(&lines[3].startp,&inst_ref,sizeof(inst_ref_t));
   inst_ref.instance_index = 0;
   memcpy(&lines[3].endp,&inst_ref,sizeof(inst_ref_t));
   lines[3].colour.select_bitmap = 1;
   line_colour = vvoid;
   lines[3].colour.e_colour = line_colour;
   lines[3].colour.s_colour = "Blue line";
//
// Create the group that shall containg the Line dataset
//
   obj_group = H5Gcreate(group_id, "Line_objects", H5P_DEFAULT,
   H5P_DEFAULT, H5P_DEFAULT);
//
// Create the dataset that shall contain the Lines
//
   line_dataset = H5Dcreate1(obj_group, "Line_instances", l_tid,
   myspace, H5P_DEFAULT);
//
// Write the Lines to the dataset
//
   if(rstat = H5Dwrite(line_dataset, l_tid, H5S_ALL, H5S_ALL,
   H5P_DEFAULT, lines)) goto err;

err:
   return(rstat);
}
```

См. распечатку результирующего файла HDF5 в С.9.

С.8 Представление значений агрегированного типа данных языка EXPRESS в HDF5

Объект “Land_survey” языка EXPRESS, определенный в контекстной схеме, содержит две вложенные агрегированные структуры.

```
//-------------------------------------------------------------------
//  Define Land Survey:
//
//  Exemplifies how references to aggregates are encoded encoded.
//  One aggregate is stored on a separate dataset. The other one is embedded.
//-------------------------------------------------------------------
herr_t define_compound_land_survey(hid_t group_id, hid_t *lls_tid)
{
   herr_t rstat;
   long   mysize,offset;
   long   string_tid_size;
   long   aggr_ref_tid_size;
   hid_t  ls_tid;
   hid_t  string_tid;
   hid_t  aggr1_ref_tid;
   hid_t  aggr2_ref_tid;
   hid_t  vlen_of_vlen_double_tid;
   hid_t  vlen_of_vlen_ref_tid;

  *lls_tid = -1;
  string_tid   = H5Tcopy(H5T_C_S1);
  if(rstat = H5Tset_size(string_tid, H5T_VARIABLE)) goto err;
  string_tid_size = H5Tget_size(string_tid);

  vlen_of_vlen_ref_tid = define_nested_ref_aggr();
  if(rstat = define_aggr_reference(group_id, vlen_of_vlen_ref_tid,
  "1", &aggr1_ref_tid)) goto err;
```

```
  vlen_of_vlen_double_tid = define_nested_real_aggr();
  if(rstat = define_aggr_reference(group_id, vlen_of_vlen_double_tid,
  "2", &aggr2_ref_tid)) goto err;
  aggr_ref_tid_size = H5Tget_size(aggr1_ref_tid);
//
// Create named compound type for LAND_SURVEY
//
   mysize = 2*sizeof(long) + string_tid_size + 2*aggr_ref_tid_size;
   offset = 0;
   ls_tid = H5Tcreate (H5T_COMPOUND, mysize);
   if(rstat = H5Tinsert(ls_tid, "set_unset_bitmap", offset,
   H5T_NATIVE_LONG)) goto err;
   offset += sizeof(long);
   if(rstat = H5Tinsert(ls_tid, "Entity-Instance-Identifier",
   offset, H5T_NATIVE_LONG)) goto err;
   offset += sizeof(long);
   if(rstat = H5Tinsert(ls_tid, "country", offset, string_tid))
   goto err;
   offset += string_tid_size;
   if(rstat = H5Tinsert(ls_tid, "properties", offset, aggr1_ref_tid))
   goto err;
   offset += aggr_ref_tid_size;
   if(rstat = H5Tinsert(ls_tid, "altitudes", offset, aggr2_ref_tid ))
   goto err;
   if(rstat = H5Tcommit2(group_id, "Land_survey",ls_tid,H5P_DEFAULT,
   H5P_DEFAULT,H5P_DEFAULT)) goto err;
   *lls_tid = ls_tid;

err:
   return(rstat);
}
```

Следующий фрагмент кода демонстрирует создание и наполнение составного типа данных HDF5, соответствующего объекту “Land_survey” языка EXPRESS с использованием API HDF5.

```
#include <stdlib.h>
#include <string.h>

#include "hdf5.h"
#include "examples.h"


//-----------------------------------------------------------------
//  Create Survey
//-----------------------------------------------------------------
herr_t create_survey(hid_t group_id, hid_t ls_tid,hid_t p_tid)
{
   herr_t          rstat;
   hid_t           ls_dataset;
   hid_t           myspace;
   hid_t           obj_group;
   hobj_ref_t      objref;
   hid_t           vlen_of_vlen_double_tid;
   hid_t           vlen_of_vlen_ref_tid;
   long            mysize;
   long            myinteger;
   char            mychar;
   hsize_t         mydim[1];
   char            *norway = "Norway";
   char            *vlen_data;
   char            *buffer;
```

```
   char            *mybuffer;

   mysize = H5Tget_size(ls_tid);
   mybuffer = (char*) malloc(mysize);
   buffer = mybuffer;

   if(rstat = create_many_points(group_id, p_tid)) goto err;

   obj_group = H5Gcreate(group_id, "Land_survey_objects", H5P_DEFAULT,
   H5P_DEFAULT, H5P_DEFAULT);

   vlen_of_vlen_double_tid = define_nested_real_aggr();
   vlen_of_vlen_ref_tid = define_nested_ref_aggr();

   if(rstat =  create_nested_real_aggr(obj_group,
   vlen_of_vlen_double_tid,&vlen_data)) goto err;
   if(rstat = create_nested_point_aggr(obj_group,vlen_of_vlen_ref_tid))
   goto err;

   memset(mybuffer,0,mysize);
   myinteger = 15;
   memcpy(mybuffer,&myinteger,sizeof(long));
   mybuffer += sizeof(long);
   myinteger = 25;
   memcpy(mybuffer,&myinteger,sizeof(long));
   mybuffer += sizeof(long);
   memcpy(mybuffer,&norway,sizeof(char*));
   mybuffer += sizeof(char*);

   mychar = (char)1;
   memcpy(mybuffer,&mychar,sizeof(char));
   mybuffer += sizeof(char);
   if(rstat = H5Rcreate(&objref, obj_group, "/Geometry_population/
   Land_survey_objects/Aggr-properties-1", H5R_OBJECT, -1)) goto err;
   memcpy(mybuffer,&objref,sizeof(hobj_ref_t));
   mybuffer += sizeof(hobj_ref_t);
   mybuffer += sizeof(hvl_t);

   mychar = (char)0;
   memcpy(mybuffer,&mychar,sizeof(char));
   mybuffer += sizeof(char);
   mybuffer += sizeof(hobj_ref_t);
   memcpy(mybuffer,vlen_data,sizeof(hvl_t));

//
// Create the LAND SURVEY dataset
//
   mydim[0] = 1;
   myspace = H5Screate_simple(1, mydim, NULL);

   ls_dataset = H5Dcreate1(obj_group, "Land_survey_instances", ls_tid,
   myspace, H5P_DEFAULT);
//
// Write lines to file
//
   if(rstat = H5Dwrite(ls_dataset, ls_tid, H5S_ALL, H5S_ALL,
   H5P_DEFAULT, buffer)) goto err;

err:
   free(mybuffer);
   return(rstat);
```

```
}
```

Для определения и создания многомерных агрегированных структур, являющихся частями составного типа данных "Land_survey", используются вспомогательные функции. Эти функции вызываются из приведенной выше функции "create_survey" и представлены ниже.

```
hid_t define_nested_real_aggr(void)
{
  hid_t  vlen_double_tid;
  hid_t  vlen_of_vlen_double_tid;

 //
 // Create variable-length datatype in two levels,
 // double beeing the base type
 //
   vlen_double_tid = H5Tvlen_create(H5T_NATIVE_DOUBLE);
   vlen_of_vlen_double_tid = H5Tvlen_create (vlen_double_tid);
   vlen_of_vlen_double_tid = H5Tvlen_create (vlen_of_vlen_double_tid);

   return(vlen_of_vlen_double_tid);
}

hid_t define_nested_ref_aggr(void)
{
  hid_t  vlen_ref_tid;
  hid_t  vlen_of_vlen_ref_tid;

 //
 // Create variable-length datatype in two levels, reference type
 // beeing the base type
 //
   vlen_ref_tid = H5Tvlen_create(instance_reference_tid);
   vlen_of_vlen_ref_tid = H5Tvlen_create (vlen_ref_tid);

   return(vlen_of_vlen_ref_tid);
}

#include <stdio.h>
#include <stdlib.h>
#include <float.h>
#include <string.h>
#include "hdf5.h"
#include "examples.h"

#define LEN0            3
#define LEN1            2

//
// NOTE:
// Memory fragments that are allocated in the following
// two functions are not released.
//
// In a real application a memory management system, where VLEN
// memory is grabbed from a pool of large reausable chunks
// should be developed.
// (The pool is released upon application exit)
//
// Did not take the time to develop a memory management system for
// this rather small example, so be aware of this small memory leak.
//
//-----------------------------------------------------------------
//  Create nested REAL aggregate
//
```

```
//  Exemplifies how a nested aggregate of REAL is encoded in HDF5
//  The number of elements on each level are:
//  2-3-6
//     -3
//     -4
//    2-3
//     -6
//-------------------------------------------------------------------
herr_t create_nested_real_aggr(hid_t group_id,  hid_t
vlen_of_vlen_double_tid, char **buffer)
{
    hvl_t *wdata;
    hvl_t *mybuf;
    hvl_t   *tA, *tB;
    herr_t  rstat;
    hsize_t dims[1] = {2};

    rstat = 0;
    /*
     * Initialize variable-length data.
     */
    wdata = (hvl_t*) malloc(2*sizeof(hvl_t));
    mybuf = (hvl_t*) malloc(sizeof(hvl_t));
    *buffer = (char*)mybuf;
    mybuf->len = 2;
    mybuf->p = wdata;
    /* Initializing vector C */

    wdata[0].len = LEN0;
    wdata[0].p = (hvl_t *) malloc ( LEN0 * sizeof (hvl_t));
    wdata[1].len = LEN1;
    wdata[1].p = (hvl_t *) malloc ( LEN1 * sizeof (hvl_t));

    /* Initialize vector A */
    tA = wdata[0].p;
    tA->len = 5;
    tA->p = (double *) malloc (tA->len * sizeof (double));
    ((double *) tA->p)[0] = 100.0;
    ((double *) tA->p)[1] = 101.1;
    ((double *) tA->p)[2] = 102.2;
    ((double *) tA->p)[3] = 103.3;
    ((double *) tA->p)[4] = 104.4;
    tA++;
    tA->len = 3;
    tA->p = (double *) malloc (tA->len * sizeof (double));
    ((double *) tA->p)[0] = 200.0;
    ((double *) tA->p)[1] = 201.1;
    ((double *) tA->p)[2] = 202.2;
    tA++;
    tA->len = 4;
    tA->p = (double *) malloc (tA->len * sizeof (double));
    ((double *)tA->p)[0] = 300.0;
    ((double *)tA->p)[1] = 301.1;
    ((double *)tA->p)[2] = 302.2;
    ((double *)tA->p)[3] = 303.3;
    /* Done with A */

    /* Initialize vector B */
    tB = wdata[1].p;
    tB->len =3;
    tB->p = (double *) malloc (tB->len * sizeof (double));
```

```
    ((double *)tB->p)[0] = 400.0;
    ((double *)tB->p)[1] = 401.1;
    ((double *)tB->p)[2] = 401.2;
    tB++;
    tB->len = 6;
    tB->p = (double *) malloc (tB->len * sizeof (double));
    ((double *) tB->p)[0] = 500.0;
    ((double *) tB->p)[1] = 501.1;
    ((double *) tB->p)[2] = 502.2;
    ((double *) tB->p)[3] = 503.3;
    ((double *) tB->p)[4] = 504.4;
    ((double *) tB->p)[5] = 505.5;
    /* Done with B */

    return(rstat);
}

//-------------------------------------------------------------------
//  Create nested aggregate of references to Point instances
//
//  The number of elements on each level are:
//  2-3-6
//     -3
//     -4
//    2-3
//     -6
//-------------------------------------------------------------------
herr_t create_nested_point_aggr(hid_t group_id,  hid_t
vlen_of_vlen_ref_tid)
{
    long rstat;

    hid_t      space;
    hid_t      dset;
    inst_ref_t inst_ref;
    inst_ref_t *s_p_ref;
    inst_ref_t *t_p_ref;

    hvl_t  wdata[2];           /* Array of vlen structures to write */
    hvl_t  *tA, *tB;
    hsize_t dims[1] = {2};

    rstat = -1;
    /*
     * Initialize variable-length data.
     */

    /* Initializing vector C */

    wdata[0].len = 3;
    wdata[0].p = (hvl_t *) malloc ( 3 * sizeof (hvl_t));
    wdata[1].len = 2;
    wdata[1].p = (hvl_t *) malloc ( 1 * sizeof (hvl_t));

    s_p_ref = &inst_ref;

    /* Initialize vector A */
    tA = wdata[0].p;
    tA->len = 5;
    tA->p = (hdset_reg_ref_t *) malloc (tA->len * sizeof (inst_ref_t));
    t_p_ref = (inst_ref_t*)tA->p;
```

```
inst_ref.dataset_index = get_dataset_index("Many_Point");
inst_ref.instance_index = 0;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 1;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 7;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 11;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 5;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));

tA++;
tA->len = 3;
tA->p = (inst_ref_t *) malloc (tA->len * sizeof (inst_ref_t));
t_p_ref = (inst_ref_t*)tA->p;
inst_ref.instance_index = 1;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 2;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 7;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));

tA++;
tA->len = 4;
tA->p = (inst_ref_t *) malloc (tA->len * sizeof (inst_ref_t));
t_p_ref = (inst_ref_t*)tA->p;
inst_ref.instance_index = 2;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 9;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 13;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 7;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));

/* Initialize vector B */
tB = wdata[1].p;
tB->len = 3;
tB->p = (inst_ref_t *) malloc (tB->len * sizeof (inst_ref_t));
t_p_ref = (inst_ref_t*)tB->p;
inst_ref.instance_index = 2;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 4;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
++t_p_ref;
inst_ref.instance_index = 9;
memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));

tB++;
tB->len = 6;
```

```
    tB->p = (inst_ref_t *) malloc (tB->len * sizeof (inst_ref_t));
    t_p_ref = (inst_ref_t*)tB->p;
    inst_ref.instance_index = 5;
    memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
    ++t_p_ref;
    inst_ref.instance_index = 11;
    memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
    ++t_p_ref;
    inst_ref.instance_index = 7;
    memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
    ++t_p_ref;
    inst_ref.instance_index = 13;
    memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
    ++t_p_ref;
    inst_ref.instance_index = 16;
    memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));
    ++t_p_ref;
    inst_ref.instance_index = 10;
    memcpy(t_p_ref,s_p_ref,sizeof(inst_ref_t));

    /* Done with B */

    /*
     * Create dataspace.  Setting maximum size to NULL sets the maximum
     * size to be the current size.
     */
    space = H5Screate_simple (1, dims, NULL);

    /*
     * Create the dataset and write the variable-length data to it.
     */
    dset = H5Dcreate (group_id, "Aggr-properties-1",
    vlen_of_vlen_ref_tid, space, H5P_DEFAULT, H5P_DEFAULT,H5P_DEFAULT);

    rstat = H5Dwrite (dset, vlen_of_vlen_ref_tid, H5S_ALL,
    H5S_ALL, H5P_DEFAULT, wdata);


    return(rstat);
}



//-------------------------------------------------------------------
//  Create many Points:
//
//  This function is similar to the create_points function.
//  The only difference is that more points are created
//  for the purpose of refering to them from an aggregate defined in
//  Land_survey Compound type.
//  Geometrically the Points make up 25 squares in a mesh.
//  (0000,4000),(1000,4000),(2000,4000),(3000,4000),(4000,4000)
//  (0000,3000),(1000,3000),(2000,3000),(3000,3000),(4000,3000)
//  (0000,2000),(1000,2000),(2000,2000),(3000,2000),(4000,2000)
//  (0000,1000),(1000,1000),(2000,1000),(3000,1000),(4000,1000)
//  (0000,0000),(1000,0000),(2000,0000),(3000,0000),(4000,0000)
//-------------------------------------------------------------------
herr_t create_many_points(hid_t group_id, hid_t p_tid)
{
  int      i,j,k;
  herr_t   rstat;
```

```
  hid_t    point_dataset;
  hid_t    myspace;
  hid_t    obj_group;
  hsize_t  mydim[1];

  point_t  many_points[NO_OF_MANY_POINTS];
//
// Populate the Points in a memory buffer
//
    k = 0;
    for(i = 0; i<5; ++i){
      for(j = 0; j<5; ++j){
        many_points[k].set_unset_bitmap = 7;
        many_points[k].id = k;
        many_points[k].x = j*1000.;
        many_points[k].y = i*1000.;
        ++k;<s
      }
    }
//
//  Create the group that shall contain the dataset
//
   obj_group = H5Gcreate(group_id, "Many_Point_objects", H5P_DEFAULT,
   H5P_DEFAULT, H5P_DEFAULT);
//
//  Create the dataset that shall contain the Points
//
   mydim[0] = NO_OF_MANY_POINTS;
   myspace = H5Screate_simple(1, mydim, NULL);
   point_dataset = H5Dcreate1(obj_group, "Many_Point_instances", p_tid,
   myspace, H5P_DEFAULT);

//
// Write the Points to the dataset
//
   rstat = H5Dwrite(point_dataset, p_tid, H5S_ALL, H5S_ALL,
   H5P_DEFAULT, many_points);

   return(rstat);
}
```

C.9 Полный результирующий файл HDF5 из приведенных примеров

Следующий фрагмент текста представляет распечатку результирующего файла HDF5, полученного после кодирования контекстной схемы в файл HDF5 с применением правил отображения, определенных в настоящем стандарте. Данный файл объединяет результаты всех примеров, представленных в данном приложении.

```
HDF5 "example.h5" {
GROUP "/" {
   GROUP "Geometry_encoding" {
      ATTRIBUTE "iso_10303_26_schema" {
         DATATYPE  H5T_STRING {
               STRSIZE 16;
               STRPAD H5T_STR_NULLTERM;
               CSET H5T_CSET_ASCII;
               CTYPE H5T_C_S1;
            }
         DATASPACE  SCALAR
         DATA {
         (0): "Geometry_schema"
         }
      }
      DATATYPE "AGGREGATE_REFERENCE_1" H5T_COMPOUND {
```

```
      H5T_STD_B8LE "obj_ref_or_vlen";
      H5T_REFERENCE "object_reference";
      H5T_VLEN { H5T_VLEN { H5T_COMPOUND {
      H5T_STD_I32LE "_HDF5_dataset_index_";
      H5T_STD_I32LE "_HDF5_instance_index_";
   }}} "vlen_array";
   }
   DATATYPE "AGGREGATE_REFERENCE_2" H5T_COMPOUND {
      H5T_STD_B8LE "obj_ref_or_vlen";
      H5T_REFERENCE "object_reference";
      H5T_VLEN { H5T_VLEN { H5T_VLEN { H5T_IEEE_F64LE}}} "vlen_array";
   }
   DATATYPE "CColour" H5T_COMPOUND {
      H5T_STD_I32LE "select_bitmap";
      H5T_STRING {
         STRSIZE H5T_VARIABLE;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "s_colour";
      H5T_ENUM {
         H5T_STD_I16LE;
         "VVOID"            0;
         "RED"              1;
         "GREEN"            2;
         "BLUE"             3;
         "WHITE"            4;
         "BLACK"            5;
      } "e_colour";
   }
   DATATYPE "Colour" H5T_ENUM {
      H5T_STD_I16LE;
      "VVOID"            0;
      "RED"              1;
      "GREEN"            2;
      "BLUE"             3;
      "WHITE"            4;
      "BLACK"            5;
   };

   DATATYPE "Land_survey" H5T_COMPOUND {
      H5T_STD_I32LE "set_unset_bitmap";
      H5T_STD_I32LE "Entity-Instance-Identifier";
      H5T_STRING {
         STRSIZE H5T_VARIABLE;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "country";
      H5T_COMPOUND {
         H5T_STD_B8LE "obj_ref_or_vlen";
         H5T_REFERENCE "object_reference";
         H5T_VLEN { H5T_VLEN { H5T_COMPOUND {
         H5T_STD_I32LE "_HDF5_dataset_index_";
         H5T_STD_I32LE "_HDF5_instance_index_";
      }}} "vlen_array";
      } "properties";
      H5T_COMPOUND {
         H5T_STD_B8LE "obj_ref_or_vlen";
         H5T_REFERENCE "object_reference";
         H5T_VLEN { H5T_VLEN { H5T_VLEN { H5T_IEEE_F64LE}}}
```

```
      "vlen_array";
   } "altitudes";
}
DATATYPE "Line" H5T_COMPOUND {
   H5T_STD_I32LE "set_unset_bitmap";
   H5T_STD_I32LE "Entity-Instance-Identifier";
   H5T_COMPOUND {
      H5T_STD_I32LE "_HDF5_dataset_index_";
      H5T_STD_I32LE "_HDF5_instance_index_";
   } "startp";
   H5T_COMPOUND {
      H5T_STD_I32LE "_HDF5_dataset_index_";
      H5T_STD_I32LE "_HDF5_instance_index_";
   } "endp";
   H5T_COMPOUND {
      H5T_STD_I32LE "select_bitmap";
      H5T_STRING {
         STRSIZE H5T_VARIABLE;
         STRPAD H5T_STR_NULLTERM;
         CSET H5T_CSET_ASCII;
         CTYPE H5T_C_S1;
      } "s_colour";
      H5T_ENUM {
         H5T_STD_I16LE;
         "VVOID"            0;
         "RED"              1;
         "GREEN"            2;
         "BLUE"             3;
         "WHITE"            4;
         "BLACK"            5;
      } "e_colour";
      } "colour";
   }
   DATATYPE "Point" H5T_COMPOUND {
      H5T_STD_I32LE "set_unset_bitmap";
      H5T_STD_I32LE "Entity-Instance-Identifier";
      H5T_IEEE_F64LE "x";
      H5T_IEEE_F64LE "y";
   }
   DATATYPE "_HDF_INSTANCE_REFERENCE_HANDLE_" H5T_COMPOUND {
      H5T_STD_I32LE "_HDF5_dataset_index_";
      H5T_STD_I32LE "_HDF5_instance_index_";
   }
}
GROUP "Geometry_population" {
   ATTRIBUTE "_10303_26_data_set_names" {
      DATATYPE  H5T_ARRAY { [4] H5T_STRING {
            STRSIZE H5T_VARIABLE;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         } }
      DATASPACE  SCALAR
      DATA {
      (0): [ "Point", "Many_Point", "Line", "Land_survey" ]
      }
   }
   ATTRIBUTE "iso_10303_26_data" {
      DATATYPE  H5T_STRING {
            STRSIZE 16;
            STRPAD H5T_STR_NULLTERM;
```

```
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         }
      DATASPACE  SCALAR
      DATA {
      (0): "Geometry_schema"
      }
   }
   GROUP "Land_survey_objects" {
      DATASET "Aggr-properties-1" {
         DATATYPE  H5T_VLEN { H5T_VLEN { H5T_COMPOUND {
            H5T_STD_I32LE "_HDF5_dataset_index_";
            H5T_STD_I32LE "_HDF5_instance_index_";
         }}}
         DATASPACE  SIMPLE { ( 2 ) / ( 2 ) }
         DATA {
         (0): (({
                     1,
                     0
                  }, {
                     1,
                     1
                  }, {
                     1,
                     7
                  }, {
                     1,
                     11
                  }, {
                     1,
                     5
                  }), ({
                     1,
                     1
                  }, {
                     1,
                     2
                  }, {
                     1,
                     7
                  }), ({
                     1,
                     2
                  }, {
                     1,
                     9
                  }, {
                     1,
                     13
                  }, {
                     1,
                     7
                  })),
         (1): (({
                     1,
                     2
                  }, {
                     1,
                     4
                  }, {
                     1,
```

```
                9
            }), ({
                1,
                5
            }, {
                1,
                11
            }, {
                1,
                7
            }, {
                1,
                13
            }, {
                1,
                16
            }, {
                1,
                10
            }))
    }
}
DATASET "Land_survey_instances" {
    DATATYPE  "/Geometry_encoding/Land_survey"
    DATASPACE  SIMPLE { ( 1 ) / ( 1 ) }
    DATA {
    (0): {
          15,
          25,
          "Norway",
          {
              0x01,
              DATASET 17632 /Geometry_population/
              Land_survey_objects/Aggr-properties-1 ,
              ()
          },
          {
              0x00,
              NULL,
              (((100, 101.1, 102.2, 103.3, 104.4),
              (200, 201.1, 202.2), (300, 301.1, 302.2, 303.3)),
              ((400, 401.1, 401.2),
              (500, 501.1, 502.2, 503.3, 504.4, 505.5)))
          }
      }
    }
  }
}
GROUP "Line_objects" {
    DATASET "Line_instances" {
        DATATYPE  "/Geometry_encoding/Line"
        DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
        DATA {
        (0): {
              7,
              4,
              {
                  0,
                  0
              },
              {
```

```
                0,
                1
            },
            {
                2,
                "",
                RED
            }
        },
    (1): {
            7,
            5,
            {
                0,
                1
            },
            {
                0,
                2
            },
            {
                2,
                "",
                BLUE
            }
        },
    (2): {
            7,
            6,
            {
                0,
                2
            },
            {
                0,
                3
            },
            {
                1,
                "Red line",
                VVOID
            }
        },
    (3): {
            7,
            7,
            {
                0,
                3
            },
            {
                0,
                0
            },
            {
                1,
                "Blue line",
                VVOID
            }
        }
    }
```

```
   }
}
GROUP "Many_Point_objects" {
   DATASET "Many_Point_instances" {
      DATATYPE  "/Geometry_encoding/Point"
      DATASPACE  SIMPLE { ( 25 ) / ( 25 ) }
      DATA {
      (0): {
            7,
            0,
            0,
            0
         },
      (1): {
            7,
            1,
            1000,
            0
         },
      (2): {
            7,
            2,
            2000,
            0
         },
      (3): {
            7,
            3,
            3000,
            0
         },
      (4): {
            7,
            4,
            4000,
            0
         },
      (5): {
            7,
            5,
            0,
            1000
         },
      (6): {
            7,
            6,
            1000,
            1000
         },
      (7): {
            7,
            7,
            2000,
            1000
         },
      (8): {
            7,
            8,
            3000,
            1000
         },
```

```
(9): {
      7,
      9,
      4000,
      1000
   },
(10): {
      7,
      10,
      0,
      2000
   },
(11): {
      7,
      11,
      1000,
      2000
   },
(12): {
      7,
      12,
      2000,
      2000
   },
(13): {
      7,
      13,
      3000,
      2000
   },
(14): {
      7,
      14,
      4000,
      2000
   },
(15): {
      7,
      15,
      0,
      3000
   },
(16): {
      7,
      16,
      1000,
      3000
   },
(17): {
      7,
      17,
      2000,
      3000
   },
(18): {
      7,
      18,
      3000,
      3000
   },
(19): {
```

```
            7,
            19,
            4000,
            3000
         },
      (20): {
            7,
            20,
            0,
            4000
         },
      (21): {
            7,
            21,
            1000,
            4000
         },
      (22): {
            7,
            22,
            2000,
            4000
         },
      (23): {
            7,
            23,
            3000,
            4000
         },
      (24): {
            7,
            24,
            4000,
            4000
         }
      }
   }
}
GROUP "Point_objects" {
   DATASET "Point_instances" {
      DATATYPE  "/Geometry_encoding/Point"
      DATASPACE  SIMPLE { ( 4 ) / ( 4 ) }
      DATA {
      (0): {
            7,
            0,
            0,
            0
         },
      (1): {
            7,
            1,
            100,
            0
         },
      (2): {
            7,
            2,
            100,
            100
         },
```

```
            (3): {
                  7,
                  3,
                  0,
                  100
               }
            }
         }
      }
   }
}
}
```

С.10 Полная программа, выполняющая все примеры настоящего стандарта

```
// ---------------------------------------------------------------------
// Main program that executes the examples found in the
// ISO-10303-P26 specification.
//
// In addition some dump functions are supplied, but these are not
// complete in the sense that all aspects of HDF5 are covered.
// Please consult the HDF5 documentation for those aspects that are
// missing.
// Please make use of the h5dump utility offered by the HDF group for
// a complete dump.
//
// Note:
// An application should close a datatype, dataspace, or dataset
// object once it is no longer needed.
// H5Tclose(datatype)
// H5Dclose(dataset)
// H5Sclose(dataspace)
// In this small prototype the closing of such objects has been
/  omitted.
// Reclaim of memory has also been omitted.
// Such a practice is certainly not recommended for a real application
// Please consult the HDF5 documentation for the details.
//---------------------------------------------------------------------

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "hdf5.h"
#include "examples.h"

long dump_dataset(hid_t mydataset);
long dump_compound(hid_t mycompound_type, hsize_t curr_offset,
char *mybuffer);
long dump_instance_by_reference(hid_t dataset_tid, long dataset_index,
long instance_index);
long dump_aggr_by_reference(hid_t obj_tid, hobj_ref_t myref);
long dump_vlen_data(hid_t datatype, long members, char *vlen_data);

void error_trap(char *file_name, long lineno);

//-------------------------------------------------------------------
#define HDF5_FILE "O:/projects/SimDM/HDF5/data/HDF5_files/example.h5"
#define REPORT_FILE "O:/projects/SimDM/HDF5/data/HDF5_files/rep_file.txt"

FILE        *rep_file;
```

```
extern      hid_t file_id;
extern char *defined_entity_names[NUMBER_OF_ENTITIES] =
{"Point","Many_Point","Line","Land_survey"};
extern char *entity_names[NUMBER_OF_ENTITIES];

hid_t obj_ref_tid;
long  level = 0;
long  current_line;
char  *current_file;
long  dump_in_aggr_mode = 0;

#define GOTO_ERR\
   {\
      current_file = __FILE__;\
      current_line = __LINE__;\
      error_trap(current_file,current_line);\
      goto err;\
   }

//
//-------------------------------------------------------------------
//
void error_trap(char *file_name, long lineno)
{
      fprintf(rep_file,"\nERROR trapped in '%s' at line: %lu",file_name,
      lineno);
}
void indent(void)
{
  long ix;

  fprintf(rep_file,"\n");
  for(ix = 0; ix< level; ix++) {
    fprintf(rep_file," ");
  }
}

void indent_nlf(void)
{
  long ix;

  for(ix = 0; ix< level; ix++) {
    fprintf(rep_file," ");
  }
}

//-------------------------------------------------------------------
//  MAIN program
//-------------------------------------------------------------------
int main(int argc, char* argv[])
{
   herr_t  mystatus;
   hid_t   dataset;
   hid_t   group_id;
   hid_t   p_tid;
   hid_t   l_tid;
   hid_t   lls_tid;
   hid_t   eenum_col_tid;
   hid_t   cc_tid;
   hid_t   pop_id;
   hid_t   attr_tid;
```

```
   hid_t   attr_type_tid;
   hid_t   native_type_tid;

   obj_ref_tid   = H5Tcopy(H5T_STD_REF_OBJ);

   rep_file = fopen(REPORT_FILE,"w");
   fprintf(rep_file,"\nWriting %s",REPORT_FILE);

//
// Create HDF5 file
//
   file_id = H5Fcreate(HDF5_FILE, H5F_ACC_TRUNC, H5P_DEFAULT,
   H5P_DEFAULT);
//
// Create group for schemata
//
   if(mystatus = create_schema_group(file_id, "Geometry_encoding",
   "Geometry_schema", &group_id)) GOTO_ERR;
//
// Create group for the population
//
   if(mystatus = create_pop_group(file_id, "Geometry_population",
   "Geometry_schema", &pop_id)) GOTO_ERR;
   //
// HDF5 definitions
//
   if(mystatus = define_instance_reference(group_id)) GOTO_ERR;
   if(mystatus = define_compound_point(group_id,&p_tid)) GOTO_ERR;
   if(mystatus = define_colour_enumeration(group_id, &eenum_col_tid))
   GOTO_ERR ;
   if(mystatus = define_compound_colour(group_id, eenum_col_tid,
   &cc_tid)) GOTO_ERR;
   if(mystatus = define_compound_line(group_id,cc_tid,&l_tid)) GOTO_ERR;
   if(mystatus = define_compound_land_survey(group_id, &lls_tid))
   GOTO_ERR;
//
// HDF5 population
//
   if(mystatus = create_ppoints(pop_id, p_tid)) GOTO_ERR;
   if(mystatus = create_llines(pop_id, l_tid)) GOTO_ERR;
   if(mystatus = create_survey(pop_id, lls_tid, p_tid)) GOTO_ERR;
//
// Dump HDF5 population
//
   attr_tid = H5Aopen_name(pop_id, "_10303_26_data_set_names");
   if(attr_tid > 0){
   attr_type_tid = H5Aget_type(attr_tid);
   native_type_tid = H5Tget_native_type(attr_type_tid,H5T_DIR_ASCEND);

   if(mystatus = H5Aread(attr_tid, native_type_tid, entity_names))
   goto err;
   if(mystatus = H5Aclose(attr_tid)) goto err;
   if(mystatus = H5Tclose(attr_type_tid)) goto err;
   if(mystatus = H5Tclose(native_type_tid)) goto err;
   } else {
     mystatus = -1;
     goto err;
   }

   fprintf(rep_file,"\n\n==========================================
```

```
   ========================================");
   dataset = H5Dopen(file_id, "/Geometry_population/Line_objects/
   Line_instances", H5P_DEFAULT);
   if( mystatus = dump_dataset(dataset)) GOTO_ERR;
   if(mystatus = H5Dclose(dataset)) GOTO_ERR;
   fprintf(rep_file,"\n\n=========================================
   ========================================");
   dataset = H5Dopen(file_id, "/Geometry_population/Land_survey_objects/
   Land_survey_instances", H5P_DEFAULT);
   if(mystatus = dump_dataset(dataset)) GOTO_ERR;
   if(mystatus = H5Dclose(dataset)) GOTO_ERR;

   if(mystatus = H5Fclose(file_id)) GOTO_ERR;
   goto ret;

err:
   fprintf(rep_file,"\n\nERROR caught");
   mystatus = H5Eprint2(H5E_DEFAULT, rep_file);

ret:
   fprintf(rep_file,"\n\nClosing report file");
   fclose(rep_file);
   return(mystatus);
}

//-------------------------------------------------------------------
// Dump dataset
//-------------------------------------------------------------------

long dump_dataset(hid_t mydataset)
{
   herr_t      rstat;
   hid_t       mydataset_datatype;
   H5T_class_t mydataset_datatype_class;
   hsize_t     mydataset_datatype_size;
   hid_t       mydataset_filespace;
   hssize_t    mydataset_noel;
   hvl_t       myvlen;
   long        ix;
   char       *mybuffer;
   long        myinteger;
   double      myreal;

//
// Get the datasets datatype, class, space and size
//
   mydataset_datatype         = H5Dget_type(mydataset);
   mydataset_datatype_class   = H5Tget_class(mydataset_datatype);
   mydataset_datatype_size    = H5Tget_size(mydataset_datatype);
   mydataset_filespace        = H5Dget_space(mydataset);
//
// Read the dataset
//
   mydataset_noel = H5Sget_simple_extent_npoints(mydataset_filespace);
   mybuffer = (char*) malloc((size_t)
   (mydataset_datatype_size*mydataset_noel));
   if(mydataset_datatype_class == H5T_STRING){
     if(rstat  = H5Dread(mydataset, mydataset_datatype,
     mydataset_filespace, mydataset_filespace, H5P_DEFAULT, &mybuffer))
```

```
      GOTO_ERR;
    } else {
      if(rstat  = H5Dread(mydataset, mydataset_datatype,
      mydataset_filespace, mydataset_filespace, H5P_DEFAULT, mybuffer))
      GOTO_ERR;
    }
//
// Handle each element according to its datatype
//
    for(ix = 0; ix< mydataset_noel; ix++) {
      if(!dump_in_aggr_mode){
        fprintf(rep_file,"\n--(%d)",ix);
      }

      switch(mydataset_datatype_class){
        case H5T_INTEGER:
          memcpy(&myinteger,mybuffer,sizeof(long));
          mybuffer += sizeof(long);
          indent();
          fprintf(rep_file,"%lu",myinteger);
        break;
        case H5T_FLOAT:
          memcpy(&myreal,mybuffer,sizeof(double));
          mybuffer += sizeof(double);
          indent();
          fprintf(rep_file,"%f",myreal);
        break;
        case H5T_STRING:
          indent();
          fprintf(rep_file,"%s",mybuffer);
          mybuffer += sizeof(char*);
        break;
        case H5T_COMPOUND:
          ++level;
          indent();
          if(rstat = dump_compound(mydataset_datatype,0,mybuffer))
          GOTO_ERR;
          mybuffer += mydataset_datatype_size;
          --level;
          break;
        case H5T_VLEN:
          memcpy(&myvlen,mybuffer,sizeof(hvl_t));
          mybuffer += sizeof(hvl_t);
          if(rstat = dump_vlen_data(mydataset_datatype,myvlen.len,
          myvlen.p)) GOTO_ERR;
        break;
        default:
          indent();
          fprintf(rep_file,"OOPS! Unhandled datatype");
      }
    }
err:
    return(rstat);
}


//-----------------------------------------------------------------
// Dump Compound type instance
//-----------------------------------------------------------------

long dump_compound(hid_t mycompound_type, hsize_t curr_offset,
```

```
char *mybuffer)
{
   hsize_t        mycompound_type_size;
   int            mycompound_type_members;
   long           dataset_index;
   long           instance_index;
   long           obj_ref_or_vlen;
   long           jx;
   long           len;
   herr_t         rstat;
   char           *mymember_name;
   hvl_t           myvlen;
   hsize_t         mymember_offset;
   hsize_t         myoffset;
   hsize_t         mymember_type_size;
   hid_t           mymember_type;
   hid_t           dataset_tid;
   H5T_class_t     mymember_class;
   long            myinteger;
   double          myreal;
   short           myenumval;
   char            *mystring;
   char            enum_name[10];
   hobj_ref_t      obj_ref;
   enum colors_t   mycolor;
   char            dataset_name[100];

   rstat = 0;

   mymember_name      = H5Tget_member_name(mycompound_type,0);
   if(!strcmp(mymember_name,"_HDF5_dataset_index_")){
     memcpy(&dataset_index,mybuffer+curr_offset,sizeof(long));
     memcpy(&instance_index,mybuffer+curr_offset+(sizeof(long)),sizeof(long));
     strcpy(dataset_name,"/Geometry_population/");
     strcat(dataset_name,entity_names[dataset_index]);
     strcat(dataset_name,"_objects/");
     strcat(dataset_name,entity_names[dataset_index]);
     strcat(dataset_name,"_instances");
     dataset_tid = H5Dopen(file_id, dataset_name, H5P_DEFAULT);
     if(rstat = dump_instance_by_reference(dataset_tid,dataset_index,
     instance_index)) GOTO_ERR;
     if(rstat = H5Dclose(dataset_tid)) GOTO_ERR;
     return(rstat);
   }

   if(!strcmp(mymember_name,"obj_ref_or_vlen")){
     obj_ref_or_vlen = (long)*(mybuffer +curr_offset);
     if(obj_ref_or_vlen == 1){
       curr_offset += sizeof(char);
       memcpy(&obj_ref,mybuffer + curr_offset,sizeof(hobj_ref_t));
       if(rstat = dump_aggr_by_reference(mycompound_type,obj_ref))
       GOTO_ERR;
       return(rstat);
     }
     if(obj_ref_or_vlen == 0){
       curr_offset += sizeof(char) + sizeof(hobj_ref_t);
       memcpy(&myvlen,mybuffer + curr_offset,sizeof(hvl_t));
       mymember_type = H5Tget_member_type(mycompound_type,2);
       if(rstat = dump_vlen_data(mymember_type,myvlen.len,myvlen.p))
       GOTO_ERR;
       return(rstat);
```

```
      }
    }

    ++level;

    len = H5Iget_name(mycompound_type, NULL,0);
    len = H5Iget_name(mycompound_type, dataset_name,len+1);
    fprintf(rep_file,"%s:" ,dataset_name);

    mycompound_type_size    = H5Tget_size(mycompound_type);
    mycompound_type_members = H5Tget_nmembers(mycompound_type);
//
// Handle membmer by member
//
    for(jx = 0; jx< mycompound_type_members; jx++) {
      indent();indent_nlf();
//
// Get the characteristics of the member
//
      mymember_name      = H5Tget_member_name(mycompound_type,jx);
      mymember_type      = H5Tget_member_type(mycompound_type,jx);
      mymember_class     = H5Tget_member_class(mycompound_type,jx);
      mymember_offset    = H5Tget_member_offset(mycompound_type,jx);
      mymember_type_size = H5Tget_size(mymember_type);
      myoffset = curr_offset + mymember_offset;
//
// Branch on datatype.
//
      switch(mymember_class){
      case H5T_INTEGER:
        memcpy(&myinteger,mybuffer + myoffset,sizeof(long));
        fprintf(rep_file,"%s: %lu",mymember_name,myinteger);
      break;
      case H5T_FLOAT:
        memcpy(&myreal,mybuffer + myoffset,sizeof(double));
        fprintf(rep_file,"%s: %f",mymember_name,myreal);
      break;
      case H5T_REFERENCE:
        if(H5Tequal(obj_ref_tid,mymember_type)){
          fprintf(rep_file,"%s: ",mymember_name);
           memcpy(&obj_ref,mybuffer + myoffset,sizeof(hobj_ref_t));
           if(rstat = dump_aggr_by_reference(mycompound_type,obj_ref))
           GOTO_ERR;
        }
      break;
      case H5T_STRING:
        memcpy(&mystring,mybuffer + myoffset,sizeof(char*));
        fprintf(rep_file,"%s: %s",mymember_name,mystring);
      break;
      case H5T_COMPOUND:
        fprintf(rep_file,"%s: ",mymember_name);
        if(rstat = dump_compound(mymember_type,myoffset,mybuffer))
        GOTO_ERR;
      break;
      case H5T_ENUM:
        fprintf(rep_file,"%s: ",mymember_name);
        memcpy(&mycolor,mybuffer + myoffset,sizeof(enum colors_t));
        if(rstat = H5Tenum_nameof(mymember_type,&mycolor,enum_name,10))
        GOTO_ERR;
        if(rstat = H5Tenum_valueof(mymember_type,enum_name,&myenumval))
```

```
        GOTO_ERR;
        fprintf(rep_file,"%d : %s",myenumval,enum_name);
      break;
      case H5T_VLEN:
        memcpy(&myvlen,mybuffer + myoffset,sizeof(hvl_t));
        if(rstat = dump_vlen_data(mymember_type,myvlen.len,myvlen.p))
        GOTO_ERR;
        break;
      default:
        fprintf(rep_file,"\nOOPS! Unhandeled datatype");
    }
  }
  --level;
err:
   return(rstat);
}
//-------------------------------------------------------------------
// Dump referenced aggregate
//-------------------------------------------------------------------
long dump_aggr_by_reference(hid_t obj_tid, hobj_ref_t myref)
{
   long        mystatus;
   hid_t       dataset_reg;

   mystatus = 0;
   dump_in_aggr_mode = 1;
   ++level;
//
// Get the reference
//
   dataset_reg = H5Rdereference(obj_tid, H5R_OBJECT,&myref);
   if(dataset_reg < 0){
     mystatus = -1;
     GOTO_ERR;
   }
//
// Dump the refered aggregate
//
   mystatus = dump_dataset(dataset_reg);
   --level;

err:
   dump_in_aggr_mode = 0;
   return(mystatus);
}
//-------------------------------------------------------------------
// Dump referenced instance
//-------------------------------------------------------------------
long dump_instance_by_reference(hid_t dataset_tid, long dataset_index,
long instance_index)
{
   long        rstat;
   hid_t       mydataset_filespace;
   hid_t       dataset_datatype;
   size_t      dataset_datatype_size;
   hsize_t     mydim[1];
   hid_t       mymem_space;
   hsize_t     mycoord[1][1];
   char       *dataset_buffer;
//
// Read the selected instance
```

```
//
   mydim[0] = 1;
   mymem_space = H5Screate_simple(1, mydim, NULL);
   dataset_datatype = H5Dget_type(dataset_tid);
   dataset_datatype_size = H5Tget_size(dataset_datatype);
   dataset_buffer = (char*) malloc(dataset_datatype_size);

   mydataset_filespace =  H5Dget_space(dataset_tid);

   mycoord[0][0] = instance_index;
   if(rstat = H5Sselect_elements(mydataset_filespace, H5S_SELECT_SET,
   1, (const hsize_t *)&mycoord)) GOTO_ERR;
   if(rstat  = H5Dread(dataset_tid, dataset_datatype, mymem_space,
   mydataset_filespace, H5P_DEFAULT, dataset_buffer)) GOTO_ERR;
//
// Dump the instance from buffer
//
   if(rstat = dump_compound(dataset_datatype,0,dataset_buffer)) GOTO_ERR;

   free(dataset_buffer);

err:
   return(rstat);
}

//-----------------------------------------------------------------
// Dump VLEN data
//-----------------------------------------------------------------
long dump_vlen_data(hid_t datatype, long members, char *vlen_data)
{
   long          rstat,i;
   long          super_size;
   hid_t         super_tid;
   H5T_class_t   super_class;
   double        myreal;
   long          myinteger;
   short         myenumval;
   enum colors_t mycolor;
   hobj_ref_t    obj_ref;
   char          enum_name[10];
   hvl_t         myvlen;
   char          *p_myvlen;

   rstat = 0;
   p_myvlen = vlen_data;
//
// Get type of VLEN data
//
   super_tid   = H5Tget_super(datatype);
   super_class = H5Tget_class(super_tid);
   super_size  = H5Tget_size(super_tid);
   ++level;++level;
   indent();
   fprintf(rep_file,"(");
//
// Branch on type for each element
//
   for(i=0; i<members; ++i){
     switch(super_class){
       case H5T_INTEGER:
         memcpy(&myinteger,p_myvlen,sizeof(long));
```

```
  p_myvlen += sizeof(long);
  fprintf(rep_file," %lu ",myinteger);
break;
case H5T_FLOAT:
  memcpy(&myreal,p_myvlen,sizeof(double));
  p_myvlen += sizeof(double);
  fprintf(rep_file," %f ",myreal);
break;
case H5T_VLEN:
  memcpy(&myvlen,p_myvlen,sizeof(hvl_t));
  if(rstat = dump_vlen_data(super_tid,myvlen.len,myvlen.p))
  GOTO_ERR;
  p_myvlen += sizeof(hvl_t);
break;
case H5T_COMPOUND:
  indent();
  if(rstat = dump_compound(super_tid,0,p_myvlen)) GOTO_ERR;
  p_myvlen += super_size;
break;
case H5T_REFERENCE:
  if(H5Tequal(obj_ref_tid,super_tid)){
    indent();
    memcpy(&obj_ref,p_myvlen,sizeof(hobj_ref_t));
    if(rstat = dump_aggr_by_reference(super_tid,obj_ref))
    GOTO_ERR;
    p_myvlen += super_size;
  }
break;
case H5T_ENUM:
  memcpy(&mycolor,p_myvlen,sizeof(enum colors_t));
  p_myvlen += sizeof(enum colors_t);
        if(rstat = H5Tenum_nameof(super_tid,&mycolor,enum_name,10))
        GOTO_ERR;
        if(rstat = H5Tenum_valueof(super_tid,enum_name,&myenumval))
        GOTO_ERR;
        fprintf(rep_file,"%d : %s",myenumval,enum_name);
      break;
      default:
        fprintf(rep_file,"\nOOPS! Unhandeled datatype");
    }
  }
  indent();
  fprintf(rep_file,")");
  --level;--level;

err:
   return(rstat);
}
//
// Get dataset index
//
long get_dataset_index(char *entity_name)
{
  long dataset_index,i;

  dataset_index = -1;
  for(i=0; i<NUMBER_OF_ENTITIES; ++i){
    if(!strcmp(defined_entity_names[i],entity_name)){
      return(i);
    }
  }
```

```
  return(dataset_index);
}
```

Содержимое файла результатов после выполнения полной программы:

```
--(0)
 /Geometry_encoding/Line:
   set_unset_bitmap: 7
   Entity-Instance-Identifier: 4
   startp: /Geometry_encoding/Point:
     set_unset_bitmap: 7
     Entity-Instance-Identifier: 0
     x: 0.000000
     y: 0.000000
   endp: /Geometry_encoding/Point:
     set_unset_bitmap: 7
     Entity-Instance-Identifier: 1
     x: 100.000000
     y: 0.000000
   colour: /Geometry_encoding/CColour:
     select_bitmap: 2
     s_colour:
     e_colour: 1 : RED
--(1)
 /Geometry_encoding/Line:
   set_unset_bitmap: 7
   Entity-Instance-Identifier: 5
   startp: /Geometry_encoding/Point:
     set_unset_bitmap: 7
     Entity-Instance-Identifier: 1
     x: 100.000000
     y: 0.000000
   endp: /Geometry_encoding/Point:
     set_unset_bitmap: 7
     Entity-Instance-Identifier: 2
     x: 100.000000
     y: 100.000000
   colour: /Geometry_encoding/CColour:
     select_bitmap: 2
     s_colour:
     e_colour: 3 : BLUE
--(2)
 /Geometry_encoding/Line:
   set_unset_bitmap: 7
   Entity-Instance-Identifier: 6
   startp: /Geometry_encoding/Point:
     set_unset_bitmap: 7
     Entity-Instance-Identifier: 2
     x: 100.000000
     y: 100.000000
   endp: /Geometry_encoding/Point:
     set_unset_bitmap: 7
     Entity-Instance-Identifier: 3
     x: 0.000000
     y: 100.000000
   colour: /Geometry_encoding/CColour:
     select_bitmap: 1
     s_colour: Red line
     e_colour: 0 : VVOID
--(3)
```

```
/Geometry_encoding/Line:
  set_unset_bitmap: 7
  Entity-Instance-Identifier: 7
  startp: /Geometry_encoding/Point:
    set_unset_bitmap: 7
    Entity-Instance-Identifier: 3
    x: 0.000000
    y: 100.000000
  endp: /Geometry_encoding/Point:
    set_unset_bitmap: 7
    Entity-Instance-Identifier: 0
    x: 0.000000
    y: 0.000000
  colour: /Geometry_encoding/CColour:
    select_bitmap: 1
    s_colour: Blue line
    e_colour: 0 : VVOID
```

```
--(0)
/Geometry_encoding/Land_survey:
  set_unset_bitmap: 15
  Entity-Instance-Identifier: 25
  country: Norway
  properties:
   (
    (
    /Geometry_encoding/Point:
          set_unset_bitmap: 7
          Entity-Instance-Identifier: 0
          x: 0.000000
          y: 0.000000
    /Geometry_encoding/Point:
          set_unset_bitmap: 7
          Entity-Instance-Identifier: 1
          x: 1000.000000
          y: 0.000000
    /Geometry_encoding/Point:
          set_unset_bitmap: 7
          Entity-Instance-Identifier: 7
          x: 2000.000000
          y: 1000.000000
    /Geometry_encoding/Point:
          set_unset_bitmap: 7
          Entity-Instance-Identifier: 11
          x: 1000.000000
          y: 2000.000000
    /Geometry_encoding/Point:
          set_unset_bitmap: 7
          Entity-Instance-Identifier: 5
          x: 0.000000
          y: 1000.000000
    )
    (
    /Geometry_encoding/Point:
          set_unset_bitmap: 7
          Entity-Instance-Identifier: 1
          x: 1000.000000
          y: 0.000000
    /Geometry_encoding/Point:
```

```
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 2
            x: 2000.000000
            y: 0.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 7
            x: 2000.000000
            y: 1000.000000
    )
    (
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 2
            x: 2000.000000
            y: 0.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 9
            x: 4000.000000
            y: 1000.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 13
            x: 3000.000000
            y: 2000.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 7
            x: 2000.000000
            y: 1000.000000
    )
  )
  (
    (
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 2
            x: 2000.000000
            y: 0.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 4
            x: 4000.000000
            y: 0.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 9
            x: 4000.000000
            y: 1000.000000
    )
    (
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 5
            x: 0.000000
            y: 1000.000000
    /Geometry_encoding/Point:
            set_unset_bitmap: 7
            Entity-Instance-Identifier: 11
```

```
        x: 1000.000000
        y: 2000.000000
  /Geometry_encoding/Point:
        set_unset_bitmap: 7
        Entity-Instance-Identifier: 7
        x: 2000.000000
        y: 1000.000000
  /Geometry_encoding/Point:
        set_unset_bitmap: 7
        Entity-Instance-Identifier: 13
        x: 3000.000000
        y: 2000.000000
  /Geometry_encoding/Point:
        set_unset_bitmap: 7
        Entity-Instance-Identifier: 16
        x: 1000.000000
        y: 3000.000000
  /Geometry_encoding/Point:
        set_unset_bitmap: 7
        Entity-Instance-Identifier: 10
        x: 0.000000
        y: 2000.000000
  )
 )
altitudes:
(
 (
  ( 100.000000  101.100000  102.200000  103.300000  104.400000
  )
  ( 200.000000  201.100000  202.200000
  )
  ( 300.000000  301.100000  302.200000  303.300000
  )
 )
 (
  ( 400.000000  401.100000  401.200000
  )
  ( 500.000000  501.100000  502.200000  503.300000  504.400000  505.500000
  )
 )
)
```

Приложение ДА
(справочное)

Сведения о соответствии ссылочных международных стандартов национальным стандартам Российской Федерации

Таблица ДА.1

Обозначение ссылочного международного стандарта	Степень соответствия	Обозначение и наименование соответствующего национального стандарта
ИСО 639-2	–	*
ИСО 8601	IDT	ГОСТ ИСО 8601–2001 «Система стандартов по информации, библиотечному и издательскому делу. Представление дат и времени. Общие требования»
ИСО 10303-1	IDT	ГОСТ Р ИСО 10303-1–99 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 1. Общие представления и основополагающие принципы»
ИСО 10303-11	IDT	ГОСТ Р ИСО 10303-11–2009 «Системы автоматизации производства и их интеграция. Представление данных об изделии и обмен этими данными. Часть 11. Методы описания. Справочное руководство по языку EXPRESS»

* Соответствующий национальный стандарт отсутствует. До его утверждения рекомендуется использовать перевод на русский язык данного международного стандарта (документа). Перевод данного международного стандарта (документа) находится в Федеральном информационном фонде технических регламентов и стандартов.

Примечание – В настоящей таблице использовано следующее условное обозначение степени соответствия стандартов:

IDT – идентичные стандарты.

Библиография

[1] HDF5: API Specification Reference Manual, Release 1.8. Available from World Wide Web: http://www.hdfgroup.org/HDF5/doc/RM/RM_H5Front.html
[2] HDF5 User's Guide, HDF5 Release 1.8. Available from World Wide Web: http://www.hdfgroup.org/HDF5/doc/UG/index.html
[3] Introduction to HDF5. Available from World Wide Web: http://hdfgroup.com/HDF5/doc/ H5.intro.html
[4] HDF5 Tutorial. Available from World Wide Web: http://hdfgroup.com/HDF5/Tutor/ index.html
[5] ISO/IEC 8824-1, Information technology — Abstract Syntax Notation One (ASN.1) — Specification of basic notation
[6] ISO 10303-41, Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support
[7] ISO 10303-43, Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures
[8] ISO 10303-50, Industrial automation systems and integration — Product data representation and exchange — Part 50: Integrated generic resource: Mathematical constructs
[9] ISO 10303-51, Industrial automation systems and integration — Product data representation and exchange — Part 51: Integrated generic resource: Mathematical representation

УДК 656.072:681.3:006.354 ОКС 25.040.40

Ключевые слова: автоматизация производства, средства автоматизации, интеграция систем автоматизации, промышленные изделия, данные об изделиях, представление данных, обмен данными, методы реализации, язык EXPRESS, двоичное представление данных

Редактор *В.А. Павлов*

Корректор *П.М. Смирнов*

Компьютерная верстка *Д. М. Кульчицкого*

Подписано в печать 08.02.2016. Формат 60x84 1/8.

Усл. печ. л. 8,84. Тираж 30 экз. Зак. 3868.

Подготовлено на основе электронной версии, предоставленной разработчиком стандарта

ФГУП «СТАНДАРТИНФОРМ»

123995 Москва, Гранатный пер., 4.

www.gostinfo.ru info@gostinfo.ru