ГОСТ 34.1702.3—92
(ИСО 8651—3—88)

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

ИНФОРМАЦИОННАЯ ТЕХНОЛОГИЯ

МАШИННАЯ ГРАФИКА

СВЯЗЬ ЯДРА ГРАФИЧЕСКОЙ СИСТЕМЫ С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ АДА

Издание официальное

МОСКВА
1993

gost.e-ecolog.ru

© Издательство стандартов, 1993

к ГОСТ Р 34.1702.3—92 (ИСО 8651-3—88) Информационная технология. Машинная графика. Связь ядра графической системы с языком программирования Ада

В каком месте	Напечатано	Должно быть
Обложка, титульный лист	ГОСТ 34.1702.3—92	ГОСТ Р 34.1702.3—92

(ИУС № 12 2007 г.)

УДК 681.3.04:006.354 Группа П85

ГОСУДАРСТВЕННЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ

Информационная технология

МАШИННАЯ ГРАФИКА

Связь ядра графической системы с языком программирования Ада

Information technology. Computer graphics — Graphical. Kernel System (GKS) language bindings — Ada

ГОСТ Р 34.1702.3—92

(ИСО 8651—3—88)

ОКСТУ 0031

Дата введения 01.01.94

Настоящий стандарт устанавливает правила привязки ядра графической системы (ЯГС) (ГОСТ 27817) к языку программирования Ада (ИСО 8651—3) и определяет:

имена и списки параметров процедур на языке Ада, соответствующие функциям ЯГС;

имена типов данных ЯГС в языке Ада;

структуры данных ЯГС в языке Ада;

имена функций обработки ошибок.

Настоящий стандарт не устанавливает:

структуры и методы реализации ЯГС;

требования к операционной среде и оборудованию;

методы связи ЯГС с другими языками программирования, отличными от языка Ада.

2. ССЫЛКИ

В качестве описания правил привязки ЯГС к языку программирования Ада использован международный стандарт ИСО 8651—3—88, приведенный в приложении.

Издание официальное

★

Настоящий стандарт не может быть полностью или частично воспроизведен, тиражирован и распространен без разрешения Госстандарта России

ПРИЛОЖЕНИЕ

МЕЖДУНАРОДНЫЙ СТАНДАРТ

Системы обработки информации

МАШИННАЯ ГРАФИКА

Связь ядра графической системы с языком программирования Ада

ИСО 8651—3—88

ПРЕДИСЛОВИЕ

Международный стандарт ИСО 8651—3 разработан Техническим комитетом ИСО/СТК 1 «Системы обработки информации».

ИСО 8651 состоит из следующих частей под общим заголовком «Системы обработки информации. Машинная графика. Связь ядра графической системы (ЯГС) с языками программирования:

Часть 1. Фортран.

Часть 2. Паскаль.

Часть 3. Ада.

0. ВВЕДЕНИЕ

Функциональное описание ядра графической системы (ЯГС), содержащееся в ГОСТ 27817 (ИСО 7942), сформулировано независимым от языка программирования способом и должно быть окружено слоем, зависящим от языка программирования (привязка к языку) для использования с определенным языком программирования.

Цель настоящего стандарта части ИСО 8651 — определение стандартной привязки к языку программирования Ада.

1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

ГОСТ 27817 (ИСО 7942) определяет языконезависимое ЯГС. Для включения его в язык программирования ЯГС помещается на

языковозависимый уровень, отвечающий соответствующим положениям этого языка. Настоящая часть ИСО 8651 определяет такой языковозависимый уровень для языка программирования Ада.

2. ССЫЛКИ

ГОСТ 27817 (ИСО 7942—85) Системы обработки информации. Машинная графика. Описание функций ядра графической системы (ЯГС).

ГОСТ 27831 (ИСО 8652—86) Язык программирования Ада.

3. СВЯЗЬ ЯГС С ЯЗЫКОМ АДА

Данная связь не предполагает, что компилятор поддерживает любые свойства языка Ада, которые являются зависящими от реализации, но подразумевает, что компилятор будет способен поддерживать декларации, входящие в связывание GKS/Ада. Не делается никаких предположений относительно машинного представления предопределенных типов чисел в Аде.

Данная связь предполагает, что прикладной программист будет применять имя файла ошибок и идентификатор связи, которые будут иметь формат, приемлемый для реализации Ады.

В данной связи не делается предположения относительно формата строки, определяющей имя файла ошибок или идентификатор связи для устройства или метафайла.

3.1. Условия соответствия стандарту

Данная связь включает правила, определенные в стандарте ГОСТ 27817 (ИСО 7942) со следующими дополнительными требованиями, специально заданными для реализации ЯГС в языке Ада.

Для определения соответствия или несоответствия реализации данному связыванию установлены следующие критерии:

а) реализация ЯГС в Аде соответствует уровню ЯГС, если она точно выполняет декларации для данного уровня ЯГС и более низких уровней, определенных данным связыванием;

б) семантика реализации должна соответствовать стандарту ЯГС с модификациями и расширениями для Ады, установленными в данном документе;

в) модуль, соответствующий реализуемому уровню ЯГС, должен быть доступен в виде блока библиотеки Ады с именем, определенным в данном документе.

3.2. **Включение в язык**

3.2.1. Отображение функций

Все функции ЯГС отображаются в процедуры Ады. Отображение использует «один-в-один» соответствие между функциями ЯГС и процедурами Ады, исключая функции ЯГС «Узнать текущий атрибут примитива» и «Узнать индивидуальный атрибут». Они связаны с отдельной процедурой Ады для каждого из запрашиваемых атрибутов; атрибуты связываются с одной записью.

3.2.2. Реализация и зависимость от компьютера

Существует ряд зависимостей от реализаций и компьютера, связанных с компилятором Ады и используемыми процедурами системы. Это будет влиять на переносимость прикладных программ и их использование ЯГС. Прикладному программисту следует придерживаться принятой практики обеспечения переносимости программ на языке Ада, чтобы избежать возникновения проблем при переносе прикладного программного обеспечения на другую систему. К зависимостям от реализаций относятся управление памятью и процессором.

3.2.3. Обработка ошибок

Справочные функции используют параметры индикаторов для возврата ошибок и не порождают прерываний. В языке Ада прикладная программа должна гарантировать, что индикаторы ошибок проверяются до попыток обращения к другим параметрам, так как некоторые реализации Ады не порождают прерываний, если обнаружено неопределенное значение.

Требования ЯГС по обработке ошибок можно выразить следующим образом.

1. По умолчанию процедура, названная ERROR_HANDLING, будет обеспечивать простую регистрацию ошибок вызовом ERRQR_LOGGING. Она вызывается из функций ЯГС, обнаруживших ошибку.

2. Процедура ERROR_HANDLING может быть заменена пользователем на другую.

Процедура ERROR_HANDLING задается, как библиотечная подпрограмма:

```
with GKS_TYPES;
use GKS_TYPES;
procedure ERROR_HANDLING (ERROR_INDICATOR
                              : in ERROR_NUMBER;
                          GKS_FUNCTION : in STRING;
                             ERROR_FILE : in STRING
                          : =DEFAULT_ERROR_FILE);
```

Процедура ERROR_HANDLING определяется как библиотечная процедура и не декларируется внутри пакета ЯГС.

Данная связь определяет два различных тела для этой программы; каждое должно быть представлено реализацией. Тело по умолчанию — это одна из требуемых ЯГС семантических конструкций. Это просто вызов ERROR_LOGGING и возврат. Функция ЯГС должна быть написана так, чтобы она не поддерживала GKS_ERROR (это требование реализации). Таким образом, в соответствии с правилами языка Ада прерывание распространяется назад к прикладной программе, вызвавшей функцию GKS, в которой обнаружилась ошибка.

Смысл пользовательской замены тела, применяемого по умолчанию, на версию, порождающую прерывание, зависит от менеджера библиотеки Ады. Некоторые реализации поддерживают множество версий тела с одной спецификацией или допускают применение иерархических библиотек с разделением общих блоков. В других реализациях может оказаться необходимым дублировать библиотеку семантических конструкций для каждой версий ERROR HALDING.

Ошибки GKS отображаются в одно прерывание GKS_ERROR, декларированное в пакете GKS. Предполагаемый стиль в обработке ошибок с использованием прерываний состоит в том, чтобы обеспечить обработчик для прерываний GKS_ERROR.

3.2.4. Отображение данных

Простые и составные типы данных ЯГС связываются с различными скалярными и составными типами языка Ада. Ограничения на допустимые значения отображаются, где это возможно, в определениях типов. Общее соотношение между типами данных ЯГС и типами данных Ады следующее:

целые ЯГС отображаются в целые типы Ады;

действительные ЯГС отображаются в типы с плавающей запятой Ады;

строки ЯГС отображаются в тип STRING Ады или в тип, предназначенный для строк переменной длины;

указатели ЯГС отображаются в типы записи Ады;

имена ЯГС отображаются в дискретный тип Ады;

нумерация ЯГС отображается в тип нумерации Ады;

вектора ЯГС отображаются в тип записи Ады;

матрицы (matrix) ЯГС отображаются в типы массив (array) Ады;

списки ЯГС из элементов конкретного типа отображаются в приватный тип Ады, декларированный в общем пакете GKS_LIST_UTILITIES;

массивы ЯГС отображаются либо в тип неограниченной матрицы, либо в тип записи, предоставляемый для массивов переменной длины;

упорядоченные пары ЯГС отображаются в типы записей Ады;

записи данных ЯГС отображаются в приватные типы Ады.

В некоторых случаях набор подпрограмм для обработки записей данных явно определяется данным стандартом. Реализация ЯГС может предоставить другие подпрограммы для манипулирования зависящими от реализации данными.

3.2.5. Многозадачность

В определении языка Ада дается явная поддержка параллельности. Модель организации прохождения задачи включает средства для декларирования и выделения задачи и операции, допускающие межзадачную связь и синхронизацию.

Стандарт ЯГС не требует и не запрещает в реализации защиту от проблем, которые могут возникнуть при асинхронном доступе к структурам данных ЯГС из параллельных задач. В пользовательской документации по реализации ЯГС должна содержаться информация о том, введена ли защита от таких проблем.

Приложение Д представляет собой пособие для тех, кто хочет, чтобы поддерживались многозадачные прикладные программы. Приложение не является составной частью стандарта, а предоставляет дополнительную информацию.

3.2.6. Пакетирование

Стандарт ЯГС определяет девять уровней графической функциональности, где уровень 0а является низшим, а 2с — высшим уровнем. В реализации ЯГС можно сделать систему единой, а можно реализовать каждый уровень отдельно. Для поддержки этой концепции данный стандарт определяет девять пакетов Ады, которые соответствуют каждому уровню ЯГС. Каждый из этих пакетов именуется package GKS is ... end GKS, чтобы обеспечить переносимость прикладных программ для уровней ЯГС. Однако содержимое пакетов различается в зависимости от уровня ЯГС, который они реализуют. Каждый из этих пакетов предоставляет подпрограммы, определенные для их уровней и все подпрограммы, заданные в нижних уровнях, как определено в п. 5.1. С каждым из этих пакетов связаны пакеты типов данных, которые обеспечивают декларации типов для соответствующего уровня, как определено в п. 4.2, а ЯГС определяет прерывания, приведенные в п. 4.3.1. Данные пакеты именуются package GKS_TYPES is ... end GKS_TYPES.

Средства библиотек программ Ады следует использовать для обеспечения отделения уровней. Так графические прикладные

программы на Аде, которые используют ЯГС, должны включать соответствующие пакеты ЯГС, которые предоставляют подпрограмму, типы и прерывания для данного уровня компилированием и включением соответствующей библиотеки Ады, которая содержит данный уровень ЯГС. Например, прикладная программа, которая использует уровень 0а, могла бы выглядеть следующим образом:

```
with GKS;
use GKS_TYPES;
procedure APPLICATION is
begin
null;
end APPLICATION.
```

Далее программа компилируется и связывается с библиотекой программ Ады, которая соответствует уровню 0а.

Данный стандарт определяет и ряд дополнительных блоков Ады. Ими являются общие пакеты:

GKS_COORDINATE_SYSTEM;
GKS_LIST_UTILITIES.

Эти общие пакеты поддерживают типы деклараций в пакете GKS_TYPE, описанном выше. GKS_COORDINATE_SYSTEM представляет собой общий пакет, в котором определен ассортимент типов для поддержания каждой координатной системы ЯГС. GKS_LIST_UTILITIES является также общим пакетом, который предоставляет декларации списков и операции для типов списков, которые соответствуют типам списков ЯГС.

3.2.7. Среда прикладных программ

Прикладная программа, применяющая реализацию ЯГС в языке Ада, должна будет знать среду, в которой находятся и ЯГС и прикладная программа.

Одним таким интерфейсом является библиотека программ Ады. Язык Ада требует, чтобы прикладная программа имела доступ к библиотеке программ, в которой размещается программное обеспечение ЯГС. Справочное руководство по языку Ада ИСО 8652 не определяет, должна быть одна библиотека или их может быть много, как предоставляется доступ к библиотеке и так далее. Пользовательская документация по реализации ЯГС должна содержать информацию о том, где в системе находится библиотека ЯГС и как осуществляется к ней доступ.

Интерфейсы ввода/вывода также являются зависящими от реализации и требуют описания в пользовательской документации. Должны быть включены в документацию интерфейсы с фай-

ловой системой. Особенно это относится к интерфейсу с файлом ошибок ЯГС, а также к памяти метафайла.

3.2.8. Регистрация*

Стандарт ЯГС резервирует различные области величин для регистрации в качестве графических элементов. Зарегистрированные графические элементы будут встроены в Аду (и другие языки программирования). Зарегистрированные встроенные элементы будут согласовываться со связыванием, представленным в данном документе.

4. ТАБЛИЦЫ

4.1. Процедуры

Таблица 1

Сокращения, используемые в именах процедур

ASF	Флаг выборки атрибутов
CHAR	Литера
ESC	Расширение (escape)
GDP	Обобщенный примитив вывода
GKS	Ядро графической системы
GKSM	Метафайл ядра графической системы
ID	Идентификатор
INQ	Справочная функция
MAX	Максимум
UGDP	Незарегистрированный обобщенный примитив вывода
UESC	Незарегистрированное расширение
WS	Станция

Таблица 2

Список процедур, использующих сокращения

ASF	
	INQ_LIST_OF_ASF
	SET_ASF
CHAR	
	INQ_CHAR_BASE_VECTOR
	INQ_CHAR_EXPANSION_FACTOR
	INQ_CHAR_HEIGHT
	INQ_CHAR_WIDTH
	INQ_CHAR_SPACING
	INQ_CHAR_UP_VECTOR

* В соответствии с правилами назначения и работы Органов регистрации в директивах ИСО Совет ИСО для этой части (ИСО 8651) назначил Национальное бюро стандартов (Научно-производственный институт ЭВМ) как орган регистрации A266 Technology Building, Gaithersburg, MD, 20899, USA.

Продолжение табл. 2

	SET_CHAR_EXPANSION_FACTOR SET_CHAR_HEIGHT SET_CHAR_SPACING SET_CHAR_UP_VECTOR
ESC	ESC UESC
GDP	GDP INQ_GDP INQ_LIST_OF_AVAILABLE_GDP UGDP
GKS	CLOSE_GKS EMERGENCY_CLOSE_GKS INQ_LEVEL_OF_GKS OPEN_GKS
GKSM	GET_ITEM_TYPE_FROM_GKSM READ_ITEM_FROM_GKSM WRITE_ITEM_TO_GKSM
ID	INQ_CURRENT_PICK_ID_VALUE SET_PICK_ID
IND	INQ_CHAR_BASE_VECTOR INQ_CHAR_EXPANSION_FACTOR INQ_CHAR_HEIGHT INQ_CHAR_WIDTH INQ_CHAR_SPACING INQ_CHAR_UP_VECTOR INQ_CHOICE_DEVICE_STATE INQ_CLIPPING INQ_COLOUR_FACILITES INQ_COLOUR_REPRESENTATION INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES INQ_CURRENT_PICK_ID_VALUE INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES INQ_DEFAULT_CHOICE_DEVICE_DATA INQ_DEFAULT_DEFERRAL_STATE_VALUES INQ_DEFAULT_LOCATOR_DEVICE_DATA INQ_DEFAULT_PICK_DEVICE_DATA INQ_DEFAULT_STRING_DEVICE_DATA INQ_DEFAULT_STROKE_DEVICE_DATA INQ_DEFAULT_VALUATOR_DEVICE_DATA INQ_DISPLAY_SPACE_SIZE INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES

Продолжение табл. 2

INQ_FILL_AREA_COLOUR_INDEX
INQ_FILL_AREA_FACILITIES
INQ_FILL_AREA_INDEX
INQ_FILL_AREA_INTERIOR_STYLE
INQ_FILL_AREA_REPRESENTATION
INQ_FILL_AREA_STYLE_INDEX
INQ_GDP
INQ_INPUT_QUEUE_OVERFLOW
INQ_LEVEL_OF_GKS
INQ_LIST_OF_ASF
INQ_LINETYPE
INQ_LINEWIDTH_SCALE_FACTOR
INQ_LIST_OF_AVAILABLE_GDP
INQ_LIST_OF_AVAILABLE_WS_TYPE
INQ_LIST_OF_COLOUR_INDICES
INQ_LIST_OF_FILL_AREA_INDICES
INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBER
INQ_LIST_OF_PATTERN_INDICES
INQ_LIST_OF_POLYLINE_INDICES
INQ_LIST_OF_POLYMARKER_INDICES
INQ_LIST_OF_TEXT_INDICES
INQ_LOCATOR_DEVICE_STATE
INQ_MAX_LENGTH_OF_WS_STATE_TABLES
INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER
INQ_MORE_SIMULTANEOUS_EVENTS
INQ_NAME_OF_OPEN_SEGMENT
INQ_NORMALIZATION_TRANSFORMATION
INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES
INQ_OPERATING_STATE_VALUE
INQ_PATTERN_FACILITIES
INQ_PATTERN_HEIGHT_VECTOR
INQ_PATTERN_REFERENCE_POINT
INQ_PATTERN_REPRESENTATION
INQ_PATTERN_WIDTH_VECTOR
INQ_PICK_DEVICE_STATE
INQ_PIXEL
INQ_PIXEL_ARRAY
INQ_PIXEL_ARRAY_DIMENSIONS
INQ_POLYLINE_COLOUR_INDEX
INQ_POLYLINE_FACILITIES
INQ_POLYLINE_INDEX
INQ_POLYLINE_REPRESENTATION
INQ_POLYMARKER_REPRESENTATION
INQ_POLYMARKER_COLOUR_INDEX
INQ_POLYMARKER_INDEX
INQ_POLYMARKER_FACILITIES
INQ_POLYMARKER_SIZE_SCALE_FACTOR

Продолжение табл. 2

	INQ_POLYMARKER_TYPE
	INQ_PREDEFINED_COLOUR_REPRESENTATION
	INQ_PREDEFINED_FILL_AREA_REPRESENTATION
	INQ_PREDEFINED_PATTERN_REPRESENTATION
	INQ_PREDEFINED_POLYLINE_REPRESENTATION
	INQ_PREDEFINED_POLYMARKER_REPRESENTATION
	INQ_PREDEFINED_TEXT_REPRESENTATION
	INQ_SEGMENT_ATTRIBUTES
	INQ_SET_OF_ACTIVE_WS
	INQ_SET_OF_ASSOCIATED_WS
	INQ_SET_OF_OPEN_WS
	INQ_SET_OF_SEGMENT_NAMES_IN_USE
	INQ_SET_OF_SEGMENT_NAMES_ON_WS
	INQ_STRING_DEVICE_STATE
	INQ_STROKE_DEVICE_STATE
	INQ_TEXT_ALIGNMENT
	INQ_TEXT_COLOUR_INDEX
	INQ_TEXT_EXTENT
	INQ_TEXT_FACILITIES
	INQ_TEXT_FONT_AND_PRECISION
	INQ_TEXT_INDEX
	INQ_TEXT_PATH
	INQ_TEXT_REPRESENTATION
	INQ_VALUATOR_DEVICE_STATE
	INQ_WS_CATEGORY
	INQ_WS_CLASSIFICATION
	INQ_WS_CONNECTION_AND_TYPE
	INQ_WS_DEFERRAL_AND_UPDATE_STATES
	INQ_WS_MAX_NUMBER
	INQ_WS_STATE
	INQ_WS_TRANSFORMATION
MAX	
	INQ_MAX_LENGTH_OF_WS_STATE_TABLES
	INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER
	INQ_WS_MAX_NUMBERS
WS	
	ACTIVATE_WS
	ASSOCIATE_SEGMENT_WITH_WS
	CLEAR_WS
	CLOSE_WS
	COPY_SEGMENT_TO_WS
	DEACTIVATE_WS
	DELETE_SEGMENT_FROM_WS
	INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES
	INQ_LIST_OF_AVAILABLE_WS_TYPE
	INQ_MAX_LENGTH_OF_WS_STATE_TABLES
	INQ_SET_OF_ACTIVE_WS
	INQ_SET_OF_ASSOCIATED_WS

Продолжение табл. 2

	INQ_SET_OF_OPEN_WS
	INQ_SET_OF_SEGMENT_NAMES_ON_WS
	INQ_WS_CATEGORY
	INQ_WS_CLASSIFICATION
	INQ_WS_CONNECTION_AND_TYPE
	INQ_WS_DEFERRAL_AND_UPDATE_STATES
	INQ_WS_MAX_NUMBER
	INQ_WS_STATE
	INQ_WS_TRANSFORMATION
	OPEN_WS
	REDRAW_ALL_SEGMENTS_ON_WS
	SET_WS_VIEWPORT
	SET_WS_WINDOW
	UPDATE_WS

Таблица 3

Функции ЯГС и имена соответствующих процедур в Аде

Имя в Аде	Функция ЯГС
ACCUMULATE_TRANSFORMATION_MATRIX	ВЫЧИСЛИТЬ РЕЗУЛЬТИРУЮЩУЮ МАТРИЦУ ПРЕОБРАЗОВАНИЙ
ACTIVATE_WS	АКТИВИРОВАТЬ СТАНЦИЮ
ASSOCIATE_SEGMENT_WITH_WS	СВЯЗАТЬ СЕГМЕНТ СО СТАНЦИЕЙ
AWAIT_EVENT	ОЖИДАТЬ СОБЫТИЕ
CELL_ARRAY	МАТРИЦА ЯЧЕЕК
CLEAR_WS	ОЧИСТИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ
CLOSE_GKS	ЗАКРЫТЬ ЯГС
CLOSE_SEGMENT	ЗАКРЫТЬ СЕГМЕНТ
CLOSE_WS	ЗАКРЫТЬ СТАНЦИЮ
COPY_SEGMENT_TO_WS	ВЫВЕСТИ КОПИЮ СЕГМЕНТА НА СТАНЦИЮ
CREATE_SEGMENT	СОЗДАТЬ СЕГМЕНТ
DEACTIVATE_WS	ДЕАКТИВИРОВАТЬ СТАНЦИЮ
DELETE_SEGMENT	УНИЧТОЖИТЬ СЕГМЕНТ
DELETE_SEGMENT_FROM_WS	УДАЛИТЬ СЕГМЕНТ СО СТАНЦИИ
EMERGENCY_CLOSE_GKS	АВАРИЙНО ЗАКРЫТЬ ЯГС
ERROR_HANDLING	ОБРАБОТАТЬ ОШИБКУ
ERROR_LOGGING	ЗАРЕГИСТРИРОВАТЬ ОШИБКУ
ESCAPE	РАСШИРЕНИЕ
EVALUATE_TRANSFORMATION_MATRIX	СФОРМИРОВАТЬ МАТРИЦУ ПРЕОБРАЗОВАНИЙ
FILL_AREA	ПОЛИГОНАЛЬНАЯ ОБЛАСТЬ
FLUSH_DEVICE_EVENTS	УДАЛИТЬ СОБЫТИЯ ОТ УСТРОЙСТВА
GDP	ОБОБЩЕННЫЙ ПРИМИТИВ ВЫВОДА (ОПВ)
GET_CHOICE	ПОЛУЧИТЬ АЛЬТЕРНАТИВУ
GET_ITEM_TYPE_FROM_GKSM	ПОЛУЧИТЬ ТИП ЗАПИСИ ИЗ ЯГС
GET_LOCATOR	ПОЛУЧИТЬ ПОЗИЦИЮ
GET_PICK	ПОЛУЧИТЬ УКАЗАТЕЛЬ

Продолжение табл. 3

Имя в Ада	Функция ЯГС
GET_STRING	ПОЛУЧИТЬ СТРОКУ
GET_STROKE	ПОЛУЧИТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ПОЗИЦИЙ
GET_VALUATOR	ПОЛУЧИТЬ ЧИСЛО
INITIALISE_CHOICE	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВЫБОРА
INITIALISE_LOCATOR	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА ПОЗИЦИИ
INITIALISE_PICK	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО УКАЗАНИЯ
INITIALISE_STRING	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА СТРОКИ
INITIALISE_STROKE	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ
INITIALISE_VALUATOR	ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА ЧИСЛА
INQ_CHOICE_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВЫБОРА
INQ_CLIPPING	УЗНАТЬ ЗНАЧЕНИЯ ОТСЕЧЕНИЯ
INQ_COLOUR_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЦВЕТА
INQ_COLOR_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА
INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES	УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ ИНДИВИДУАЛЬНЫХ АТРИБУТОВ

Функция ЯГС «Узнать значение текущего индивидуального атрибута» отображается в следующие функции:

INQ_CHAR_EXPANSION_FACTOR
INQ CHAR_SPACING
INQ_FILL_AREA_COLOUR_INDEX
INQ_FILL_AREA_INTERIOR_STYLE
INQ_LINETYPE
INQ LINEWIDTH_SCALE_FACTOR
INQ_LIST_OF_ASF
INQ_POLYLINE_COLOUR_INDEX
INQ POLYMARKER_COLOUR_INDEX
INQ_POLYMARKER_SIZE_SCALE_FACTOR
INQ_POLYMARKER_TYPE
INQ_TEXT_COLOUR_INDEX
INQ_TEXT_FONT_AND_PRECISION

INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER	УЗНАТЬ НОМЕР ТЕКУЩЕГО ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ
INQ_CURRENT_PICK_ID_VALUE	УЗНАТЬ ЗНАЧЕНИЕ ИДЕНТИФИКАТОРА УКАЗАНИЯ
INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES	УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ АТРИБУТОВ ПРИМИТИВОВ

Функция ЯГС «Узнать текущие значения атрибутов примитивов» отображается в следующие функции:

INQ_CHAR_BASE_VECTOR
INQ_CHAR_HEIGHT
INQ_CHAR_WIDTH
INQ_CHAR_UP_VECTOR
INQ_FILL_AREA_INDEX
INQ_PATTERN_HEIGHT_VECTOR
INQ_PATTERN_REFERENCE_POINT
INQ_PATTERN_WIDTH_VECTOR
INQ_POLYLINE_INDEX
INQ_POLYMARKER_INDEX
INQ_TEXT_ALIGNMENT
INQ_TEXT_INDEX
INQ_TEXT_PATH

Продолжение табл. 3

Имя в Аде	Функция ЯГС
INQ_DEFAULT_CHOICE_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВЫБОРА
INQ_DEFAULT_DEFERRAL_STATE_VALUES	УЗНАТЬ РЕЖИМ ЗАДЕРЖКИ ПО УМОЛЧАНИЮ
INQ_DEFAULT_LOCATOR_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ПОЗИЦИИ
INQ_DEFAULT_PICK_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА УКАЗАНИЯ
INQ_DEFAULT_STRING_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА СТРОКИ
INQ_DEFAULT_STROKE_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ

Продолжение табл. 3

Имя в Ада	Функция ЯГС
INQ_DEFAULT_VALUATOR_DEVICE_DATA	УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ЧИСЛА
INQ_DISPLAY_SPACE_SIZE	УЗНАТЬ РАЗМЕР НОСИТЕЛЯ ИЗОБРАЖЕНИЯ
INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES	УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОЙ МОДИФИКАЦИИ АТРИБУТОВ СЕГМЕНТОВ
INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES	УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОГО ОБНОВЛЕНИЯ ХАРАКТЕРИСТИК ИЗОБРАЖЕНИЯ НА СТАНЦИИ
INQ_FILL_AREA_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
INQ_FILL_AREA_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
INQ_GDP	УЗНАТЬ ХАРАКТЕРИСТИКИ ОБОБЩЕННОГО ПРИМИТИВА ВЫВОДА
INQ_INPUT_QUEUE_OVERFLOW	УЗНАТЬ НАЛИЧИЕ ПЕРЕПОЛНЕНИЯ ОЧЕРЕДИ СОБЫТИЙ
INQ_LEVEL_OF_GKS	УЗНАТЬ УРОВЕНЬ ЯГС
INQ_LIST_OF_AVAILABLE_GDP	УЗНАТЬ ИДЕНТИФИКАТОРЫ ДОСТУПНЫХ ОБОБЩЕННЫХ ПРИМИТИВОВ ВЫВОДА
INQ_LIST_OF_AVAILABLE_WS_TYPES	УЗНАТЬ ДОСТУПНЫЕ ТИПЫ СТАНЦИЙ
INQ_LIST_OF_COLOUR_INDICES	УЗНАТЬ ИНДЕКС ЦВЕТА
INQ_LIST_OF_FILL_AREA_INDCES	УЗНАТЬ ИНДЕКСЫ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBERS	УЗНАТЬ СПИСОК ПРЕОБРАЗОВАНИЙ НОРМИРОВАНИЯ
INQ_LIST_OF_PATTERN_INDICES	УЗНАТЬ ИНДЕКСЫ ШАБЛОНА
INQ_LIST_OF_POLYINE_INDICES	УЗНАТЬ ИНДЕКСЫ ЛОМАНОЙ
INQ_LIST_OF_POLYMARKER_INDICES	УЗНАТЬ ИНДЕКСЫ ПОЛИМАРКЕРА
INQ_LIST_OF_TEXT_INDICES	УЗНАТЬ ИНДЕКСЫ ТЕКСТА
INQ_LOCATOR_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОЗИЦИИ
INQ_MAX_LENGTH_OF_WS_STATE_TABLES	УЗНАТЬ ДЛИНУ ТАБЛИЦ, ХАРАКТЕРИЗУЮЩИХ СТАНЦИЮ
INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER	УЗНАТЬ МАКСИМАЛЬНЫЙ НОМЕР ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ
INQ_MORE_SIMULTANEOUS_EVENTS	УЗНАТЬ НАЛИЧИЕ ОДНОВРЕМЕННЫХ СОБЫТИЙ

Продолжение табл. 3

Имя в Ада	Функция ЯГС
INQ_NAME_OF_OPEN_SEGMENT	УЗНАТЬ ИМЯ ОТКРЫТОГО СЕГМЕНТА
INQ_NORMATIZATION_TRANSFORMATION	УЗНАТЬ ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES	УЗНАТЬ ЧИСЛО ДОПУСТИМЫХ УСТРОЙСТВ ВВОДА
INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED	УЗНАТЬ ДОПУСТИМОЕ ЧИСЛО ПРИОРИТЕТОВ СЕГМЕНТОВ
INQ_OPERATING_STATE_VALUE	УЗНАТЬ ФУНКЦИОНАЛЬНОЕ СОСТОЯНИЕ
INQ_PATTERN_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ШАБЛОНА
INQ_PATTERN_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА
INQ_PICK_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА УКАЗАНИЯ
INQ_PIXEL	УЗНАТЬ ЦВЕТ ПИКСЕЛЯ
INQ_PIXEL_ARRAY	УЗНАТЬ МАТРИЦУ ПИКСЕЛЕЙ
INQ_PIXEL_ARRAY_DIMENSION	УЗНАТЬ РАЗМЕРНОСТЬ МАТРИЦЫ ПИКСЕЛЕЙ
INQ_POLYLINE_FACTILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЛОМАНОЙ
INQ_POLYLINE REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ
INQ_POLYMARKER_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ПОЛИМАРКЕРА
INQ_POLYMARKER_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА
INQ_PREDEFINED_COLOUR_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА ПО УМОЛЧАНИЮ
INQ_PREDEFINED_FILL_AREA_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ПО УМОЛЧАНИЮ
INQ_PREDEFINED_PATTERN_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА ПО УМОЛЧАНИЮ
INQ_PREDEFINED_POLYLINE_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ ПО УМОЛЧАНИЮ
INQ_PREDEFINED_POLYMARKER_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА ПО УМОЛЧАНИЮ
INQ_PREDEFINED_TEXT_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА ПО УМОЛЧАНИЮ
INQ_SEGMENT_ATTRIBUTES	УЗНАТЬ АТРИБУТЫ СЕГМЕНТА
INQ_SET_OF_ACTIVE_WS	УЗНАТЬ НАБОР АКТИВНЫХ СТАНЦИЙ
INQ_SET_OF_ASSOCIATED_WS	УЗНАТЬ СТАНЦИИ СВЯЗАННЫЕ С СЕГМЕНТОМ
INQ_SET_OF_OPEN_WS	УЗНАТЬ НАБОР ОТКРЫТЫХ СТАНЦИЙ

Продолжение табл. 3

Имя в Ада	Функция ЯГС
INQ_SET_OF_SEGMENT_NAMES_IN_USE	УЗНАТЬ ИМЕНА СУЩЕСТВУЮЩИХ СЕГМЕНТОВ
INQ_SET_SEGMENT_NAMES_ON_WS	УЗНАТЬ ИМЕНА СЕГМЕНТОВ, ХРАНИМЫХ НА СТАНЦИИ
INQ_STRING_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА СТРОКИ
INQ STROKE_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ
INQ_TEXT_EXTENT	УЗНАТЬ ГАБАРИТЫ ТЕКСТА
INQ TEXT_FACILITIES	УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ТЕКСТА
INQ_TEXT_REPRESENTATION	УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА
INQ_VALUATOR_DEVICE_STATE	УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ЧИСЛА
INQ_WS_CATEGORY	УЗНАТЬ КАТЕГОРИЮ СТАНЦИИ
INQ_WS_CLASSIFICATION	УЗНАТЬ КЛАСС СТАНЦИИ
INQ_WS_CONNECTION_AND_TYPE	УЗНАТЬ ТИП И ИДЕНТИФИКАТОР СВЯЗИ СТАНЦИИ
INQ_WS_DEFERRAL_AND_UPDATE_STATES	УЗНАТЬ РЕЖИМЫ ЗАДЕРЖКИ И ОБНОВЛЕНИЯ СТАНЦИИ
INQ_WS_MAX_NUMBERS	УЗНАТЬ ДОПУСТИМОЕ ЧИСЛО СТАНЦИЙ
INQ_WS_STATE	УЗНАТЬ СОСТОЯНИЕ СТАНЦИИ
INQ_WS_TRANSFORMATION	УЗНАТЬ ПРЕОБРАЗОВАНИЕ СТАНЦИИ
INSERT_SEGMENT	ВСТАВИТЬ СЕГМЕНТ
INTERPRET_ITEM	ИНТЕРПРЕТИРОВАТЬ ЗАПИСЬ
MESSAGE	СООБЩЕНИЕ
OPEN_GKS	ОТКРЫТЬ ЯГС
OPEN_WS	ОТКРЫТЬ СТАНЦИЮ
POLYLINE	ЛОМАНАЯ
POLYMARKER	ПОЛИМАРКЕР
READ_ITEM_FROM_GKSM	ПРОЧИТАТЬ ЗАПИСЬ ИЗ ЯГС
REDRAW_ALL_SEGMENTS_ON_WS	ПЕРЕРИСОВАТЬ ВСЕ СЕГМЕНТЫ НА СТАНЦИИ
RENAME_SEGMENT	ПЕРЕИМЕНОВАТЬ СЕГМЕНТ
REQUEST_CHOICE	ЗАПРОСИТЬ УСТРОЙСТВО ВЫБОРА
REQUEST_LOCATOR	ЗАПРОСИТЬ ВВОД ПОЗИЦИИ
REQUEST_PICK	ЗАПРОСИТЬ УКАЗАНИЯ
REQUEST_STRING	ЗАПРОСИТЬ ВВОД СТРОКИ
REQUEST_STROKE	ЗАПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ
REQUEST_VALUATOR	ЗАПРОСИТЬ ВВОД ЧИСЛА
SAMPLE_CHOICE	ОПРОСИТЬ ВЫБОР
SAMPLE_LOCATOR	ОПРОСИТЬ ВВОД ПОЗИЦИИ
SAMPLE_PICK	ОПРОСИТЬ УКАЗАНИЕ
SAMPLE_STRING	ОПРОСИТЬ ВВОД СТРОКИ

Продолжение табл. 3

Имя в Ада	Функция ЯГС
SAMPLE_STROKE	ОПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ
SAMPLE_VALUATOR	ОПРОСИТЬ ВВОД ЧИСЛА
SELECT_NORMALIZATION_TRANSFORMATION	ВЫБРАТЬ ПРЕОБРАЗОВАНИЕ НОРМИРОВАНИЯ
SET_ASF	ЗАДАТЬ ФЛАГИ ВЫБОРКИ АТРИБУТОВ
SET_CHAR_EXPANSION_FACTOR	ЗАДАТЬ МАСШТАБ РАСШИРЕНИЯ ЛИТЕРЫ
SET_CHAR_HEIGHT	ЗАДАТЬ ВЫСОТУ ЛИТЕР
SET_CHAR_SPACING	ЗАДАТЬ МЕЖЛИТЕРНЫЙ ПРОСВЕТ
SET_CHAR_UP_VECTOR	ЗАДАТЬ ВЕРТИКАЛЬ ЛИТЕРЫ
SET_CHOICE_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВЫБОРА
SET_CLIPPING_INDICATOR	ЗАДАТЬ ИНДИКАТОР ОТСЕЧЕНИЯ
SET_COLOUR_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА
SET_DEFERRAL_STATE	ЗАДАТЬ РЕЖИМ ЗАДЕРЖКИ
SET_DETECTABILITY	ЗАДАТЬ ЧУВСТВИТЕЛЬНОСТЬ К УКАЗАНИЮ
SET_FILL_AREA_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_INDEX	ЗАДАТЬ ИНДЕКС ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_INTERIOR_STYLE	ЗАДАТЬ ВИД ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_FILL_AREA_STYLE_INDEX	ЗАДАТЬ ИНДЕКС ЗАПОЛНИТЕЛЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
SET_HIGHLIGHTING	ЗАДАТЬ ВЫДЕЛЕНИЕ
SET_LINETYPE	ЗАДАТЬ ТИП ЛИНИИ
SET_LINEWIDTH_SCALE_FACTOR	ЗАДАТЬ МАСШТАБ ТОЛЩИНЫ ЛИНИИ
SET_LOCATOR_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ПОЗИЦИИ
SET_MARKER_SIZE_SCALE_FACTOR	ЗАДАТЬ МАСШТАБ МАРКЕРА
SET_MARKER_TYPE	ЗАДАТЬ ТИП МАРКЕРА
SET_PATTERN_REFERENCE_POINT	ЗАДАТЬ ТОЧКУ ПРИВЯЗКИ ШАБЛОНА
SET_PATTERN_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА
SET_PATTERN_SIZE	ЗАДАТЬ РАЗМЕР ШАБЛОНА
SET_PICK_ID	ЗАДАТЬ ИДЕНТИФИКАТОР УКАЗАНИЯ
SET_PICK_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА УКАЗАНИЯ
SET_POLYLINE_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ЛОМАНОЙ

Продолжение табл. 3

Имя в Аде	Функция ЯГС
SET_POLYLINE_INDEX	ЗАДАТЬ ИНДЕКС ЛОМАНОЙ
SET_POLYLINE_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ
SET_POLYMARKER_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛИМАРКЕРА
SET_POLYMARKER_INDEX	ЗАДАТЬ ИНДЕКС ПОЛИМАРКЕРА
SET_POLYMARKER_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА
SET_SEGMENT_PRIORITY	ЗАДАТЬ ПРИОРИТЕТ СЕГМЕНТА
SET_SEGMENT_TRANSFORMATION	ЗАДАТЬ ПРЕОБРАЗОВАНИЕ СЕГМЕНТА
SET_STRING_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА СТРОКИ
SET_STROKE_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ
SET_TEXT_ALIGNMENT	ЗАДАТЬ ВЫРАВНИВАНИЕ ТЕКСТА
SET_TEXT_COLOUR_INDEX	ЗАДАТЬ ИНДЕКС ЦВЕТА ТЕКСТА
SET_TEXT_FONT_AND_PRECISION	ЗАДАТЬ ШРИФТ И ТОЧНОСТЬ ТЕКСТА
SET_TEXT_INDEX	ЗАДАТЬ ИНДЕКС ТЕКСТА
SET_TEXT_PATH	ЗАДАТЬ НАПРАВЛЕНИЕ ТЕКСТА
SET_TEXT_REPRESENTATION	ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА
SET_VALUATOR_MODE	ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ЧИСЛА
SET_VIEWPORT	ЗАДАТЬ ПОЛЕ ВЫВОДА
SET_VIEWPORT_INPUT_PRIORITY	ЗАДАТЬ ПРИОРИТЕТ ПОЛЯ ВЫВОДА ПРИ ВВОДЕ
SET_VISIBILITY	ЗАДАТЬ ВИДИМОСТЬ
SET_WINDOW	ЗАДАТЬ ОКНО
SET_WS_VIEWPORT	ЗАДАТЬ ПОЛЕ ВЫВОДА СТАНЦИИ
SET_WS_WINDOW	ЗАДАТЬ ОКНО СТАНЦИИ
TEXT	ТЕКСТ
UPDATE_WS	ОБНОВИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ
WRITE_ITEM_TO_GKSM	ЗАПИСЬ В МЕТАФАЙЛ

Функции ЯГС, упорядоченные по алфавиту

Функции находятся в том же порядке, в котором перечислены имена процедур, соответствующие именам функций ЯГС. В табл. 3 перечислены по алфавиту имена функций ЯГС.

Таблица 4

Список функций ЯГС по уровням и по алфавиту

Уровень 0а

ACTIVATE_WS
CELL_ARREY
CLEAR_WS
CLOSE_GKS
CLOSE_WS
DEACTIVATE_WS
EMERGENCY_CLOSE_GKS
ERROR_HANDLING
ERROR_LOGGING
ESCAPE
FILL_AREA
GDP
GET_ITEM_TYPE_FROM_GKSM
INQ_CLIPPING
INQ_COLOUR_FACILITIES
INQ_COLOUR_REPRESENTATION
INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES

Функция ЯГС «Узнать значение текущего индивидуального атрибута» отображается в следующие функции:

INQ_CHAR_EXPANSION_FACTOR
INQ_CHAR_SPACING
INQ_FILL_AREA_COLOUR_INDEX
INQ_FILL_AREA_INTERIOR_STYLE
INQ_FILL_AREA_STYLE_INDEX
INQ_LINETYPE
INQ_LINEWIDTH_SCALE_FACTOR
INQ_LIST_OF_ASF
INQ_POLYLINE_COLOUR_INDEX
INQ_POLYMARKER_COLOUR_INDEX
INQ_POLYMARKER_SIZE_SCALE_FACTOR
INQ_POLYMARKER_TYPE
INQ_TEXT_COLOUR_INDEX
INQ_TEXT_FONT_AND_PRECISION
INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER
INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES

Функция ЯГС «Узнать значение атрибута текущего примитива вывода» отображается в следующие функции:

INQ_POLYLINE_INDEX
INQ_POLYMARKER_INDEX
INQ_TEXT_INDEX
INQ_CHAR_HEIGHT
INQ_CHAR_UP_VECTOR
INQ_CHAR_WIDTH

INQ_CHAR_BASE_VECTOR
INQ_TEXT_PATH
INQ_TEXT_ALIGNMENT
INQ_FILL_AREA_INDEX
INQ_PATTERN_HEIGHT_VECTOR
INQ_PATTERN_WIDTH_VECTOR
INQ_PATTERN_REFERENCE_POINT
INQ_DISPLAY_SPACE_SAZE
INQ_FILL_AREA_FACILITIES
INQ_GDP
INQ_LEVEL_OF_GKS
INQ_LIST_OF_AVAILABLE_GDP
INQ_LIST_OF_AVAILABLE_WS_TYPES
INQ_LIST_OF_COLOUR_INDICES
INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBER
INQ_MAX_LENGTH_WS_STATE_TABLES
INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER
INQ_NORMALIZATION_TRANSFORMATION
INQ_OPERATING_STATE_VALUE
INQ_PATTERN_FACILITES
INQ_PIXEL
INQ_PIXEL_ARREY
INQ_PIXEL_ARREY_DIMENSIONS
INQ_POLYLINE_FACILITIES
INQ_POLYMARKER_FACILITIES
INQ_PREDEFINED_COLOUR_REPRESENTATION
INQ_PREDEFINED_FILL_AREA_REPRESENTATION
INQ_PREDEFINED_PATTERN_REPRESENTATION
INQ_PREDEFINED_POLYLINE_REPRESENTATION
INQ_PREDEFINED_POLYMARKER_REPRESENTATION
INQ_PREDEFINED_TEXT_REPRESENTATION
INQ_SET_OF_OPEN_WS
INQ_TEXT_EXTENT
INQ_TEXT_FACILITIES
INQ_WS_CATEGORIES
INQ_WS_CLASSIFICATION
INQ_WS_CONNECTION_AND_TYPE
INQ_WS_DEFERRAL_AND_UPDATE_STATES
INQ_WS_STATE
INQ_WS_TRANSFORMATION
INTERPRET_ITEM
OPEN_GKS
OPEN_WS
POLYLINE
POLYMARKER
READ_ITEM_FROM_GKSM
SELECT_NORMALIZATION_TRANSFORMATION
SET_ASF
SET_CHAR_EXPANSION_FACTOR
SET_CHAR_HEIGHT

Продолжение табл. 4

	SET_CHAR_SPACING SET_CHAR_UP_VECTOR SET_CLIPPING_INDICATOR SET_COLOUR_REPRESENTATION SET_FILL_AREA_COLOUR_INDEX SET_FILL_AREA_INDEX SET_FILL_AREA_INTERIOR_STYLE SET_FILL_AREA_STYLE_INDEX SET_LINETYPE SET_LINEWIDTH_SCALE_FACTOR SET_MARKER_SIZE_SCALE_FACTOR SET_MARKER_TYPE SET_PATTERN_REFERENCE_POINT SET_PATTERN_SIZE SET_POLYLINE_COLOUR_INDEX SET_POLYMARKER_COLOUR_INDEX SET_POLYLINE_INDEX SET_POLYMARKER_INDEX SET_TEXT_ALIGNMENT SET_TEXT_COLOUR_INDEX SET_TEXT_FONT_AND_PRECISION SET_TEXT_INDEX SET_TEXT_PATH SET_VIEWPORT SET_WINDOW SET_WS_VIEWPORT SET_WS_WINDOW TEXT UPDATE_WS WRITE_ITEM_TO_GKSM
Уровень 0b	
	INITIALISE_CHOICE INITIALISE_LOCATOR INITIALISE_STRING INITIALISE_STROKE INITIALISE_VALUATOR INQ_CHOICE_DEVICE_STATE INQ_DEFAULT_CHOICE_DEVICE_STATE INQ_DEFAULT_LOCATOR_DEVICE_DATA INQ_DEFAULT_STRING_DEVICE_DATA INQ_DEFAULT_STROKE_DEVICE_DATA INQ_DEFAULT_VALUATOR_DEVICE_DATA INQ_LOCATOR_DEVICE_STATE INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES INQ_STRING_DEVICE_STATE INQ_STROKE_DEVICE_STATE INQ_VALUATOR_DEVICE_STATE REQUEST_CHOICE REQUEST_LOCATOR

Продолжение табл. 4

	REQUEST_STRING REQUEST_STROKE REQUEST_VALUATOR SET_CHOICE_MODE SET_LOCATOR_MODE SET_STRING_MODE SET_STROKE_MODE SET_VALUATOR_MODE SET_VIEWPORT_INPUT_PRIORITY
Уровень 0с	
	AWAIT_EVENT FLUSH_DEVICE_EVENTS GET_CHOICE GET_LOCATOR GET_STRING GET_STROKE GET_VALUATOR INQ_INPUT_QUEUE_OVERFLOW INQ_MORE_SIMULTANEOUS_EVENTS SAMPLE_CHOICE SAMPLE_LOCATOR SAMPLE_STRING SAMPLE_STROKE SAMPLE_VALUATOR
Уровень 1а	
	ACCUMULATE_TRANSFORMATION_MATRIX CLOSE_SEGMENT CREATE_SEGMENT DELETE_SEGMENT DELETE_SEGMENT_FROM_WS EVALUATE_TRANSFORMATION_MATRIX INQ_DEFAULT_DEFERRAL_STATE_VALUES INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES INQ_FILL_AREA_REPRESENTATION INQ_LIST_OF_FILL_AREA_INDICES INQ_LIST_OF_PATTERN_INDICES INQ_LIST_OF_POLYLINE_INDICES INQ_LIST_OF_POLYMARKER_INDICES INQ_LIST_OF_TEXT_INDICES INQ_NAME_OF_OPEN_SEGMENT INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED INQ_PATTERN_REPRESENTATION INQ_POLYLINE_REPRESENTATION INQ_POLYMARKER_REPRESENTATION

Продолжение табл. 4

	INQ_SEGMENT_ATTRIBUTES INQ_SET_OF_ACTIVE_WS INQ_SET_OF_ASSOCIATED_WS INQ_SET_OF_SEGMENT_NAMES_IN_USE INQ_SET_OF_SEGMENT_NAMES_ON_WS INQ_TEXT_REPRESENTATION INQ_WS_MAX_NUMBERS MESSAGE REDRAW_ALL_SEGMENTS_ON_WS RENAME_SEGMENT SET_DEFERRAL_STATE SET_FILL_AREA_REPRESENTATION SET_HIGHLIGHTING SET_PATTERN_REPRESENTATION SET_POLYLINE_REPRESENTATION SET_POLYMARKER_REPRESENTATION SET_SEGMENT_PRIORITY SET_SEGMENT_TRANSFORMATION SET_TEXT_REPRESENTATION SET_VISIBILITY
Уровень 1b	
	INITIALISE_PICK INQ_CURRENT_PICK_ID_VALUE INQ_DEFAULT_PICK_DEVICE_DATA INQ_PICK_DEVICE_STATE REQUEST_PICK SET_DETECTABLILITY SET_PICK_ID SET_PICK_MODE
Уровень 1c	
	GET_PICK SAMPLE_PICK
Уровень 2c	
	ASSOCIATE_SEGMENT_WITH_WS COPY_SEGMENT_TO_WS INSERT_SEGMENT
Уровень 2b	
	Отсутствуют
Уровень 2c	
	Отсутствуют

4.2. **Определение типов данных**

4.2.1. Сокращения, используемые в определениях типов данных

ASF	Флаг выборки атрибутов
CHAR	Литера
DC	Координата устройства
GDP	Обобщенный примитив вывода
GKS	Ядро графической системы
GKSM	Метафайл ядра графической системы
ID	Идентификатор
MAX	Максимум
NDC	Нормализованные координаты устройства
WC	Мировые координаты
WS	Станция

4.2.2. Определение типов в алфавитном порядке

В данном разделе в алфавитном порядке даются определения типов данных, используемые в связывании Ады с ЯГС. Каждая такая декларация задает уровень, на котором должна иметься декларация данных в реализации ЯГС данного уровня и любого более высокого уровня, в котором декларация типа впервые понадобится (аналогично функциям). Каждый декларируемый элемент также включает комментарий о типе и/или/использовании типа. Некоторые декларации в определении типа содержат константы. Все эти декларации констант включены в пакет GKS_TYPE.

ASF Уровень 0а

```
type ASF is (BUNDLED, INDIVIDUAL);
```

Данный тип определяет флаг выборки атрибутов, чье значение указывает, должен ли атрибут примитива загружаться из таблицы связок или из индивидуального атрибута.

ASF_LIST Уровень 0а

```
type ASF_LIST is
   record
      TYPE_OF_LINE_ASF        : ASF;
      WIDTH_ASF               : ASF;
      LINE_COLOUR_ASF         : ASF;
      TYPE_OF_MARKER_ASF      : ASF;
      SIZE_ASF                : ASF;
```

```
    MARKER_COLOUR_ASF      : ASF;
    FONT_PRECISION_ASF     : ASF;
    EXPANSION_ASF          : ASF;
    SPACING_ASF            : ASF;
    TEXT_COLOUR_ASF        : ASF;
    INTERIOR_ASF           : ASF;
    STYLE_ASF              : ASF;
    FILL_AREA_COLOUR_      : ASF;
    _ASF
    end record;
```

Запись содержит все исходные флаги выборки атрибутов с компонентами, указанными индивидуальными флагами.

ATTRIBUTES_USED — Уровень 0a

```
package ATTRIBUTES_USED is
  new GKS_LIST_UTILITIES (ATTRIBUTES_USED_TYPE);
```

Предоставляет список используемых атрибутов.

ATTRIBUTES_USED_TYPE — Уровень 0a

```
type ATTRIBUTES_USED_TYPE is (POLYLINE_ATTRIBUTES,
                              POLYMARKER_ATTRIBUTES,
                              TEXT_ATTRIBUTES,
                              FILL_AREA_ATTRIBUTES);
```

Типы атрибутов, которые могут быть использованы в генерируемом выводе для GDP и в генерируемой информации подсказки и эха различных классов входных устройств.

CHAR_EXPANSION — Уровень 0a

```
type CHAR_EXPANSION is new SCALE_FACTOR range
  SCALE_FACTOR'SAFE_SMALL .. SCALE_FACTOR'LAST;
```

Определяет масштаб расширения литер. Масштаб должен быть больше нуля.

CHAR_SPACING — Уровень 0a

```
type CHAR_SPACING is new SCALE_FACTOR;
```

Определяет межлитерный просвет. Положительное значение межлитерного просвета в строке текста, а отрицательное значение обозначает перекрытие между прямоугольниками литер в строке текста.

CHOICE_DEVICE_NUMBER — Уровень 0b

```
type CHOICE_DEVICE_NUMBER is new DEVICE_NUMBER;
```

Обеспечивает идентификаторы устройств выбора.

CHOICE_PROMPT Уровень 0b

type CHOICE_PROMPT is (OFF, ON);

Указывает на выбор типа подсказки и эха и будет или нет отображаться заданная подсказка.

CHOICE_PROMPTS Уровень 0b

package CHOICE_PROMPT is
new GKS_LIST_UTILITIES (CHOICE_PROMPT);

Предоставляет списки подсказок.

CHOICE_PROMPT_ECHO_TYPE Уровень 0b

type CHOICE_PROMPT_ECHO_TYPE is new INTEGER;

Определяет выбранный тип подсказки и эха.

CHOICE_PROMPT_ECHO_TYPES Уровень 0b

package CHOICE_PROMPT_ECHO_TYPES is
new GKS_LIST_UTILITIES (CHOICE_PROMPT_ECHO_TYPE);

Предоставляет списки выбранной подсказки и эха.

CHOICE_PROMPT_STRING Уровень 0b

type CHOICE_PROMPT_STRING (LENGTH:STRING_SMALL_NATURAL := 0) is
record
CONTENTS: STRING (1 .. LENGTH);
end record;

Предоставляется для подсказок переменной длины. Типы должны быть декларированы таким образом, чтобы была возможна динамическая модификация длины.

CHOICE_PROMPT_STRING_ARRAY Уровень 0b

type CHOICE_PROMPT_STRING_ARRAY is array (POSITIVE range<>)
of CHOICE_PROMPT_STRING;

Обеспечивает матрицу строк подсказок.

CHOICE_PROMPT_STRING_LIST Уровень 0b

type CHOICE_PROMPT_STRING_LIST (LENGTH:CHOICE.SMALL. NATURAL :=0)
is record
LIST:CHOICE_PROMPT_STRING_ARRAY (1 .. LENGTH);
is record

Обеспечивает списки строк подсказок.

CHOICE_REQUEST_STATUS Уровень 0b

type CHOICE_REQUEST_STATUS is (OK, NOCHOICE, NONE);

Определяет статус выбора входной операции для функции запроса.

CHOICE_SMALL_NATURAL Уровень 0b

subtype CHOICE_SMALL_NATURAL is NATURAL RANGE 0 .. CHOICE_SMALL_NATURAL_MAX;

Это декларация подтипа, которая допускает вызов объектов записей для типа CHOICE_PROMPT_STRING_LIST без возникновения прерывания STORAGE_ERROR.

CHOICE_STATUS Уровень 0b

sybtupe CHOICE_STATUS is CHOICE_REQUEST_STATUS range OK .. NOCHOICE;

Указывает, был ли сделан оператором выбор для функций «опросить», «получить» и «узнать».

CHOICE_VALUE Уровень 0b

type CHOICE_VALUE is new POSITIVE;

Определяет выбор значений, существующих для реализации.

CLIPPING_INDICATOR Уровень 0a

type CLIPPING_INDICATOR is (CLIP, NOCLIP);

Указывает, будет или нет выполняться отсечение.

COLOUR_AVAILABLE Уровень 0a

type COLOUR_AVAILABLE is (COLOUR, MONOCHROME);

Указывает, существует ли вывод цвета на станции.

COLOUR_INDEX Уровень 0a

subtupe COLOUR_INDEX is PIXEL_COLOUR_INDEX range 0 .. PIXEL_COLOUR_INDEX'LAST;

Указывает на тип в таблицах цвета.

COLOUR_INDICES Уровень 0a

package COLOUR_INDICES is new GKS_LIST_UTILITIES (COLOUR_INDEX);

Предназначены для установки индикаторов цвета, которые имеются на конкретной станции.

COLOUR_MATRIX Уровень 0a

type COLOUR_MATRIX is array (POSITIVE range <>, POSITIVE range <>) of COLOUR_INDEX;

Обеспечивает матрицы, содержащие индикаторы цвета, соответствующие матрице ячеек или матрице шаблонов.

COLOUR_REPRESENTATION Уровень 0a

```
type COLOUR_REPRESENTATION is
 record
     RED        : INTENSITY;
     GREEN      : INTENSITY;
     BLUE       : INTENSITY;
 end record;
```

Определяет представление цвета как комбинацию интенсивностей в системе цветов красный—зеленый—голубой.

CONTROL_FLAG Уровень 0a

type CONTROL_FLAG is (CONDITIONALLY, ALWAYS);

Флаги управления используются для указания условий, при которых носитель изображения должен быть очищен.

DC Уровень 0a

package DC is new GKS_COORDINATE_SYSTEM (DC_TYPE);

Определяет систему координат устройства.

DC_TYPE Уровень 0a

type DC_TYPE is digits PRECISION;

Тип координат в системе координат устройства.

DC_UNITS Уровень 0a

type DC_UNITS is (METRES, OTHER);

Единицы координат устройства для конкретной станции должны задаваться в метрах, если устройство не способно порождать точно масштабированные образы, или в соответствующих зависящих от станции единицах в противном случае.

DEFERRAL_MODE Уровень 0a

type DEFERRAL_MODE is (ASAP, BNIG, BNIL, ASTI);

Определяет четыре задержанных режима ЯГС.

DEVICE_NUMBER Уровень 0b

```
package DEVICE_NUMBER_TYPE is
        DEVICE_NUMBER_TYPE is new POSITIVE;
```

```
end DEVICE_NUMBER_TYPE;
```

К логическим устройствам входа обращаются по номерам устройств.

DISPLAY_CLASS Уровень 0a

```
type DISPLAY_CLASS is (VECTOR_DISPLAY,
                       RASTER_DISPLAY,
                       OTHER_DISPLAY);
```

Классификация станций по категориям OUTPUT (вывод) или OUTIN (ввод/вывод).

DISPLAY_SURFACE_EMPTY Уровень 0a

```
type DISPLAY_SURFACE_EMPTY is (EMPTY, NOTEMPTY);
```

Указывает, очищен ли носитель изображения.

DYNAMIC_MODIFICATION Уровень 1a

```
type DYNAMIC_MODIFICATION is (IRG, IMM);
```

Указывает, обновление списка состояний выполняется немедленно или требуется повторная генерация.

ECHO_SWITCH Уровень 0b

```
type ECHO_SWITCH is (ECHO, NOECHO);
```

Указывает на то, выполняется или нет эхо-вывод подсказки.

ERROR_NUMBER Уровень 0a

```
type ERROR_NUMBER is new INTEGER;
```

Определяет тип для значений индикаторов ошибок.

EVENT_DEVICE_NUMBER Уровень 0c

```
type EVENT_DEVICE_
        NUMBER (CLASS  : INPUT_CLASS :=NONE) is
record
  case CLASS is
  when NONE                => null,
  when LOCATOR_INPUT       => LOCATOR_EVENT_
                                 _DEVICE
  when STROKE_INPUT        => STROKE_EVENT_DEVICE
                              : STROKE_DEVICE_
                                 _NUMBER;
  when VALUATOR_           => VALUATOR_EVENT_
    _INPUT                       _DEVICE
                              : VALUATOR_DEVICE_
                                 _NUMBER;
```

```
    when CHOICE_INPUT      => CHOICE_EVENT_DEVICE
                              : CHOICE_DEVICE_
                                        _NUMBER;
    when PICK_INPUT        => PICK_EVENT_DEVICE
                              : PICK_DEVICE_
                                        _NUMBER;
    when STRING_INPUT      => STRING_EVENT_DEVICE
                              : STRING_DEVICE_
                                        _NUMBER;
  end case;
end record;
```

Обеспечивает возврат любого класса номера устройства из очереди событий.

EVENT_OVERFLOW_DEVICE_NUMBER Уровень 0с

```
type EVENT_OVERFLOW_DEVICE_NUMBER
  (CLASS : INPUT_QUEUE_CLASS := LOCATOR_INPUT) is
record
case CLASS is
  when LOCATOR_INPUT   => LOCATOR_EVENT_
                                  _DEVICE
                          : LOCATOR_DEVICE_
                                  _NUMBER;
  when STROKE_INPUT    => STROKE_EVENT_DEVICE
                              : STROKE_DEVICE_
                                  _NUMBER;
  when VALUATOR_       => VALUATOR_EVENT_
    _INPUT                        _DEVICE
                          : VALUATOR_DEVICE_
                                  _NUMBER;
  when CHOICE_INPUT    => CHOICE_EVENT_
                                  _DEVICE
                          : CHOICE_DEVICE_
                                  _NUMBER;
  when PICK_INPUT      => PICK_EVENT_DEVICE
                          : PICK_DEVICE_
                                  _NUMBER;
  when STRING_INPUT    => STRING_EVENT_DEVICE
                          : STRING_DEVICE_
                                  _NUMBER;
  end case;
end record;
```

Дается номер класса устройства для возврата для очереди событий.

FILL_AREA_INDEX Уровень 0a

type FILL_AREA_INDEX is new POSITIVE;

Определяются индексы таблицы связок полигональной области.

FILL_AREA_INDICES Уровень 0a

package FILL_AREA_INDICES is
new GKS_LIST_UTILITIES (FILL_AREA_INDEX);

Обеспечивает список индексов таблицы связок полигональных областей.

GDP_ID Уровень 0a

type GDP_ID is new INTEGER;

Осуществляет выбор среди различных обобщенных примитивов вывода.

GDP_IDS Уровень 0a

package GDP_IDS is new GKS_LIST_UTILITIES (GDP_ID);

Предоставляет список идентификаторов обобщенных примитивов вывода.

GKS_Level Уровень 0a

type GKS_Level is (L0a, L0b, L0c, L1a, L1b, L1c, L2a, L2b, L2c);

Имеющиеся уровни ЯГС.

GKSM_ITEM_TYPE Уровень 0a

type GKSM_ITEM_TYPE is new NATURAL;

Тип элементов, содержащихся в метафайле GKSM.

HATCH_STYLE Уровень 0a

subtupe HATCH_STYLE is STYLE_INDEX;

Определяет тип варианта штриховки полигональной области.

HATCH_STYLES Уровень 0a

package HATCH_STYLES is new GKS_LIST_UTILITIES
(HATCH_STYLE);

Предоставляет список видов штриховки.

HORIZONTAL_ALIGNMENT Уровень 0a

type HORIZONTAL_ALIGNMENT is (NORMAL, LEFT,
CENTRE, RIGHT);

Выравнивание параллелограмма текста по отношению к горизонтальной позиции текста.

IMPLEMENTATION_DEFINED_ERROR Уровень 0a

```
subtype IMPLEMENTATION_DEFINED_ERROR is
                                        ERROR_NUMBER
    range ERROR_NUMBER'FIRST .. —1;
```

Определяет область значений номеров ошибок, чтобы указать, какие ошибки, заданные в реализации, могут произойти.

INDIVIDUAL_ATTRIBUTE_VALUES Уровень 0a

```
type INDIVIDUAL_ATTRIBUTE_VALUES is
  record
    TYPE_OF_LINE          : LINETYPE;
    WIDTH                 : LINEWIDTH;
    LINE_COLOUR           : COLOUR_INDEX;
    TYPE_OF_MARKER        : MARKER_TYPE;
    SIZE                  : MARKER_SIZE;
    MARKER_COLOUR         : COLOUR_INDEX;
    FONT_PRECISION        : TEXT_FONT_PRECISION;
    EXPANSION             : CHAR_EXPANSION;
    SPACING               : CHAR_SPACING;
    TEXT_COLOUR           : COLOUR_INDEX;
    INTERIOR              : INTERIOR_STYLE;
    STYLE                 : STYLE_INDEX;
    FILL_AREA_COLOUR      : COLOUR_INDEX;
    ASF                   : ASF_LIST;
  end record;
```

Запись, содержащая все текущие индивидуальные атрибуты для процедуры INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES.

INPUT_CLASS Уровень 0b

```
type INPUT_CLASS is (NONE,
                     LOCATOR_INPUT,
                     STROKE_INPUT,
                     VALUATOR_INPUT,
                     CHOICE_INPUT,
                     PICK_INPUT,
                     STRING_INPUT);
```

Определяет классификации входных устройств для рабочих станций категорий INPUT и OUTIN.

INPUT_QUEUE_CLASS Уровень 0c

```
subtype INPUT_QUEUE_CLASS is INPUT_CLASS range
        LOCATOR_INPUT .. STRING_INPUT;
```

Определяет классификации для входных устройств для ситуаций, в которых отсутствие классификации невозможно.

INPUT_STATUS Уровень 0b

type INPUT_STATUS is (OK, NONE);

Определяет состояние устройства ввода позиции, ввода последовательности позиций, ввода числа и ввода строки.

INPUT_STRING Уровень 0b

type INPUT_STRING (LENGTH : STRING_SMALL_NATURAL :=0) is
record
CONTENTS : STRING (1 .. LENGTH),
end record;

Предоставляет строку переменной длины. Объекты данного типа следует декларировать, чтобы позволить динамическую модификацию длины.

INTENSITY Уровень 0a

type INTENSITY is digits PRECISION range 0.0 .. 1.0;

Определяет область возможных значений интенсивностей цвета.

INTERIOR_STYLE Уровень 0a

type INTERIOR_STYLE is (HOLLOW, SOLID, PATTERN, HATCH);

Определяет вид заполнения полигональных областей.

INTERIOR_STYLES Уровень 0a

package INTERIOR_STYLES is
new GKS_LIST_UTILITIES (INTERIOR_STYLE);

Предоставляет список видов заполнения.

INVALID_VALUES_INDICATOR Уровень 0a

type INVALID_VALUES_INDICATOR is (ABSENT, PRESENT);

Указывает, присутствует или отсутствует значение —1 в параметре PIXEL_ARRAY, возвращаемом INQ_PIXEL_ARRAY.

LANGUAGE_BINDING_ERROR Уровень 0a

subtype LANGUAGE_BINDING_ERROR is ERROR_NUMBER range 2500 .. 2999;

Обозначает диапазон номеров ошибок для обозначения ошибок связки с языком, которые могут произойти.

LINETYPE Уровень 0a

type LINETYPE is new INTEGER;

Определяет типы линий, представляемых ЯГС.

LINETYPES Уровень 0a

package LINETYPES is new GKS_LIST_UTILITIES (LINETYPE);

Предоставляет список типов линий.

LINEWIDTH Уровень 0a

type LINEWIDTH is new SCALE_FACTOR range 0.0 .. SCALE_FACTOR'LAST;

Толщина линии определяется коэффициентом масштабирования.

LOCATOR_DEVICE_NUMBER Уровень 0b

type LOCATOR_DEVICE_NUMBER is new DEVICE_NUMBER;

Определяет идентификаторы индикаторов местоположений устройства.

LOCATOR_PROMPT_ECHO_TYPE Уровень 0b

type LOCATOR_PROMPT_ECHO_TYPE is new INTEGER;

Определяет подсказку определителя местоположения и типы эха, поддерживаемые реализацией.

LOCATOR_PROMPT_ECHO_TYPES Уровень 0b

package LOCATOR_PROMPT_ECHO_TYPES is new GKS_LIST_UTILITIES (LOCATOR_PROMPT_ECHO_TYPE);

Предоставляет списки подсказок индикаторов местоположения и типов эха.

MARKER_SIZE Уровень 0a

type MARKER_SIZE is new SCALE_FACTOR range 0.0 .. SCALE_FACTOR'LAST;

Размер маркера указывается масштабом.

MARKER_TYPE Уровень 0a

type MARKER_TYPE is new INTEGER;

Определяет типы маркеров, представляемых ЯГС.

MARKER_TYPES Уровень 0a

package MARKER_TYPES is new GKS_LIST_UTILITIES

(MARKER_TYPE);
Предоставляет список типов маркеров.

MORE_EVENTS Уровень 0c
type MORE_EVENTS is (NOMORE, MORE);
Указывает, содержатся ли события в очереди входных событий.

NDC Уровень 0a
package NDC is new GKS_COORDINATE_SYSTEM
(NDC_TYPE);
Определяет нормализованную систему координат устройства.

NDC_TYPE Уровень 0a
type NDC_TYPE is digits PRECISION;
Определяет типы координат в нормализованной системе координат устройства.

NEW_FRAME_NECESSARY Уровень 0a
type NEW_FRAME_NECESSARY is (NO, YES);
Указывает, необходимы ли новые действия над кадром при модификации.

OPERATING_MODE Уровень 0b
type OPERATING_MODE is (REQUEST_MODE,
SAMPLE_MODE, EVENT_MODE);
Определяет режимы работы входного устройства.

OPERATING_STATE Уровень 0a
type OPERATING_STATE is (GKCL, CKOP, WSOP, WSAC,
SGOP);
Определяет пять рабочих состояний ЯГС.

PATTERN_INDEX Уровень 0a
subtype PATTERN_INDEX is STYLE_INDEX range 1 . .
STYLE_INDEX'LAST;
Определяет область возможных значений индекса таблицы шаблонов.

PATTERN_INDICES Уровень 0a
package PATTERN_INDICES is
new GKS_LIST_UTILITIES (PATTERN_INDEX);
Обеспечивает списки индексов шаблонов.

PICK_DEVICE_NUMBER Уровень 1b
type PICK_DEVICE_NUMBER is new DEVICE_NUMBER;
Предоставляется для выбранных устройств.

PICK_ID Уровень 1b
type PICK_ID is new POSITIVE;
Определяет область значений идентификаторов выбора, доступных в данной реализации.

PICK_IDS Уровень 1b
package PICK_IDS is new GKS_LIST_UTILITIES (PICK_ID);
Предоставляет список идентификаторов выбора.

PICK_PROMPT_ECHO_TYPE Уровень 0b
type PICK_PROMPT_ECHO_TYPE is new INTEGER;
Определяет типы подсказки и эха устройства выбора.

PICK_PROMPT_ECHO_TYPES Уровень 0b
package PICK_PROMPT_ECHO_TYPES is
new GKS_LIST_UTILITIES
(PICK_PROMPT_ECHO_TYPE);
Предоставляет списки типов подсказок и эха устройства выбора.

PICK_REQUEST_STATUS Уровень 1b
type PICK_REQUEST_STATUS is (OK, NOPICK, NONE);
Определяет статус входных операций указания для функций запроса.

PICK_STATUS Уровень 1b
subtype PICK_STATUS is PICK_REQUEST_STATUS
range OK. . NOPICK;
Определяет статус входных операций указания для всех функций.

PIXEL_COLOUR_INDEX Уровень 0a
type PIXEL_COLOUR_INDEX is new INTEGER range —1 . .
INTEGER'LAST;
Типы для цветов пикселей, где значение —1 обозначает недопустимый индекс цвета.

PIXEL_COLOUR_MATRIX Уровень 0a
type PIXEL_COLOUR_MATRIX is array (POSITIVE

```
        range <>, POSITIVE range <>)
        of PIXEL_COLOUR_INDEX;
```

Предоставляет матрицу цветов пикселей.

POLYLINE_INDEX — Уровень 0а

```
type POLYLINE_INDEX is new POSITIVE;
```

Определяет область значений индексов ломаной.

POLYLINE_INDICES — Уровень 0а

```
package POLYLINE_INDICES is new
        GKS_LIST_UTILITIES (POLYLINE_INDEX);
```

Предоставляет списки индексов ломаной.

POLYMARKER_INDEX — Уровень 0а

```
type POLYMARKER_INDEX is new POSITIVE;
```

Определяет область значений индикаторов таблицы связок полимаркеров.

POLYMARKER_INDICES — Уровень 0а

```
package POLYMARKER_INDICES is new
        GKS_LIST_UTILITIES (POLYMARKER_INDEX);
```

Предоставляет списки индексов полимаркеров.

POSITIVE_TRANSFORMATION_NUMBER — Уровень 0а

```
subtype POSITIVE_TRANSFORMATION_NUMBER is
   TRANSFORMATION_NUMBER
range 1 .. TRANSFORMATION_NUMBER'LAST;
```

Номер преобразования нормирования, соответствующий устанавливаемому преобразованию.

PRIMITIVE_ATTRIBUTE_VALUES — Уровень 0а

```
type PRIMITIVE_ATTRIBUTE_VALUES is
   record
      INDEX_POLYLINE      : POLYLINE_INDEX;
      INDEX_POLYMARKER    : POLYMARKER_INDEX;
      INDEX_TEXT          : TEXT_INDEX;
      CHAR_HEIGHT         : WC. MAGNITUDE;
      CHAR_UP             : WC. VECTOR;
      CHAR_WIDTH          : WC. MAGNITUDE;
      CHAR_BASE           : WC. VECTOR;
      PATH                : TEXT_PATH;
      ALIGNMENT           : TEXT_ALIGNMENT;
      INDEX_FILL_AREA     : FILL_AREA_INDEX;
```

```
    PATTERN_WIDTH_          : WC. VECTOR;
      _VECTOR
    PATTERN_HEIGHT_         : WC. VECTOR;
      VECTO
    PATTERN_REFE-           : WC. POINT;
      RENCE_PO
end record;
```

Запись, содержащая все атрибуты текущего примитива для процедуры «Запросить значения атрибутов текущего примитива».

RADIANS Уровень 1а

```
type RADIANS is digits PRECISION;
```

Величины, используемые при выполнении преобразований сегментов (угол вращения). Положительные значения обозначают вращение против часовой стрелки.

RANGE_OF_EXPRESIONS Уровень 0а

```
type RANGE_OF_EXPRESIONS is
 record
   MIN : CHAR_EXPANSION;
   MAX : CHAR_EXPANSION;
 end record;
```

Предоставляет область значений масштабов расширения литер.

RASTER_UNITS Уровень 0а

```
type RASTER_UNITS is new POSITIVE;
```

Определяет область значений элементов растра.

RASTER_UNIT_SIZE Уровень 0а

```
type RASTER_UNIT. SIZE is
 record
   X : RASTER_UNITS;
   Y : RASTER_UNITS;
 end record;
```

Определяет размер экрана дисплея в растровых единицах на растровом устройстве.

REGENERATION_MODE Уровень 0а

```
type REGENERATION_MODE is (SUPPRESSED, ALLOWED);
```

Указывает, запрещена или разрешена неявная повторная генерация.

RELATIVE_PRIORITY Уровень 0а

type RELATIVE_PRIORITY is (HIGHER, LOWER);

Обозначает относительный приоритет между двумя преобразованиями нормирования.

RETURN_VALUE_TYPE Уровень 0а

type RETURN_VALUE_TYPE is (SET, REALIZED);

Указывает на то, является ли возвращаемое значение таким, какое оно было установлено в программе, или как оно реализовано на устройстве.

SCALE_FACTOR Уровень 0а

package SCALE_FACTOR_TYPE is
type SCALE_FACTOR is digits PRECISION;
end SCALE_FACTOR_TYPE;

Тип, используемый для масштаба.

SEGMENT_DETECTABILITY Уровень 1а

type SEGMENT_DETECTABILITY is (UNDETECTABLE, DETECTABLE);

Указывает на то, являются или нет сегменты обнаруживаемыми.

SEGMENT_HIGHLIGHTING Уровень 1а

type SEGMENT_HIGHLIGHTING is (NORMAL, HIGHLIGHTED);

Указывает, является ли сегмент выделенным.

SEGMENT_NAME Уровень 1а

type SEGMENT_NAME is new POSITIVE;

Определяет диапазон имен сегмента.

SEGMENT_NAMES Уровень 1а

package SEGMENT_NAMES is new GKS_LIST_UTILITIES (SEGMENT_NAME);

Предоставляет список имен сегментов.

SEGMENT_PRIORITY Уровень 1а

type SEGMENT_PRIORITY is digits PRECISION range 0.0 . . 1.0;

Определяет приоритет сегмента.

SEGMENT_VISIBILITY Уровень 1a
type SEGMENT_VISIBILITY is (VISIBLE, INVISIBLE);
Определяет видимость сегмента.

SMALL_NATURAL Уровень 0a
subtype SMALL_NATURAL is NATURAL range 0 .
SMALL_NATURAL_MAX;
Это декларация подтипа, которая позволяет объектам быть записями различных типов без возникновения прерываний STORAGE_ERROR

STRING_DEVICE_NUMBER Уровень 0b
type STRING_DEVICE_NUMBER is new DEVICE_NUMBER;
Предоставляет устройства ввода строки.

STRING_PROMPT_ECHO_TYPE Уровень 0b
type STRING_PROMPT_ECHO_TYPE is new INTEGER;
Определяет типы подсказки и эха устройства ввода строки.

STRING_PROMPT_ECHO_TYPES Уровень 0b
package STRING_PROMPT_ECHO_TYPES is new
GKS_LIST_UTILITIES (STRING_PROMPT_ECHO_TYPE);
Предоставляет списки типов подсказок и эха устройства ввода строки.

STRING_SMALL_NATURAL Уровень 0a
subtype STRING_SMALL_NATURAL is NATURAL
range 0 .. STRING_SMALL_NATURAL_MAX;
Это декларация подтипа, которая позволяет объектам быть записями различных типов без возникновения прерываний STORAGE_ERROR.

STROKE_DEVICE_NUMBER Уровень 0b
type STROKE_DEVICE_NUMBER is new DEVICE_NUMBER;
Предоставляет номера устройств ввода последовательностей позиций.

STROKE_PROMPT_ECHO_TYPE Уровень 0b
type STROKE_PROMPT_ECHO_TYPE is new INTEGER;
Определяет типы подсказок и эха устройств ввода последовательностей позиций.

STROKE_PROMPT_ECHO_TYPES Уровень 0b

```
package STROKE_PROMPT_ECHO_TYPES is
        new GKS_LIST_UTILITIES (STROKE_PROMPT_
                                ECHO_TYPE);
```

Предоставляет списки типов подсказок и эха устройств ввода последовательностей позиций.

STYLE_INDEX Уровень 0a

```
type STYLE_INDEX is new INTEGER;
```

Индекс вида — это либо HATCH STYLE, либо PATTERN_STYLE.

TEXT_ALIGNMENT Уровень 0a

```
type TEXT_ALIGNMENT is
 record
   HORIZONTAL   : HORIZONTAL_ALIGNMENT;
   VERTICAL     : VERTICAL_ALIGNMENT;
end record;
```

Тип атрибута, управляющего позиционированием параллелограмма текста по отношению к позиции текста, имеющего горизонтальную и вертикальную составляющие, как задано выше.

TEXT_EXTERN_PARALLELOGRAM Уровень 0a

```
type TEXT_EXTERN_PARALLELOGRAM is
 record
   LOWER_LEFT    : WC. POINT;
   LOWER_RIGHT   : WC. POINT;
   UPPER_RIGHT   : WC. POINT;
   UPPER_LEFT    : WC. POINT;
end record;
```

Определяет угловые точки параллелограмма размера текста по отношению к вертикально позиционируемому тексту.

TEXT_FONT Уровень 0a

```
type TEXT_FONT is new INTEGER;
```

Определяет типы шрифтов, предоставляемых реализацией.

TEXT_FONT_PRECISION Уровень 0a

```
type TEXT_FONT_PRECISION is
 record
   FONT         : TEXT_FONT;
   PRECISION    : TEXT_PRECISION;
```

```
end record;
```

Данный тип определяет запись, описывающую шрифт и точность текста.

TEXT_FONT_PRECISIONS — Уровень 0а

```
package TEXT_FONT_PRECISIONS is
        new GKS_LIST_UTILITIES (TEXT_FONT_
                                PRECISION);
```

Предоставляет списки пар, описывающих шрифт и точность текста.

TEXT_INDEX — Уровень 0а

```
type TEXT_INDEX is new POSITIVE;
```

Определяет область значений индексов таблицы связок текстов.

TEXT_INDICES — Уровень 0а

```
package TEXT_INDICES is new GKS_LIST_UTILITIES
        (TEXT_INDEX);
```

Предоставляет списки индексов текста.

TEXT_PATH — Уровень 0а

```
type TEXT_PATH is (RIGHT, LEFT, UP, DOWN);
```

Направление, выбранное для строки текста.

TEXT_PRECISION — Уровень 0а

```
type TEXT_PRECISION is (STRING_PRECISION,
                        CHAR_PRECISION,
                        STROKE_PRECISION);
```

Точность, с которой появляется текст.

TRANSFORMATION_FACTOR — Уровень 1а

```
type TRANSFORMATION_FACTOR is
 record
   X : NDC_TYPE;
   Y : NDC_TYPE;
 end record;
```

Масштаб, используемый в матрицах преобразований для выполнения преобразования сегментов.

TRANSFORMATION_MATRIX — Уровень 1а

```
type TRANSFORMATION_MATRIX is array (1..2, 1..3) of
     NDC_TYPE;
```

Для преобразований сегментов, отображаемых внутрь пространства НК.

TRANSFORMATION_NUMBER Уровень 0a
type TRANSFORMATION_NUMBER is new NATURAL;
Номер преобразования нормирования.

TRANSFORMATION_PRIORITY_ARRAY Уровень 0a
type TRANSFORMATION_PRIORITY_ARRAY is array
(POSITIVE range <>) of TRANSFORMATION_
NUMBER;
Тип для запоминания номеров преобразования.

TRANSFORMATION_PRIORITY_LIST Уровень 0a
type TRANSFORMATION_PRIORITY_LIST (LENGTH:
SMALL_NATURAL :=0) is
record
CONTENTS: TRANSFORMATION_PRIORITY_ARRAY
(1..LENGTH);
end record;
Предоставляет список приоритетов номеров преобразований.

UPDATE_REGENERATION_FLAG Уровень 0a
type UPDATE_REGENERATION_FLAG is (PERFORM,
POSTPONE);
Флаг, показывающий на действие повторной генерации на изображение.

UPDATE_STATE Уровень 0a
type UPDATE_STATE is (NOTPENDING, PENDING);
Указывает на то, что было запрошено, но еще не выполнено изменение преобразования для станции.

VALUATOR_DEVICE_NUMBER Уровень 0b
type VALUATOR_DEVICE_NUMBER is new
DEVICE_NUMBER;
Предоставляет идентификаторы устройств ввода числа.

VALUATOR_INPUT_VALUE Уровень 0b
type VALUATOR_INPUT_VALUE is digits PRECISION;
Определяет диапазон точности входных значений для реализации.

VALUATOR_PROMPT_ECHO_TYPE Уровень 0b

```
type VALUATOR_PROMPT_ECHO_TYPE is new INTEGER;
```

Определяет возможный диапазон типов подсказок и эха для устройства ввода числа.

VALUATOR_PROMPT_ECHO_TYPES Уровень 0b

```
package VALUATOR_PROMPT_ECHO_TYPES is
  new GKS_LIST_UTILITIES (VALUATOR_PROMPT_ECHO_
                                        _TYPE);
```

Предоставляет список типов подсказок и эха для устройства ввода числа.

VARIABLE_COLOUR_MATRIX Уровень 0a

```
type VARIABLE_COLOUR_MATRIX
      (DX:SMALL_NATURAL:=0;
      DY.SMALL_NATURAL :=0) is
  record
    MATRIX:COLOUR_MATRIX (1 . . DX, 1 . . DY);
```

Предоставляет матрицы переменной длины, содержащие индексы цвета, соответствующие матрице ячеек или матрице шаблонов.

VARIABLE_CONNECTION_ID Уровень 0a

```
type VARIABLE_CONNECTION_ID
      (LENGTH:STRING_SMALL_NATURAL:=0) is
  record
    CONNECT:STRING (1 . . LENGTH);
  end record;
```

Определяет идентификатор связи переменной длины для INQ_WS_CONNECTION_AND_TYPE.

VARIABLE_PIXEL_COLOUR_MATRIX Уровень 0a

```
type VARIABLE_PIXEL_COLOUR_MATRIX
    (DX:SMALL_NATURAL:=0; DY:SMALL_NATURAL:=0) is
  record
    MATRIX:PIXEL_COLOUR_MATRIX (1 . . DX, 1 . . DY);
  end record;
```

Предоставляет матрицы переменного размера для цвета пикселей.

VERTICAL_ALIGNMENT Уровень 0a

```
type VERTICAL_ALIGNMENT is (NORMAL, TOP, CAP, HALF,
                                    BASE, BOTTOM);
```

Выравнивание параллелограмма текста по отношению к вертикальной позиции текста.

WC Уровень 0а

package WC is new GKS_COORDINATE_SYSTEM (WC_TYPE);

Определяет мировую систему координат.

WC_TYPE Уровень 0а

type WC_TYPE is digits PRECISION;

Определяет точность для типов мировых координат.

WS_CATEGORY Уровень 0а

type WS_CATEGORY is (OUTPUT, INPUT, OUTIN, WISS, M0, M1);

Тип для категорий станций ЯГС.

WS_ID Уровень 0а

type WS_ID is new POSITIVE;

Определяет область значений идентификаторов станций.

WS_IDS Уровень 0а

package WS_IDS is new GKS_LIST_UTILITIES (WS_ID);

Предназначен для списков идентификаторов станций.

WS_STATE Уровень 0а

type WS_STATE is (INACTIVE, ACTIVE);

Состояние станции.

WS_TYPE Уровень 0а

type WS_TYPE is new POSITIVE;

Диапазон значений, соответствующих правильным типам станций. Константы, определяющие имена для различных типов станций, должны быть обеспечены реализацией.

WS_TYPES Уровень 0а

package WS_TYPES is new GKS_LIST_UTILITIES (WS_TYPE);

Предназначена для списков типов станций.

4.2.3. Список определений личных типов

В данном разделе дается в алфавитном порядке список определений личных типов, используемых для задания связывания

Ады и ЯГС. Каждая из этих деклараций определяет уровень ЯГС, на котором должна иметься декларация типа в реализации ЯГС данного или любого более высокого уровня, в котором декларация типа впервые понадобится (аналогично функциям). Все эти элементы являются декларациями типа PRIVATE языка Ада. Эти декларации включены в пакет ЯГС для того, чтобы дать возможность манипулировать личными типами.

CHOICE_DATA_RECORD Уровень 0b

type CHOICE_DATA_RECORD (PROMPT_ECHO_TYPE:
CHOICE_PROMPT_ECHO_TYPE:=DEFAULT_CHOICE)
is private;

Определяет запись для инициализации ввода выбора. Структура записи зависит от реализации. Так как это личный тип, компоненты записи могут быть найдены только через использование подпрограмм для манипулирования записями данных ввода (п. 5.2.1).

GKSM_DATE_RECORD Уровень 0a

type GKSM_DATA_RECORD (TYPE_OF_ITEM:
GKSM_ITEM_TYPE:=0; LENGTH:NATURAL:=0)
is private;

Запись данных для метафайла GKSM. Так как это личный тип, то до компонента записи можно добраться только через использование подпрограмм для манипулирования записями данных ввода (п. 5.2.1).

LOCATOR_DATA_RECORD Уровень 0b

type LOCATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
LOCATOR_PROMPT_ECHO_TYPE:=DEFAULT_LOCATOR)
is private;

Определяет запись для инициализации ввода позиции. Структура записи задается при реализации. Так как это личный тип, то до компонента записи можно добраться только через использование подпрограмм для манипулирования записями данных ввода (п. 5.2.1).

PICK_DATA_RECORD Уровень 0b

type PICK_DATA_RECORD (PROMPT_ECHO_TYPE:
PICK_PROMPT_ECHO_TYPE:=DEFAULT_PICK) is private;

Определяет запись для инициализации ввода указания. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя под-

программы для манипулирования записями данных ввода (п. 5.2.1).

STRING_DATA_RECORD Уровень 0b

type STRING_DATA_RECORD (PROMPT_ECHO_TYPE:
STRING_PROMPT_ECHO_TYPE:=DEFAULT_STRING)
is private;

Определяет запись для инициализации ввода строки. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя подпрограммы для манипулирования записями данных ввода (п. 5.2.1).

STROKE_DATA_RECORD Уровень 0b

type STROKE_DATA_RECORD (PROMPT_ECHO_TYPE.
STROKE_PROMPT_ECHO_TYPE:=DEFAULT_STROKE)
is private;

Определяет запись для инициализации ввода последовательности позиций. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя подпрограммы для манипулирования записямч данных ввода (п. 5.2.1).

VALUATOR DATA_RECORD Уровень 0b

type VALUATOR_DATA_RECORD (PROMPT ECHO TYPE:
VALUATOR_PROMPT_ECHO_TYPE :=DEFAULT_
VALUATOR)
is private;

Опрéделяет запись для инициализации ввода числа. Структура записи определяется в реализации. Так как это личный тип, то до компонента записи можно добраться, только используя подпрограммы для манипулирования записями данных ввода (п. 5.2.1).

4.2.4. Список деклараций констант

В данном разделе приведены декларации зависящих от реализации констант для задания типов Ада/ЯГС. Некоторые из констант используют для задания принимаемых по умолчанию значений параметров для процедур ЯГС, определяемых в разд. 5. В данном разделе приведены также константы, которые представляют стандартные значения, задаваемые для некоторых типов ЯГС/Ада.

Следующие константы определяют стандартные типы линий ЯГС:

```
SOLID_LINE          : constant LINETYPE := 1;
DASHED_LINE         : constant LINETYPE := 2;
DOTTED_LINE         : constant LINETYPE := 3;
DASHED_DOTTED_LINE  : constant LINETYPE := 4;
```

Следующие константы определяют стандартные типы маркеров ЯГС:

```
DOT_MARKER          : constant MARKER_TYPE := 1;
PLUS_MARKER         : constant MARKER_TYPE := 2;
STAR_MARKER         : constant MARKER_TYPE := 3;
ZERO_MARKER         : constant MARKER_TYPE := 4;
X_MARKER            : constant MARKER_TYPE := 5;
```

Следующие константы определяют стандартные типы подсказок и эха, поддерживаемые ЯГС:

```
DEFAULT_LOCATOR     : constant LOCATOR_PROMPT_ECHO_TYPE := 1;
CROSS_HAIR_LOCATOR  : constant LOCATOR_PROMPT_ECHO_TYPE := 2;
TRACKING_CROSS_LOCATOR : constant LOCATOR_PROMPT_ECHO_TYPE := 3;
RUBBER_BAND_LINE_LOCATOR : constant LOCATOR_PROMPT_ECHO_TYPE := 4;
RECTANGLE_LOCATOR   : constant LOCATOR_PROMPT_ECHO_TYPE := 5;
DIGITAL_LOCATOR     : constant LOCATOR_PROMPT_ECHO_TYPE := 6;
DEFAULT_STROKE      : constant STROKE_PROMPT_ECHO_TYPE := 1;
DIGITAL_STROKE      : constant STROKE_PROMPT_ECHO_TYPE := 2;
MARKER_STROKE       : constant STROKE_PROMPT_ECHO_TYPE := 3;
LINE_STROKE         : constant STROKE_PROMPT_ECHO_TYPE := 4;
DEFAULT_VALUATOR    : constant VALUATOR_PROMPT_ECHO_TYPE := 1;
GRAPHICAL_VALUATOR  : constant VALUATOR_PROMPT_ECHO_TYPE := 2;
DIGITAL_VALUATOR    : constant VALUATOR_PROMPT_ECHO_TYPE := 3;
DEFAULT_CHOICE      : constant CHOICE_PROMPT_ECHO_TYPE := 1;
```

```
PROMPT_ECHO_          : constant CHOICE_PROMPT_ECHO_
_CHOICE                                  _TYPE :=2;
STRING_PROMPT         : constant CHOICE_PROMPT_ECHO_
_CHOICE                                  _TYPE :=3;
STRING_INPUT_         : constant CHOICE_PROMPT_ECHO_
_CHOICE                                  _TYPE :=4;
SEGMENT_              : constant CHOICE_PROMPT_ECHO_
_CHOICE                                  _TYPE :=5;
DEFAULT_STRING        : constant STRING_PROMPT_ECHO_
                                         _TYPE :=1;
DEFAULT_PICK          : constant PICK_PROMPT_ECHO_
                                         _TYPE :=1;
GROUP_HIGH_           : constant PICK_PROMPT_ECHO_
_LIGHT_PICK                              _TYPE :=2;
SEGMENT_              : constant PICK_PROMPT_ECHO_
_HIGHLIGHT_PICK                          _TYPE :=3;
```

Следующие константы используют для определения принимаемых по умолчанию значений параметров для процедур ЯГС, определяемых в разд. 5:

```
DEFAULT_MEMO-         : constant :=implementation_defined;
RY_UNITS
PRECISION             : constant :=implementation_defined;
SMALL_NATURAL_        : constant :=implementation_defined;
_MAX
CHOICE_SMALL_         : constant :=implementation_defined;
_NATURAL_MAX
STRING_SMALL_         : constant :=implementation_defined;
_NATURAL_MAX
DEFAULT_ERROR_        : constant :=implementation_defined;
_FILE
```

Следующая строка задает прерывание GKS_ERROR, определенное в п. 3.2.3:

```
GKS_ERROR : exception;
```

4.3. **Коды ошибок**

Связывание требует применения процедуры ERROR_HANDLING для обработки любых ошибок, которые возникают в процедурах ЯГС, исключая процедуры запросов. Полное описание требований по обработке ошибок имеется в п. 3.2.3.

Функции запроса ЯГС не порождают прерываний. Вместо этого они возвращают параметр индикатора ошибок, который содер-

жит номер ошибки, которая обнаружена. Это согласуется с философией ЯГС, говорящей, что при запросе не возникает ошибок. Номера ошибок соответствуют номерам ошибок из приложения Б спецификации ЯГС плюс дополнительные ошибки, определенные в данном документе. Отметим, что различные известные ошибочные условия могут быть обнаружены вне контроля ЯГС благодаря природе языка Ада и могут привести к прерыванию при запросе.

4.3.1 Задание кодов ошибок

Стандарт ИСО 7942 дает отображение номеров ошибок для каждой функции ЯГС. Ряд известных ошибок ЯГС не может быть обнаружен реализацией ЯГС в Аде из-за свойств языка Ада, таких как строгость определения типов данных. Эти ошибки приведены в разделе о кодах устраненных ошибок.

В дополнение к определенным в ЯГС ошибкам могут существовать ошибки, задаваемые при реализации, и ошибки, определяемые связыванием

IMPLEMENTATION_DEFINED_ERROR

Ошибки, задаваемые при реализации, описаны в Руководстве пользователя по реализации и имеют коды ошибок меньше нуля.

LANGUAGE_BINDING_ERROR

Ошибки связывания с языком являются специфическими для связывания ЯГС с Адой. Номера ошибок с 2500 по 2999 зарезервированы для зависящих от связывания с языком Ада ошибок. Следующие ошибки определены данным связыванием для специфических ошибок связывания с языком:

2500 Неправильное использование записи данных ввода.

Когда происходят следующие ошибки, автоматически возникает предопределенное прерывание в Аде.

2501 Неизвестная ошибка, произошедшая при обработке.

2502 Ошибка применения утилиты LIST GKS.

4.3.2. Коды устраняемых ошибок

Следующие ошибки ЯГС представлены отдельно из-за некоторых свойств языка Ада или их использования в данной связке; они никогда не могут произойти в данной реализации ЯГС. Ошибки могут быть обнаружены компилятором или во время выполнения вне области действия ЯГС

Коды ошибок, устраняемых функциями:

20 Неверно задан идентификатор станции

22 Неверно задан тип станции

65 Масштаб толщины линии меньше нуля

71 Масштаб маркера меньше нуля
77 Масштаб расширения литеры меньше или равен нулю
78 Высота литеры меньше или равна нулю
87 Значение размера шаблона не положительно
91 Неверно заданы размерности массива индексов цвета
92 Индекс цвета меньше нуля
96 Интенсивность цвета лежит вне диапазона от нуля до единицы
97 Неверно задан идентификатор указания
120 Неверно задано имя сегмента
126 Приоритет сегмента вне диапазона от нуля до единицы
151 Неверно задано время ожидания
166 Неверно задана максимальная длина записи данных

5. ФУНКЦИИ В АДЕ, СВЯЗАННЫЕ С ЯДРОМ ГРАФИЧЕСКОЙ СИСТЕМЫ

5.1. Функции ЯГС

OPEN GKS
ОТКРЫТЬ GKS — Уровень 0а

```
procedure OPEN_GKS
  (ERROR_FILE          : in STRING := DEFAULT_
                                      ERROR_FILE;
   AMOUNT_OF_MEMORY    : in NATURAL := DEFAULT_
                         MEMORY_UNITS);
```

CLOSE GKS
ЗАКРЫТЬ ЯГС — Уровень 0а

```
procedure CLOSE_GKS
```

OPEN WORKSTATION
ОТКРЫТЬ СТАНЦИЮ — Уровень 0а

```
procedure OPEN_WS
  (WS              : in WS_ID;
   CONNECTION      : in STRING;
   TYPE_OF_WS      : in WS_TYPE);
```

CLOSE WORKSTATION
ЗАКРЫТЬ СТАНЦИЮ — Уровень 0а

```
procedure CLOSE_WS
  (WS    : in WS_ID);
```

ACTIVATE WORKSTATION
АКТИВИРОВАТЬ СТАНЦИЮ — Уровень 0а

```
procedure ACTIVATE_WS
  (WS    : in WS_ID);
```

DEACTIVATE WORKSTATION
ДЕАКТИВИРОВАТЬ СТАНЦИЮ Уровень 0a

```
procedure DEACTIVATE_WS
 (WS    : in WS_ID;
```

CLEAR WORKSTATION
ОЧИСТИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ Уровень 0a

```
procedure CLEAR_WS
 (WS    : in WS_ID;
 FLAG   : in CONTROL_FLAG);
```

REDRAW ALL SEGMENTS ON WORKSTATION
ПЕРЕРИСОВАТЬ ВСЕ СЕГМЕНТЫ НА СТАНЦИИ Уровень 1a

```
procedure REDRAW ALL SEGMENTS_ON_WS
 (WS    : in WS_ID);
```

UPDATE WORKSTATION
ОБНОВИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ Уровень 1a

```
procedure UPDATE_WS
 (WS                : in WS_ID,
 REGENERATION       : in UPDATE_REGENERATION_
                          _FLAG);
```

SET DEFERRAL STATE
ЗАДАТЬ РЕЖИМ ЗАДЕРЖКИ Уровень 1a

```
procedure SET_DEFERRAL_STATE
 (WS                : in WS_ID;
 DEFERRAL           : in DEFERRAL_MODE;
 REGENERATION       : in REGENERATION_MODE);
```

MESSAGE
СООБЩЕНИЕ Уровень 1a

```
procedure MESSAGE
 (WS                : in WS_ID;
 CONTENTS           : in STRING).
```

ESCAPE
РАСШИРЕНИЕ Уровень 0a

Функции расширения рассматриваются в данном связывании как отдельные процедуры для каждого типа расширения, предоставляемого реализацией, каждая со списком формальных пара-

метров, соответствующих реализованной процедуре. Зарегистрированные процедуры ESCAPE будут находиться в библиотечном пакете, названном GKS_ESCAPE. Имена и параметры ESCAPE зарегистрированы в Международном журнале графических элементов ИСО, который ведется Органом регистрации.

Каждая незарегистрированная процедура ESCAPE будет находиться в библиотечном пакете; при этом используются следующие соглашения по именованию:

```
package GKS_UESC_<имя процедуры расширения> is
 procedure ESC;
   — код на Аде для процедуры UESC
end GKS_UESC_<имя процедуры расширения>;
— Единственным именем процедуры, используемым в пакете,
будет ESC
```

Для того, чтобы поддержать возможность записывать ESCAPE в метафайл, эти зарегистрированные расширения могут быть привлечены, используя типы данных и форму процедуры GENERALIZED_ESC, которая имеет спецификацию, приведенную ниже:

```
package GKS_ESCAPE is
 type ESCAPE_ID is new INTEGER;
 type ESCAPE_FLOAT is digits PRECISION;
 type ESC_INTEGER_ARRAY is array (SMALL_NATURAL
     range <>) of INTEGER;
 type ESC_FLOAT_ARRAY is array (SMALL_NATURAL
     range <>) of ESCAPE_FLOAT;
 type ESC_STRING_ARRAY is array (SMALL_NATURAL
     range <>) of STRING (1 .. 80);
 type ESC_DATA_RECORD (NUM_OF_INTEGER
                       : SMALL_NATURAL_:=0;
NUM_OF_REALS           : SMALL_NATURAL :=0;
NUM_OF_STRING          : SMALL_NATURAL :=0) is
  record
   INTEGER_ARRAY : ESC_INTEGER_ARRAY
                                (1 .. NUM_OF_INTEGERS);
   REAL_ARRAY : ESC_FLOAT_ARRAY (1 .. NUM_OF_REALS);
   ESC_STRINGS : ESC_STRING_ARRAY
                                (1 .. NUM_OF_STRINGS);
  end record;
  procedure GENERALIZED_ESC
    (ESCAPE_NAME : in ESCAPE_ID;
    ESC_DATA_IN : in ESC_DATA_RECORD;
    ESC_DATA_OUT : in ESC_DATA_RECORD);
```

end GKS_ESCAPE;

Представляет типы данных и процедуры для реализации неподдерживаемых расширений.

POLYLINE Уровень 0а
ЛОМАНАЯ

```
procedure POLYLINE
  (POINTS      : in WC. POINT_ARRAY);
```

POLYMARKER Уровень 0а
ПОЛИМАРКЕР

```
procedure POLYMARKER
  (POINTS      : in WC. POINT_ARRAY);
```

TEXT Уровень 0а
ТЕКСТ

```
procedure TEXT
  (POSITION        : in WC. POINT;
  CHAR_STRING      : in STRING);
```

FILL AREA Уровень 0а
ПОЛИГОНАЛЬНАЯ ОБЛАСТЬ

```
procedure FILL_AREA
  (POINTS      : in WC. POINT_ARRAY);
```

CELL_ARRAY Уровень 0а
МАТРИЦА ЯЧЕЕК

```
procedure CELL_ARRAY
  (CORNER_1_1          : in WC. POINT;
  CORNER_DX DY         : in WC. POINT;
  CELLS                : in COLOUR_MATRIX);
```

GENERALIZED DRAWING PRIMITIVE Уровень 0а
ОБОБЩЕННЫЙ ПРИМИТИВ ВЫВОДА

Обобщенный графический примитив вывода ОПВ связывается по принципу один во многие с отдельной процедурой, реализованной для каждого ОПВ, каждая из которых имеет свой собственный интерфейс. Зарегистрированные ОПВ находятся в библиотечном пакете, названном GKS_GDR. Имена ОПВ и параметры зарегистрированы в Международном журнале графических записей ИСО, который ведется Органом регистрации.

Каждая незарегистрированная процедура GDP будет находиться в библиотечном пакете, использующем следующие правила именования:

```
package GKS_UGDP_<имя процедуры GDP>
 procedure GDP;
    — код на Аде процедуры UGDP
end GKS_UGDP_<имя процедуры GDP>;
— Единственным именем процедуры, используемым в пакете,
будет GDP
```

Для того, чтобы поддержать возможность записывать ОПВ данной реализации в метафайл, эти зарегистрированные ОПВ могут быть привлечены, используя типы данных и форму процедуры GENERALIZED_GDP, которая имеет следующую спецификацию:

```
package GKS_GDP is
type GDP_FLOAT is digits PRECISION;
type GDP_INTEGER_ARRAY is array (SMALL_NATURAL
range <>)
    of INTEGER;
type GDP_FLOAT_ARRAY is array (SMALL_NATURAL
    range <>) of GDP_FLOAT;
type GDP_STRING_ARRAY is array (SMALL_NATURAL
    range <>) of STRING (1..80);
type GDP_DATA_RECORD (NUM_OF_INTEGERS : SMALL_
                      _NATURAL := 0;
                      NUM_OF_REAL : SMALL_
                      _NATURAL := 0;
                      NUM_OF_STRINGS : SMALL_
                      _NATURAL := 0)
is
record
INTEGER_ARRAY : GDP_INTEGER_ARRAY (1..NUM_OF_
                                   _INTEGERS);
REAL_ARRAY : GDP_FLOAT_ARRAY (1..NUM_OF_
                              _REALS);
GDP_STRINGS : GDP_STRING_ARRAY (1..NUM_OF_
                                _STRINGS);
end record;
 procedure GENERALIZED_GDP (GDP_NAME : in GDP_ID;
                    POINTS : in WC.POINT_LIST;
                    GDP_DATA : in GDP_DATA_RECORD);
end GKS_GDP;
```

Предоставляет типы данных и процедуру для реализации неподдержанных обобщенных примитивов вывода.

SET POLYLINE INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ЛОМАНОЙ
```
procedure SET_POLYLINE_INDEX
  (INDEX                : in POLYLINE_INDEX);
```

SET LINETYPE Уровень 0a
ЗАДАТЬ ТИП ЛИНИИ
```
procedure SET_LINETYPE
  (TYPE_OF_LINE         : in LINETYPE);
```

SET LINEWIDTH SCALE FACTOR Уровень 0a
ЗАДАТЬ МАСШТАБ ТОЛЩИНЫ ЛИНИИ
```
procedure SET_LINEWIDTH_SCALE_FACTOR
  (WIDTH                . in LINEWIDTH);
```

SET POLYLINE COLOUR INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ЦВЕТА ЛОМАНОЙ
```
procedure SET_POLYLINE_COLOUR_INDEX
  (LINE_COLOUR          : in COLOUR_INDEX);
```

SET POLYMARKER INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ПОЛИМАРКЕРА
```
procedure SET_POLYMARKER_INDEX
  (INDEX                : in POLYMARKER_INDEX);
```

SET MARKER TYPE Уровень 0a
ЗАДАТЬ ТИП МАРКЕРА
```
procedure SET_MARKER_TYPE
  (TYPE)OF_MARKER       : in MARKER_TYPE);
```

SET MARKER SIZE SCALE FACTOR Уровень 0a
ЗАДАТЬ МАСШТАБ МАРКЕРА
```
procedure SET_MARKER_SIZE_SCALE_FACTOR
  (SIZE                 : in MARKER_SIZE);
```

SET POLYMARKER COLOUR INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛУМАРКЕРА
```
procedure SET_POLYMARKER_COLOUR_INDEX
  (MARKER_COLOUR        : in COLOUR_INDEX);
```

SET TEXT INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ТЕКСТА
```
procedure SET_TEXT_INDEX
  (INDEX                : in TEXT_INDEX);
```

SET TEXT FONT AND_PRECISION Уровень 0а
ЗАДАТЬ ШРИФТ И ТОЧНОСТЬ ТЕКСТА
```
procedure SET_TEXT_FONT_AND_PRECISION
  (FONT_PRECISION        : in TEXT_FONT_PRECISION);
```

SET CHARACTER EXPANSION FACTOR Уровень 0а
ЗАДАТЬ МАСШТАБ РАСШИРЕНИЯ ЛИТЕРЫ
```
procedure SET_CHAR_EXPANSION_FACTOR
  (EXPANSION             : in CHAR_EXPANSION);
```

SET CHARACTER SPACING Уровень 0а
ЗАДАТЬ МЕЖЛИТЕРНЫЙ ПРОСВЕТ
```
procedure SET_CHAR_SPACING
  (SPACING               : in CHAR_SPACING);
```

SET TEXT COLOUR INDEX Уровень 0а
ЗАДАТЬ ИНДЕКС ЦВЕТА ТЕКСТА
```
procedure SET_TEXT_COLOUR_INDEX
  (TEXT_COLOUR           : in COLOUR_INDEX);
```

SET CHARACTER HEIGHT Уровень 0а
ЗАДАТЬ ВЫСОТУ ЛИТЕР
```
procedure SET_CHAR_HEIGHT
  (HEIGHT                : in WC. MAGNITUDE);
```

SET CHARACTER UP VECTOR Уровень 0а
ЗАДАТЬ ВЕРТИКАЛЬ ЛИТЕРЫ
```
procedure SET_CHAR_UP_VECTOR
  (CHAR_UP_VECTOR        : in WC. VECTOR);
```

SET TEXT PATH Уровень 0а
ЗАДАТЬ НАПРАВЛЕНИЕ ТЕКСТА
```
procedure SET_TEXT_PATH
  (PATH : in TEXT_PATH);
```

SE TEXT ALIGNMENT Уровень 0а
ЗАДАТЬ ВЫРАВНИВАНИЕ ТЕКСТА
```
procedure SET_TEXT_ALIGNMENT
  (ALIGNMENT             : in TEXT_ALIGNMENT);
```

SET FILL AREA INDEX Уровень 0а
ЗАДАТЬ ИНДЕКС ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
```
procedure SET_FILL_AREA_INDEX
  (INDEX                 : in FILL_AREA_INDEX);
```

SET FILL AREA INTERIOR STYLE Уровень 0a
ЗАДАТЬ ВИД ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
```
procedure SET_FILL_AREA_INTERIOR_STYLE
  (INTERIOR                : in INTERIOR_STYLE);
```

SET FILL AREA STYLE INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
```
procedure SET_FILL_AREA_STYLE_INDEX
  (STYLE                   : in STYLE_INDEX);
```

SET FILL AREA COLOUR INDEX Уровень 0a
ЗАДАТЬ ИНДЕКС ЦВЕТА ПОЛИГОНАЛЬНОЙ ОБЛАСТИ
```
procedure SET_FILL_AREA_COLOUR_INDEX
  (FILL_AREA_COLOUR        : in COLOUR_INDEX);
```

SET PATTERN SIZE Уровень 0a
ЗАДАТЬ РАЗМЕР ШАБЛОНА
```
procedure SET_PATTERN_SIZE
  (SIZE                    : in WC. SIZE);
```

SET PATTERN REFERENCE POINT Уровень 0a
ЗАДАТЬ ТОЧКУ ПРИВЯЗКИ ШАБЛОНА
```
procedure SET_PATTERN_REFERENCE_POINT
  (POINT                   : in WC. POINT);
```

SET ASPECTSOURCE FLAGS Уровень 0a
ЗАДАТЬ ФЛАГИ ВЫБОРКИ АТРИБУТОВ
```
procedure SET_ASF
  (ASF                     : in ASF_LIST);
```

SET PICK IDENTIFIER Уровень 1b
ЗАДАТЬ ИДЕНТИФИКАТОР УКАЗАНИЯ
```
procedure SET_PICK_ID
  (PICK                    : in PICK_ID);
```

SET POLYLINE REPRESENTATION Уровень 1a
ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ
```
procedure SET_POLYLINE_REPRESENTATION
  (WS                      : in WS_ID;
  INDEX                    : in POLYLINE_INDEX;
  TYPE_OF_LINE             : in LINETYPE;
```

```
    WIDTH                    : in LINEWIDTH;
    LINE_COLOUR              : in COLOUR_INDEX);
```

SET POLYMARKER REPRESENTATION — Уровень 1a
ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА

```
  procedure SET_POLYMARKER_REPRESENTATION
    (WS                      : in WS_ID;
    INDEX                    : in POLYMARKER_INDEX;
    TYPE_OF_MARKER           : in MARKER_TYPE;
    SIZE                     : in MARKER_SIZE;
    MARKER_COLOUR            : in COLOUR_INDEX);
```

SET TEXT REPRESENTATION — Уровень 1a
ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА

```
  procedure SET_TEXT_REPRESENTATION
    (WS                      : in WS_ID;
    INDEX                    : in TEXT_INDEX;
    FONT_PRECISION           : in TEXT_FONT_PRECISION;
    EXPANSION                : in CHAR_EXPANSION;
    SPACING                  : in CHAR_SPACING;
    TEXT_COLOUR              : in COLOUR_INDEX);
```

SET FILL AREA REPRESENTATION — Уровень 1a
ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ

```
  procedure SET_FILL_AREA_REPRESENTATION
    (WS                      : in WS_ID;
    INDEX                    : in FILL_AREA_INDEX;
    INTERIOR                 : in INTERIOR_STYLE;
    STYLE                    : in STYLE_INDEX;
    FILL_AREA_COLOUR         : in COLOUR_INDEX);
```

SET PATTERN REPRESENTATION — Уровень 1a
ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА

```
  procedure SET_PATTERN_REPRESENTATION
    (WS                      : in WS_ID;
    INDEX                    : in PATTERN_INDEX;
    PATTERN                  : in COLOUR_MATRIX);
```

SET COLOUR REPRESENTATION — Уровень 0a
ЗАДАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА

```
  procedure SET_COLOUR_REPRESENTATION
    (WS                      : in WS_ID);
```

```
    INDEX                    : in COLOUR_INDEX;
    RGB_COLOUR               : in COLOUR_REPRESEN-
                                  TATION);
```

SET WINDOW — Уровень 0a
ЗАДАТЬ ОКНО

```
  procedure SET_WINDOW
    (TRANSFORMATION          : in POSITIVE_TRANSFORMA-
                                  TION_NUMBER;
    WINDOW_LIMITS            : in WC. RECTANGLE_LIMITS);
```

SET VIEWPORT — Уровень 0a
ЗАДАТЬ ПОЛЕ ВЫВОДА

```
  procedure SET_VIEWPORT
    (TRANSFORMATION          : in POSITIVE_TRANSFOR-
                                  MATION_NUMBER;
    VIEWPORT_LIMITS          : in NDC. RECTANGLE_
                                  _LIMITS);
```

SET VIEWPORT INPUT PRIORITY — Уровень 0b
ЗАДАТЬ ПРИОРИТЕТ ПОЛЯ ВЫВОДА ПРИ ВВОДЕ

```
  procedure SET_VIEWPORT_INPUT_PRIORITY
    (TRANSFORMATION          : in TRANSFORMATION_
                                  _NUMBER;
    REFERENCE_
      _TRANSFORMATION        : in TRANSFORMATION_
                                  _NUMBER;
    PRIORITY                 : in RELATIVE PRIORITY);
```

SELECT NORMALIZATION TRANSFORMATION — Уровень 0a
ВЫБРАТЬ ПРЕОБРАЗОВАНИЕ НОРМИРОВАНИЯ

```
  procedure SELECT_NORMALIZATION_TRANSFORMATION
    (TRANSFORMATION          : in TRANSFORMATION_
                                  _NUMBER;
```

SET CLIPPING INDICATOR — Уровень 0a
ЗАДАТЬ ИНДИКАТОР ОТСЕЧЕНИЯ

```
  procedure SET_CLIPPING_INDICATOR
    (CLIPPING                : in CLIPPING_INDICATOR);
```

SET WORKSTATION WINDOU — Уровень 0a
ЗАДАТЬ ОКНО СТАНЦИИ

```
  procedure SET_WS_WINDOW
```

```
    (WS                          : in WS_ID;
    WS_WINDOW_LIMITS             : in NDC.RECTANGLE_LIMITS);
```

SET WORKSTATION VIEWPORT — Уровень 0a
ЗАДАТЬ ПОЛЕ ВЫВОДА СТАНЦИИ

```
  procedure SET_WS_VIEWPORT
    (WS                          : in WS_ID;
    WS_VIEWPORT_LIMITS           : in DC. RECTANGLE_LIMITS);
```

CREATE SEGMENT — Уровень 1a
СОЗДАТЬ СЕГМЕНТ

```
    procedure CREATE_SEGMENT
    (SEGMENT                     : in SEGMENT_NAME);
```

CLOSE SEGMENT — Уровень 1a
ЗАКРЫТЬ СЕГМЕНТ

```
  procedure CLOUSE_SEGMENT;
```

RENAME SEGMENT — Уровень 1a
ПЕРЕИМЕНОВАТЬ СЕГМЕНТ

```
  procedure RENAME_SEGMENT
    (OLD_NAME                    : in SEGMENT_NAME;
    NEW_NAME                     : in SEGMENT_NAME);
```

DELETE SEGMENT — Уровень 1a
УНИЧТОЖИТЬ СЕГМЕНТ

```
  procedure DELETE_SEGMENT
    (SEGMENT                     : in SEGMENT_NAME);
```

DELETE SEGMENT FROM WORKSTATION — Уровень 1a
УДАЛИТЬ СЕГМЕНТ СО СТАНЦИИ

```
  procedure DELETE_SEGMENT_FROM_WS
    (WS                          : in WS_ID;
    SEGMENT                      : in SEGMENT_NAME);
```

ASSOCIATE SEGMENT WITH WORKSTATION — Уровень 2a
СВЯЗАТЬ СЕГМЕНТ СО СТАНЦИЕЙ

```
  procedure ASSOCIATE_SEGMENT_WITH_WS
    (WS                          : in WS_ID;
    SEGMENT                      : in SEGMENT_NAME);
```

COPY SEGMENT TO WORKSTATION Уровень 2a
ВЫВЕСТИ КОПИЮ СЕГМЕНТА НА СТАНЦИЮ

```
procedure COPY_SEGMENT_TO_WS
  (WS                    : in WS_ID;
  SEGMENT                : in SEGMENT_NAME);
```

INSERT SEGMENT Уровень 2a
ВСТАВИТЬ СЕГМЕНТ

```
procedure INSERT_SEGMENT
  (SEGMENT               : in SEGMENT_NAME;
  TRANSFORMATION         : in TRANSFORMATION_
                                    _MATRIX);
```

SET SEGMENT TRANSFORMATION Уровень 1a
ЗАДАТЬ ПРЕОБРАЗОВАНИЕ СЕГМЕНТА

```
procedure SET_SEGMENT_TRANSFORMATION
  (SEGMENT               : in SEGMENT_NAME;
  TRANSFORMATION         : in TRANSFORMATION_
                                    _MATRIX);
```

SET VISIBILITY Уровень 1a
ЗАДАТЬ ВИДИМОСТЬ

```
procedure SET_VISIBILITY
  (SEGMENT               : in SEGMENT_NAME;
  VISIBILITY             : in SEGMENT_VISIBILITY);
```

SET HIGHLIGHTING Уровень 1a
ЗАДАТЬ ВЫДЕЛЕНИЕ

```
procedure SET_HIGHLIGHTING
  (SEGMENT               : in SEGMENT_NAME;
  HIGHLIGHTING           : in SEGMENT_
                                    _HIGHLIGHTING);
```

SET SEGMENT PRIORITY Уровень 1a
ЗАДАТЬ ПРИОРИТЕТ СЕГМЕНТА

```
procedure SET_SEGMENT_PRIORITY
  (SEGMENT               : in SEGMENT_NAME;
  PRIORITY               : in SEGMENT_PRIORITY);
```

SET DETECTABILITY Уровень 1b
ЗАДАТЬ ЧУВСТВИТЕЛЬНОСТЬ К УКАЗАНИЮ

```
procedure SET_DETECTABILITY
```

```
   (SEGMENT                    : in SEGMENT_NAME;
   DETECTABILITY               : in SEGMENT_DETECTA-
                                                BILITY);
```

INITIALISE LOCATOR — Уровень 0b
ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ПОЗИЦИИ

```
   procedure INITIALISE_LOCATOR
    (WS                        . in WS_ID;
    DEVICE                     : in LOCATOR_DEVICE_
                                              _NUMBER;
    INITIAL_TRANSFOR-          : in TRANSFORMATION_
      MATION                                  _NUMBER;
    INITIAL_POSITION           : in WC. POINT;
    ECHO_AREA                  : in DC. RECTANGLE_LIMITS;
    DATA_RECORD                : in LOCATOR_DATA_
                                              _RECORD);
```

INITIALISE STROKE — Уровень 0b
ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ

```
   procedure INITIALISE_STROKE
    (WS                        : in WS_ID;
    DEVICE                     : in STROKE_DEVICE_
                                              _NUMBER;
    INITIAL_TRANSFOR-          : in TRANSFORMATION_
      MATION                                  _NUMBER;
    INITIAL_STROKE             : in WC. POINT_ARRAY;
    ECHO_AREA                  : in DC. RECTANGLE_LIMITS;
    DATA_RECORD                : in STROKE_DATA_RECORD);
```

INITIALISE VALUATOR — Уровень 0b
ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА ЧИСЛА

```
   procedure INITIALISE_VALUATOR
    (WS                        : in WS_ID;
    DEVICE                     : in VALUATOR_DEVICE_
                                              _NUMBER;
    INITIAL_VALUE              : in VALUATOR_INPUT_
                                              _VALUE;
    ECHO_AREA                  : in DC. RECTANGLE_LIMITS;
    DATA_RECORD                : in VALUATOR_DATA_
                                              _RECORD);
```

INITIALISE CHOICE — Уровень 0b
ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВЫБОРА

```
procedure INITIALISE_CHOICE
  (WS                          : in WS_ID;
  DEVICE                       : in CHOICE_DEVICE_
                                                _NUMBER;
  INITIAL_STATUS               : in CHOICE_STATUS;
  INITIAL_CHOICE               : in CHOICE_VALUE;
  ECHO_AREA                    : in DC. RECTANGLE_LIMITS;
  DATA_RECORD                  : in CHOICE_DATA_RECORD);
```

INITIALISE PICK — Уровень 1b
ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО УКАЗАНИЯ

```
procedure INITIALISE_PICK
  (WS                          : in WS_ID;
  DEVICE                       : in PICK_DEVICE_NUMBER;
  INITIAL_STATUS               : in PICK_STATUS;
  INITIAL_SEGMENT              : in SEGMENT_NAME;
  INITIAL_PICK                 : in PICK_ID;
  ECHO_AREA                    : in DC. RECTANGLE_LIMITS;
  DATA_RECORD                  : in PICK_DATA_RECORD);
```

INITIALISE STRING — Уровень 0b
ИНИЦИАЛИЗИРОВАТЬ УСТРОЙСТВО ВВОДА СТРОКИ

```
procedure INITIALIZE_STRING
  (WS                          : in WS_ID;
  DEVICE                       : in STRING_DEVICE_
                                                _NUMBER;
  INITIAL_STRING               : in INPUT_STRING;
  ECHO_AREA                    : in DC. RECTANGLE_LIMITS;
  DATA_RECORD                  : in STRING_DATA_RECORD);
```

SET LOCATOR MODE — Уровень 0b
ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА ПОЗИЦИИ

```
procedure SET_LOCATOR_MODE
  (WS                          : in WS_ID;
  DEVICE                       : in LOCATOR_DEVICE_
                                                _NUMBER;
  MODE                         : in OPERATING_MODE;
  SWITCH                       : in ECHO_SWITCH);
```

SET STROKE MODE — Уровень 0b
ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ

```
procedure SET_STROKE_MODE
```

```
    (WS                          : in WS_ID;
    DEVICE                       : in STROKE_DEVICE_
                                                _NUMBER;
    MODE                         : in OPERATION_MODE;
    SWITCH                       : in ECHO_SWITCH);
```

SET VALUATOR MODE — Уровень 0b
ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ЧИСЛА

```
  procedure SET_VALUATOR_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in VALUATOR_DEVICE_
                                                _NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);
```

SET CHOICE MODE — Уровень 0b
ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВЫБОРА

```
  procedure SET_CHOICE_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in CHOICE_DEVICE_
                                                _NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);
```

SET PICK MODE — Уровень 1b
ЗАДАТЬ РЕЖИМ УСТРОЙСТВА УКАЗАНИЯ

```
  procedure SET_PICK_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in PICK_DEVICE_NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);
```

SET STRING MODE — Уровень 0b
ЗАДАТЬ РЕЖИМ УСТРОЙСТВА ВВОДА СТРОКИ

```
  procedure SET_STRING_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in STRING_DEVICE_
                                                _NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);
```

REQUEST LOCATOR — Уровень 0b
ЗАПРОСИТЬ ВВОД ПОЗИЦИИ

```
  procedure REQUEST_LOCATOR
```

```
    (WS                          : in WS_ID;
    DEVICE                       : in LOCATOR_DEVICE_
                                        _NUMBER;
    STATUS                       : out INPUT_STATUS;
    TRANSFORMATION               : out TRANSFORMATION_
                                        _NUMBER;
    POSITION                     : out WC POINT);
```

REQUEST STROKE — Уровень 0b
ЗАПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИИ

```
  procedure REQUEST_STROKE
    (WS                          : in WS_ID;
    DEVICE                       : in STROKE_DEVICE_
                                        _NUMBER;
    STATUS                       : out INPUT_STATUS;
    TRANSFORMATION               : out TRANSFORMATION_
                                        _NUMBER;
    STROKE_POINTS                : out WC. POINT_LIST);
```

REQUEST VALUATOR — Уровень 0b
ЗАПРОСИТЬ ВВОД ЧИСЛА

```
  procedure REQUEST_VALUATOR
    (WS                          : in WS_ID;
    DEVICE                       : in VALUATOR_DEVICE_
                                        _NUMBER;
    STATUS                       : out INPUT_STATUS;
    VALUE                        : out VALUATOR_INPUT_
                                        _VALUE);
```

REQUEST CHOICE — Уровень 0b
ЗАПРОСИТЬ УСТРОЙСТВО ВЫБОРА

```
  procedure REQUEST_CHOICE
    (WS                          . in WS_ID;
    DEVICE                       : in CHOICE_DEVICE_
                                        _NUMBER;
    STATUS                       : out CHOICE_REQUEST_
                                        _STATUS;
    CHOICE_NUMBER                : out CHOICE_VALUE);
```

REQUEST PICK — Уровень 1b
ЗАПРОСИТЬ УКАЗАНИЯ

```
  procedure REQUEST_PICK
    (WS                          : in WS_ID;
```

```
    DEVICE                  : in PICK_DEVICE_NUMBER;
    STATUS                  : out PICK_REQUEST_STATUS;
    SEGMENT                 : out SEGMENT_NAME;
    PICK                    : out PICK_ID);
```

REQUEST STRING — Уровень 0b
ЗАПРОСИТЬ ВВОД СТРОКИ

```
  procedure REQUEST_STRING
    (WS                     : in WS_ID;
    DEVICE                  : in STRING_DEVICE_
                                        _NUMBER;
    STATUS                  : out INPUT_STATUS;
    CHAR_STRING             : out INPUT_STRING);
```

SAMPLE LOCATOR — Уровень 0c
ОПРОСИТЬ ВВОД ПОЗИЦИИ

```
  procedure SAMPLE_LOCATOR
    (WS                     : in WS_ID;
    DEVICE                  : in LOCATOR_DEVICE_
                                        _NUMBER;
    TRANSFORMATION          : out TRANSFORMATION_
                                        _NUMBER;
    POSITION                : out WC. POINT);
```

SAMPLE STROKE — Уровень 0c
ОПРОСИТЬ ВВОД ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ

```
  procedure SAMPLE_STROKE
    (WS                     : in WS_ID;
    DEVICE                  : in STROKE_DEVICE_
                                        _NUMBER;
    TRANSFORMATION          : out TRANSFORMATION_
                                        _NUMBER;
    STROKE_POINTS           : out WC. POINT_LIST);
```

SAMPLE VALUATOR — Уровень 0c
ОПРОСИТЬ ВВОД ЧИСЛА

```
  procedure SAMPLE_VALUATOR
    (WS                     : in WS_ID;
    DEVICE                  : in VALUATOR_DEVICE_
                                        _NUMBER;
    VALUE                   : out VALUATOR_INPUT_
                                        _VALUE);
```

SAMPLE CHOICE Уровень 0c
ОПРОСИТЬ ВЫБОР

```
procedure SAMPLE_CHOICE
  (WS                  : in WS_ID;
   DEVICE              : in CHOICE_DEVICE_
                                  _NUMBER;
   STATUS              : out CHOICE_STATUS;
   CHOICE_NUMBER       : out CHOICE_VALUE);
```

SAMPLE PICK Уровень 1c
ОПРОСИТЬ УКАЗАНИЕ

```
procedure SAMPLE_PICK
  (WS                  : in WS_ID;
   DEVICE              : in PICK_DEVICE_NUMBER;
   STATUS              : out PICK_STATUS;
   SEGMENT             : out SEGMENT_NAME;
   PICK                : out PICK_ID);
```

SAMPLE STRING Уровень 0c
ОПРОСИТЬ ВВОД СТРОКИ

```
procedure SAMPLE_STRING
  (WS                  : in WS_ID;
   DEVICE              : in STRING_DEVICE_
                                  _NUMBER;
   CHAR_STRING         : out INPUT_STRING);
```

AWAIT EVENT Уровень 0c
ОЖИДАТЬ СОБЫТИЕ

```
procedure AWAIT-EVENT
  (TIMEOUT             : in DURATION;
   WS                  : out WS_ID;
   CLASS               : out INPUT_CLASS;
   DEVICE              : out EVENT_DEVICE_
                                  _NUMBER;
```

FLUSH DEVICE EVENTS Уровень 0c
УДАЛИТЬ СОБЫТИЯ ОТ УСТРОЙСТВА

```
procedure FLUSH_DEVICE_EVENTS
  (WS                  : in WS_ID;
   CLASS               : in INPUT_QUEUE_CLASS;
   DEVICE              : in EVENT_OVERFLOW_
                                  _DEVICE_NUMBER);
```

GET LOCATOR Уровень 0с
ПОЛУЧИТЬ ПОЗИЦИЮ
```
procedure GET_LOCATOR
  (TRANSFORMATION      : out TRANSFORMATION_
                           _NUMBER;
   POSITION            : out WC. POINT);
```

GET STROKE Уровень 0с
ПОЛУЧИТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ПОЗИЦИЙ
```
procedure GET_STROKE
  (TRANSFORMATION      : out TRANSFORMATION_
                           _NUMBER;
   STROKE_POINTS       : out WC. POINT_LIST);
```

GET VALUATOR Уровень 0с
ПОЛУЧИТЬ ЧИСЛО
```
procedure GET_VALUATOR
  (VALUE               : out VALUATOR_INPUT_
                           _VALUE);
```

GET CHOICE Уровень 0с
ПОЛУЧИТЬ АЛЬТЕРНАТИВУ
```
procedure GET_CHOICE
  (STATUS              : out CHOICE_STATUS;
   CHOICE_NUMBER       : out CHOICE_VALUE);
```

GET PICK Уровень 1с
ПОЛУЧИТЬ УКАЗАТЕЛЬ
```
procedure GET_PICK
  (STATUS              : out PICK_STATUS;
   SEGMENT             : out SEGMENT_NAME;
   PICK                : out PICK_ID);
```

GET STRING Уровень 0с
ПОЛУЧИТЬ СТРОКУ
```
procedure GET_STRING
  (CHAR_STRING         : out INPUT_STRING);
```

WRITE ITEM TO GKSM Уровень 0а
ЗАПИСЬ В МЕТАФАЙЛ
```
procedure WRITE_ITEM_TO_GKSM
  (WS                  : in WS_ID;
   ITEM                : in GKSM_DATA_RECORD);
```

GET ITEM TYPE FROM GKSM — Уровень 0а
ПОЛУЧИТЬ ТИП ЗАПИСИ ИЗ ЯГС

```
procedure GET_ITEM_TYPE_FROM_GKSM
  (WS                    : in WS_ID;
  TYPE_OF ITEM           : out GKSM_ITEM_TYPE;
  LENGTH                 : out NATURAL);
```

READ ITEM FROM GKSM — Уровень 0а
ПРОЧИТАТЬ ЗАПИСЬ ИЗ ЯГС

```
procedure READ_ITEM_FROM_GKSM
  (WS                    : in WS_ID;
  MAX_LENGTH             : in NATURAL;
  ITEM                   : out GKSM_DATA_RECORD);
```

INTERPRET ITEM — Уровень 0а
ИНТЕРПРЕТИРОВАТЬ ЗАПИСЬ

```
procedure INTERPRET_ITEM
  (ITEM                  : in GKSM_DATA_RECORD);
```

INQUIRE OPERATING STATE VALUE — Уровень 0а
УЗНАТЬ ФУНКЦИОНАЛЬНОЕ СОСТОЯНИЕ

```
procedure INQ_OPERATING_STATE_VALUE
  (VALUE                 : out OPERATING_STATE);
```

INQUIRE LEVEL OF GKS — Уровень 0а
УЗНАТЬ УРОВЕНЬ ЯГС

```
procedure INQ_LEVEL_OF_GKS
  (ERROR_INDICATOR       : out ERROR_NUMBER;
  LEVEL                  : out GSK_LEVEL);
```

INQUIRE LIST OF AVAILABLE — Уровень 0а
WORKSTATION TYPES
УЗНАТЬ ДОСТУПНЫЕ ТИПЫ СТАНЦИЙ

```
procedure INQ_LIST_OF_AVAILABLE_WS_TYPES
  (ERROR_INDICATOR       : out ERROR_NUMBER;
  TYPES                  : out WS_TYPES. LIST_OF);
```

INQUIRE WORKSTATION MAXIMUM — Уровень 1а
NUMBERS
УЗНАТЬ ДОПУСТИМЫЕ КОЛИЧЕСТВА СТАНЦИЙ

```
procedure INQ_WS_MAX_NUMBERS
  (ERROR_INDICATOR       : out ERROR_NUMBER;
  MAX_OPEN_WS            : out POSITIVE;
```

```
    MAX_ACTIVE_WS              : out POSITIVE;
    MAX_SEGMENT_WS             : out POSITIVE;
```

INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER — Уровень 0а

УЗНАТЬ МАКСИМАЛЬНЫЙ НОМЕР ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ

```
  procedure INQ_MAX_NORMALIZATION_TRANSFORMATION_
                                          _NUMBER
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    TRANSFORMATION             : out TRANSFORMATION_
                                          _NUMBER);
```

INQUIRE SET OF OPEN WORKSTATIONS — Уровень 0а

УЗНАТЬ НАБОР ОТКРЫТЫХ СТАНЦИЙ

```
  procedure INQ_SET_OF_OPEN_WS
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    WS                         : out WS_IDS.LIST_OF);
```

INQUIRE SET OF ACTIVE WORKSTATIONS — Уровень 1а

УЗНАТЬ НАБОР АКТИВНЫХ СТАНЦИЙ

```
  procedure INQ_SET_OF_ACTIVE_WS
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    WS                         : out WS_IDS.LIST_OF);
```

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES — Уровень 0а

УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ АТРИБУТОВ ПРИМИТИВОВ

```
  procedure INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    ATTRIBUTES                 : out PRIMITIVE_ATTRIBUTE_
                                          _VALUES);

  procedure INQ_POLYLINE_INDEX
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    INDEX                      : out POLYLINE_INDEX);
  procedure INQ_POLYMARKER_INDEX
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    INDEX                      : out POLYMARKER_INDEX);
  procedure INQ_TEXT_INDEX
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    INDEX                      : out TEXT_INDEX);
```

```
procedure INQ_CHAR_HEIGHT
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  HEIGHT                    : out WC. MAGNITUDE);
procedure INQ_CHAR_UP_VECTOR
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  VECTOR                    : out WC. VECTOR);
procedure INQ_CHAR_WIDTH
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  WIDTH                     : out WC. MAGNITUDE);
procedure INQ_CHAR_BASE_VECTOR
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  VECTOR                    : out WC. VECTOR);
procedure INQ_TEXT_PATH
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  PATH                      : out TEXT_PATH);
procedure INQ_TEXT_ALIGNMENT
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  ALIGNMENT                 : out TEXT_ALIGNMENT);
procedure INQ_FILL_AREA_INDEX
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  INDEX                     : out FILL_AREA_INDEX);
procedure INQ_PATTERN_WIDTH_VECTOR
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  WIDTH                     : out WC. VECTOR);
procedure INQ_PATTERN_HEIGTH_VECTOR
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  VECTOR                    : out WC. VECTOR);
procedure INQ_PATTERN_REFERENCE_POINT
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  REFERENCE_POINT           : out WC. POINT);
```

INQUIRE CURRENT PICK IDENTIFIER VALUE — Уровень 1b

УЗНАТЬ ЗНАЧЕНИЕ ИДЕНТИФИКАТОРА УКАЗАНИЯ

```
procedure INQ_CURRENT_PICK_ID_VALUE
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  PICK                      : out PICK_ID);
```

INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES — Уровень 0a

УЗНАТЬ ТЕКУЩИЕ ЗНАЧЕНИЯ ИНДИВИДУАЛЬНЫХ АТРИБУТОВ

```
procedure INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_
                                                    _VALUES
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 ATTRIBUTES                : out INDIVIDUAL_ATTRIBUTE
                                                    _VALUES);
procedure INQ_LINETYPE
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 TYPE_OF_LINE              : out LINETYPE);
procedure INQ_LINEWIDTH_SCALE_FACTOR
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 WIDTH                     : out LINEWIDTH);
procedure INQ_POLYLINE_COLOUR_INDEX
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 LINE_COLOUR               : out COLOUR_INDEX);
procedure INQ_POLYMARKER_TYPE
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 TYPE_OF_MARKER            : out MARKER_TYPE);
procedure INQ_POLYMARKER_SIZE_SCALE_FACTOR
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 SIZE                      : out MARKER_SIZE);
procedure INQ_POLYMARKER_COLOUR_INDEX
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 MARKER_COLOUR             : out COLOUR_INDEX);
procedure INQ_TEXT_FONT_AND_PRECISION
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 FONT_PRECISION            : out TEXT_FONT_PRECISION);
procedure INQ_CHAR_EXPANSION_FACTOR
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 EXPANSION                 : out CHAR_EXPANSION);
procedure INQ_TEXT_COLOUR_INDEX
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 TEXT_COLOUR               : out COLOUR_INDEX);
procedure INQ_FILL_AREA_INTERIOR_STYLE
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 INTERIOR                  : out INTERIOR_STYLE);
procedure INQ_FILL_AREA_STYLE_INDEX
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 STYLE                     : out STYLE_INDEX);
procedure INQ_FILL_AREA_COLOUR_INDEX
 (ERROR_INDICATOR          : out ERROR_NUMBER;
 FILL_AREA_COLOUR          : out COLOUR_INDEX);
```

```
procedure INQ_LIST_OF_ASF
  (ERROR_INDICATOR        : out ERROR_NUMBER;
  LIST                    : out ASF_LIST);
```

INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER — Уровень 0a

УЗНАТЬ НОМЕР ТЕКУЩЕГО ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ

```
procedure INQ_CURRENT_NORMALIZATION_
                          _TRANSFORMATION_NUMBER
  (ERROR_INDICATOR        : out ERROR_NUMBER;
  TRANSFORMATION          : out TRANSFORMATION_
                                      _NUMBER);
```

INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS — Уровень 0a

УЗНАТЬ СПИСОК ПРЕОБРАЗОВАНИЙ НОРМИРОВАНИЯ

```
procedure INQ_LIST_OF_NORMALIZATION_
                          _TRANSFORMATION_NUMBERS
  (ERROR_INDICATOR        : out ERROR_NUMBER;
  LIST                    : out TRANSFORMATION_
                                  _PRIORITY_LIST);
```

INQUIRE NORMALIZATION TRANSFORMATION — Уровень 0a

УЗНАТЬ ПРЕОБРАЗОВАНИЯ НОРМИРОВАНИЯ

```
procedure INQ_NORMALIZATION_TRANSFORMATION
  (TRANSFORMATION         : in TRANSFORMATION_
                                      _NUMBER;
  ERROR_INDICATOR         : out ERROR_NUMBER;
  WINDOW_LIMITS           : out WC. RECTANGLE_LIMITS;
  VIEWPORT_LIMITS         : out NDC. RECTANGLE_
                                      _LIMITS);
```

INQUIRE CLIPPING — Уровень 0a

УЗНАТЬ ЗНАЧЕНИЯ ОТСЕЧЕНИЯ

```
procedure INQ_CLIPPING
  (ERROR_INDICATOR        : out ERROR_NUMBER;
  CLIPPING                : out CLIPPING_INDICATOR;
  CLIPPING_RECTANGLE      : out NDC. RECTANGLE_
                                      _LIMITS);
```

INQUIRE NAME OF OPEN SEGMENT — Уровень 1a

УЗНАТЬ ИМЯ ОТКРЫТОГО СЕГМЕНТА

```
procedure INQ_NAME_OF_OPEN_SEGMENT
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  SEGMENT                   : out SEGMENT_NAME);
```

INQUIRE SET OF SEGMENT NAMES IN USE — Уровень 1a

УЗНАТЬ ИМЕНА СУЩЕСТВУЮЩИХ СЕГМЕНТОВ

```
procedure INQ_SET_OF_SEGMENT_NAMES_IN_USE
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  SEGMENTS                  : out SEGMENT_NAMES.
                                        LIST_OF);
```

INQUIRE MORE SIMULTANEOUS EVENTS — Уровень 0c

УЗНАТЬ НАЛИЧИЕ ОДНОВРЕМЕННЫХ СОБЫТИЙ

```
procedure INQ_MORE_SIMULTANEOUS_EVENTS
  (ERROR_INDICATOR          : out ERROR_NUMBER;
  EVENTS                    : out MORE_EVENTS);
```

INQUIRE WORKSTATION CONNECTION AND TYPE — Уровень 0a

УЗНАТЬ ТИП И ИДЕНТИФИКАТОР СВЯЗИ СТАНЦИИ

```
procedure INQ_WS_CONNECTION_AND_TYPE
  (WS                       : in WS_ID;
  ERROR_INDICATOR           : out ERROR_NUMBER;
  CONNECTION                : out VARIABLE_CONNECTION
                                        _ID;
  TYPE_OF_WS                : out WS_TYPE);
```

INQUIRE WORKSTATION STATE — Уровень 0a

УЗНАТЬ СОСТОЯНИЕ СТАНЦИИ

```
procedure INQ_WS_STATE
  (WS                       : in WS_ID;
  ERROR_INDICATOR           : out ERROR_NUMBER;
  STATE                     : out WS_STATE);
```

INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES — Уровень 0a

УЗНАТЬ РЕЖИМЫ ЗАДЕРЖКИ И ОБНОВЛЕНИЯ СТАНЦИИ

```
procedure INQ_WS_DEFERRAL_AND_UPDATE_STATES
  (WS                       : in WS_ID;
  ERROR_INDICATOR           : out ERROR_NUMBER;
```

```
    DEFERRAL                : out DEFERRAL_MODE;
    REGENERATION            : out REGENERATION_MODE;
    DISPLAY                 : out DISPLAY_SURFACE_
                                        _EMPTY;
    FRAME_ACTION            : out NEW_FRAME
                                        _NECESSARY);
```

INQUIRE LIST OF POLYLINE INDICES — Уровень 1a
УЗНАТЬ ИНДЕКСЫ ЛОМАНОЙ

```
  procedure INQ_LIST_OF_POLYLINE_INDICES
    (WS                     : in WS_ID;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    INDICES                 : out POLYLINE_INDICES.
                                        LIST_OF);
```

INQUIRE POLYLINE REPRESENTATION — Уровень 1a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ

```
  procedure INQ_POLYLINE_REPRESENTATION
    (WS                     : in WS_ID;
    INDEX                   : in POLYLINE_INDEX;
    RETURNED_VALUES         : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    TYPE_OF_LINE            : out LINETYPE;
    WIDTH                   : out LINEWIDTH;
    LINE_COLOUR             : out COLOUR_INDEX);
```

INQUIRE LIST OF POLYMARKER INDICES — Уровень 1a
УЗНАТЬ ИНДЕКСЫ ПОЛИМАРКЕРА

```
  procedure INQ_LIST_OF_POLYMARKER_INDICES
    (WS                     : in WS_ID;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    INDICES                 : out POLYMARKER_INDICES.
                                        LIST_OF);
```

INQUIRE POLYMARKER REPRESENTATION — Уровень 1a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА

```
  procedure INQ_POLYMARKER_REPRESENTATION
    (WS                     : in WS_ID;
    INDEX                   : in POLYMARKER_INDEX;
    RETURNED_VALUES         : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR         : out ERROR_NUMBER;
```

```
    TYPE_OF_MARKER          : out MARKER_TYPE;
    SIZE                    . out MARKER_SIZE;
    MARKER_COLOUR           : out COLOUR_INDEX);
```

INQUIRE LIST OF TEXT INDICES — Уровень 1a
УЗНАТЬ ИНДЕКСЫ ТЕКСТА

```
  procedure INQ_LIST_OF_TEXT_INDICES
    (WS                     : in WS_ID;
    ERROR_INDICATOR         . out ERROR_NUMBER;
    INDICES                 : out TEXT_INDICES. LIST_OF);
```

INQUIRE TEXT REPRESENTATION — Уровень 1a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА

```
  procedure INQ_TEXT_REPRESENTATION
    (WS                     : in WS_ID;
    INDEX                   : in TEXT_INDEX;
    RETURNED_VALUES         : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    FONT_PRECISION          : out TEXT_FONT_PRECISION;
    EXPANSION               : out CHAR_EXPANSION;
    SPACING                 : out CHAR_SPACING;
    TEXT_COLOUR             : out COLOUR_INDEX);
```

INQUIRE TEXT EXTENT — Уровень 0a
УЗНАТЬ ГАБАРИТЫ ТЕКСТА

```
  procedure INQ_TEXT_EXTENT
    (WS                     : in WS_ID;
    POSITION                : in WC. POINT;
    CHAR_STRING             : in STRING;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    CONCATENATION_          : out WC. POINT;
      _POINT
    TEXT_EXTENT             : out TEXT_EXTENT_
                              _PARALLELOGRAM);
```

INQUIRE LIST OF FILL AREA INDICES — Уровень 1a
УЗНАТЬ ИНДЕКСЫ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ

```
  procedure INQ_LIST_OF_FILL_AREA_INDICES
    (WS                     : in WS_ID;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    INDICES                 : out FILL_AREA_INDICES.
                                        LIST_OF);
```

INQUIRE FILL AREA REPRESENTATION Уровень 1a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ

```
procedure INQ_FILL_AREA_REPRESENTATION
  (WS                    : in WS_ID;
  INDEX                  : in FILL_AREA_INDEX;
  RETURNED_VALUES        : in RETURN_VALUE_TYPE;
  ERROR_INDICATOR        : out ERROR_NUMBER;
  INTERIOR               : out INTERIOR_STYLE;
  STYLE                  : out STYLE_INDEX;
  FILL_AREA_COLOUR       : out COLOUR_INDEX);
```

INQUIRE LIST OF PATTERN INDICES Уровень 1a
УЗНАТЬ ИНДЕКСЫ ШАБЛОНА

```
procedure INQ_LIST_OF_PATTERN_INDICES
  (WS                    : in WS_ID;
  ERROR_INDICATOR        : out ERROR_NUMBER;
  INDICES                : out PATTERN_INDICES.
                                    LIST_OF);
```

INQUIRE PATTERN REPRESENTATION Уровень 1a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА

```
procedure INQ_PATTERN_REPRESENTATION
  (WS                    : in WS_ID;
  INDEX                  : in PATTERN_INDEX;
  RETURNED_VALUES        : in RETURN_VALUE_TYPE;
  ERROR_INDICATOR        : out ERROR_NUMBER;
  PATTERN                : out VARIABLE_COLOUR_
                                    _MATRIX);
```

INQUIRE LIST OF COLOUR INDICES Уровень 0a
УЗНАТЬ ИНДЕКСЫ ЦВЕТА

```
procedure INQ_LIST_OF_COLOUR_INDICES
  (WS                    : in WS_ID;
  ERROR_INDICATOR        : out ERROR_NUMBER;
  INDICES                : out COLOUR_INDICES.
                                    LIST_OF);
```

INQUIRE COLOUR REPRESENTATION Уровень 0a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА

```
procedure INQ_COLOUR_REPRESENTATION
  (WS                    : in WS_ID;
  INDEX                  : in COLOUR_INDEX;
```

```
    RETURNED_VALUES          : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR          : out ERROR_NUMBER;
    RGB_COLOUR               : out COLOUR_
                                   _REPRESENTATION);
```

INQUIRE WORKSTATION TRANSFORMATION — Уровень 0a

УЗНАТЬ ПРЕОБРАЗОВАНИЕ СТАНЦИИ

```
  procedure INQ_WS_TRANSFORMATION
    (WS                      : in WS_ID;
    ERROR_INDICATOR          : out ERROR_NUMBER;
    UPDATE                   : out UPDATE_STATE;
    REQUESTED_WINDOW         : out NDC. RECTANGLE_
                                   _LIMITS;
    CURRENT_WINDOW           : out NDC. RECTANGLE_
                                   _LIMITS;
    REQUESTED_VIEWPORT       : out DC. RECTANGLE_LIMITS;
    CURRENT_VIEWPORT         : out DC. RECTANGLE_
                                   _LIMITS);
```

INQUIRE SET OF SEGMENT NAMES ON WORKSTATION — Уровень 1a

УЗНАТЬ ИМЕНА СЕГМЕНТОВ, ХРАНИМЫХ НА СТАНЦИИ

```
  procedure INQ_SET_OF_SEGMENT_NAMES_ON_WS
    (WS                      : in WS_ID;
    ERROR_INDICATOR          : out ERROR_NUMBER;
    SEGMENTS                 : out SEGMENT_NAMES.
                                   LIST_OF);
```

INQUIRE LOCATOR DEVICE STATE — Уровень 0b

УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОЗИЦИИ

```
  procedure INQ_LOCATOR_DEVICE_STATE
    (WS                      : in WS_ID;
    DEVICE                   : in LOCATOR_DEVICE_
                                   _NUMBER;
    RETURNED_VALUES          : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR          : out ERROR_NUMBER;
    MODE                     : out OPERATING_MODE;
    SWITCH                   : out ECHO_SWITCH;
    INITIAL_TRANSFOR-
      MATION                 : out TRANSFORMATION_
                                   _NUMBER;
    INITIAL_POSITION         : out WC. POINT;
```

```
    ECHO_AREA                  : out DC.RECTANGLE_
                                       _LIMITS;
    DATA_RECORD                : out LOCATOR_DATA_
                                       _RECORD);
```

INQUIRE STROKE DEVICE STATE — Уровень 0b
УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ

```
  procedure INQ_STROKE_DEVICE_STATE
    (WS                        : in WS_ID;
    DEVICE                     : in STROKE_DEVICE_
                                       _NUMBER;
    RETURNED_VALUES            : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR            : out ERROR_NUMBER;
    MODE                       : out OPERATING_MODE;
    SWITCH                     : out ECHO_SWITCH;
    INITIAL_TRANSFOR-          : out TRANSFORMATION_
      MATION                           _NUMBER;
    INITIAL_STROKE_            : out WC. POINT_LIST;
      _POINTS
    ECHO_AREA                  : out DC.RECTANGLE_LIMITS;
    DATA_RECORD                : out STROKE_DATA_RECORD);
```

INQUIRE VALUATOR DEVICE STATE — Уровень 0b
УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА ЧИСЛА

```
  procedure INQ_VALUATOR_DEVICE_STATE
    (WS                        : in WS_ID;
    DEVICE                     : in VALUATOR_DEVICE_
                                       _NUMBER;
    ERROR_INDICATOR            : out ERROR_NUMBER;
    MODE                       : out OPERATING_MODE;
    SWITCH                     : out ECHO_SWITCH;
    INITIAL_VALUE              : out VALUATOR_INPUT_
                                       _VALUE;
    ECHO_AREA                  : out DC.RECTANGLE_LIMITS;
    DATA_RECORD                : out VALUATOR_DATA_
                                       _RECORD);
```

INQUIRE CHOICE DEVICE STATE — Уровень 0b
УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВЫБОРА

```
  procedure INQ_CHOICE_DEVICE_STATE
    (WS                        : in WS_ID;
    DEVICE                     : in CHOICE_DEVICE_
                                       _NUMBER;
```

```
   ERROR_INDICATOR          : out ERROR_NUMBER;
   MODE                     : out OPERATING_MODE;
   SWITCH                   : out ECHO_SWITCH;
   INITIAL_STATUS           : out CHOICE_STATUS;
   INITIAL_CHOICE           : out CHOICE_VALUE;
   ECHO_AREA                : out DC.RECTANGLE_
                                        _LIMITS;
   DATA_RECORD              : out CHOICE_DATA_RECORD);
```

INQUIRE PICK DEVICE STATE — Уровень 1b
УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА УКАЗАНИЯ

```
   procedure INQ_PICK_DEVICE_STATE
    (WS                     : in WS_ID;
   DEVICE                   : in PICK_DEVICE_NUMBER;
   RETURNED_VALUES          : in RETURN_VALUE_TYPE;
   ERROR_INDICATOR          : out ERROR_NUMBER;
   MODE                     : out OPERATING_MODE;
   SWITCH                   : out ECHO_SWITCH;
   INITIAL_STATUS           : out PICK_STATUS;
   INITIAL_SEGMENT          : out SEGMENT_NAME;
   INITIAL_PICK             : out PICK_ID;
   ECHO_AREA                : out DC.RECTANGLE_LIMITS;
   DATA_RECORD              : out PICK_DATA_RECORD);
```

INQUIRE STRING DEVICE STATE — Уровень 0b
УЗНАТЬ СОСТОЯНИЕ УСТРОЙСТВА ВВОДА СТРОКИ

```
   procedure INQ_STRING_DEVICE_STATE
    (WS                     : in WS_ID;
   DEVICE                   : in STRING_DEVICE_
                                        _NUMBER;
   ERROR_INDICATOR          : out ERROR_NUMBER;
   MODE                     : out OPERATING_MODE;
   SWITCH                   : out ECHO_SWITCH;
   INITIAL_STRING           : out INPUT_STRING;
   ECHO_AREA                : out DC.RECTANGLE_LIMITS;
   DATA_RECORD              : out STRING_DATA_RECORD);
```

INQUIRE WORKSTATION _CATEGORY — Уровень 0a
УЗНАТЬ КАТЕГОРИЮ СТАНЦИИ

```
   procedure INQ_WS_CATEGORY
    (TYPE_OF_WS             : in WS_TYPE;
   ERROR_INDICATOR          : out ERROR_NUMBER;
   CATEGORY                 : out WS_CATEGORY);
```

INQUIRE WORKSTATION _CLASSIFICATION — Уровень 0a

УЗНАТЬ КЛАСС СТАНЦИИ

```
procedure INQ_WS_CLASSIFICATION
  (TYPE_OF_WS          : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   CLASS               : out DISPLAY_CLASS);
```

INQUIRE DISPLAY SPACE SIZE — Уровень 0a

УЗНАТЬ РАЗМЕР НОСИТЕЛЯ ИЗОБРАЖЕНИЯ

```
procedure INQ_DISPLAY_SPACE_SIZE
  (TYPE_OF_WS          : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   UNITS               : out DC_UNITS;
   MAX_DC_SIZE         : out DC.SIZE;
   MAX_RASTER_UNIT_    : out RASTER_UNIT_SIZE);
     _SIZE
```

INQUIRE DYNAMIC MODIFICATION OF WORKSTATION_ATTRIBUTES — Уровень 1a

УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОГО ОБНОВЛЕНИЯ ХАРАКТЕРИСТИК ИЗОБРАЖЕНИЯ НА СТАНЦИИ

```
procedure INQ_DYNAMIC_MODIFICATION_OF_WS_
                                    _ATTRIBUTES
  (TYPE_OF_WS          : in WS_TYPE;
   ERROR_INDICATOR     : out ERROR_NUMBER;
   POLYLINE_           : out DYNAMIC_
     _REPRESENTATION                _MODIFICATION;
   POLYMARKER_         : out DYNAMIC_
     _REPRESENTATION                _MODIFICATION;
   TEXT_REPRESEN-      : out DYNAMIC_
     TATION                         _MODIFICATION;
   FILL_AREA_REPRESEN- : out DYNAMIC_
     TATION                         _MODIFICATION;
   PATTERN_REPRESEN-   : out DYNAMIC_
     TATION                         _MODIFICATION;
   COLOUR_REPRESEN-    : out DYNAMIC_
     TATION                         _MODIFICATION;
   TRANSFORMATION      : out DYNAMIC_
                                    _MODIFICATION);
```

INQUIRE DEFAULT DEFERRAL STATE VALUES — Уровень 1a
УЗНАТЬ РЕЖИМ ЗАДЕРЖКИ ПО УМОЛЧАНИЮ

```
procedure INQ_DEFAULT_DEFERRAL_STATE_VALUES
 (TYPE_OF_WS              : in WS_TYPE;
 ERROR_INDICATOR          . out ERROR_NUMBER;
 DEFERRAL                 : out DEFERRAL_MODE;
 REGENERATION             : out REGENERATION_MODE);
```

INQUIRE POLYLINE FACILITIES — Уровень 0a
УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЛОМАНОЙ

```
procedure INQ_POLYLINE_FACILITIES
 (TYPE_OF_WS              : in WS_TYPE;
 ERROR_INDICATOR          : out ERROR_NUMBER;
 LIST_OF_TYPES            : out LINETYPES.LIST_OF;
 NUMBER_OF_WIDTHS         : out NATURAL;
 NOMINAL_WIDTH            : out DC.MAGNITUDE;
 RANGE_OF_WIDTHS          : out DC.RANGE_OF_
                                  _MAGNITUDES;
 NUMBER_OF_INDICES        : out NATURAL);
```

INQUIRE PREDEFINED POLYLINE REPRESENTATION — Уровень 0a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ ПО УМОЛЧАНИЮ

```
procedure INQ_PREDEFINED_POLYLINE_REPRESENTATION
 (TYPE_OF_WS              : in WS_TYPE;
 INDEX                    : in POLYLINE_INDEX;
 ERROR_INDICATOR          : out ERROR_NUMBER;
 TYPE_OF_LINE             : out LINETYPE;
 WIDTH                    : out LINEWIDTH;
 LINE_COLOUR              : out COLOUR_INDEX);
```

INQUIRE POLYMARKER FACILITIES — Уровень 0a
УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ПОЛИМАРКЕРА

```
procedure INQ_POLYMARKER_FACILITIES
 (TYPE_OF_WS              : in WS_TYPE;
 ERROR_INDICATOR          : out ERROR_NUMBER;
 LIST_OF_TYPES            : out MARKER_TYPES.LIST_OF;
 NUMBER_OF_SIZES          : out NATURAL;
 NOMINAL_SIZE             : out DC.MAGNITUDE;
```

```
    RANGE_OF_SIZES          : out DC.RANGE_OF_
                                    _MAGNITUDES;
    NUMBER_OF_INDICES       : out NATURAL);
```

INQUIRE PREDEFINED POLYMARKER REPRESENTATION — Уровень 0а

УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА ПО УМОЛЧАНИЮ

```
  procedure INQ_PREDEFINED_POLYMARKER_
                                  _REPRESENTATION
    (TYPE_OF_WS             : in WS_TYPE;
    INDEX                   : in POLYMARKER_INDEX;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    TYPE_OF_MARKER          : out MARKER_TYPE;
    SIZE                    : out MARKER_SIZE;
    MARKER_COLOUR           : out COLOUR_INDEX),
```

INQUIRE TEXT FACILITIES — Уровень 0а

УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ТЕКСТА

```
  procedure INQ_TEXT_FACILITIES
    (TYPE_OF_WS             : in WS_TYPE;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    LIST_OF_FONT_           : out TEX_FONT_PRECISIONS.
      _PRECISION_PAIRS                    LIST_OF;
    NUMBER_OF_HEIGTS        : out NATURAL;
    RANGE_OF_HEIGTS         : out DC.RANGE_OF_
                                    _MAGNITUDES;
    NUMBER_OF_              : out NATURAL;
      _EXPANSIONS
    EXPANSION_RANGE         : out RANGE_OF_EXPANSIONS;
    NUMBER_OF_INDICES       : out NATURAL);
```

INQUIRE PREDEFINED TEXT REPRESENTATION — Уровень 0а

УЗНАТЬ ПРЕДСТАВЛЕНИЕ ТЕКСТА ПО УМОЛЧАНИЮ

```
  procedure INQ_PREDEFINED_TEXT_REPRESENTATION
    (TYPE_OF_WS             : in WS_TYPE;
    INDEX                   : in TEXT_INDEX;
    ERROR_INDICATOR         : out ERROR_NUMBER;
    FONT_PRECISION          : out TEXT_FONT_PRECISION;
    EXPANSION               : out CHAR_EXPANSION;
    SPACING                 : out CHAR_SPACING;
    TEXT_COLOUR             : out COLOUR_INDEX;
```

INQUIRE FILL AREA FACILITIES Уровень 0а
УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ

```
procedure INQ_FILL_AREA_FACILITIES
 (TYPE_OF_WS                : in WS_TYPE;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 LIST_OF_INTERIOR_          : out INTERIOR_STYLES.
  _STYLES                                    LIST_OF;
 LIST_OF_HATCH_             : out HATCH_STYLES.LIST_OF;
  _STYLES
 NUMBER_OF_INDICES          :  out NATURAL);
```

INQUIRE PREDEFINED FILL AREA REPRESENTATION Уровень 0а
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ ПО УМОЛЧАНИЮ

```
procedure INQ_PREDEFINED_FILL_AREA_
                                  _REPRESENTATION
 (TYPE_OF_WS                : in WS_TYPE;
 INDEX                      : in FILL_AREA_INDEX;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 INTERIOR                   : out INTERIOR_STYLE;
 STYLE                      : out STYLE_INDEX;
 FILL_AREA_COLOUR           : out COLOUR_INDEX);
```

INQUIRE PATTERN FACILITIES Уровень 0а
УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ШАБЛОНА

```
procedure INQ_PATTERN_FACILITIES
 (TYPE_OF_WS                : in WS_TYPE;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 NUMBER_OF_INDICES          : out NATURAL);
```

INQUIRE PREDEFINED PATTERN REPRESENTATION Уровень 0а
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ШАБЛОНА ПО УМОЛЧАНИЮ

```
procedure INQ_PREDEFINED_PATTERN_REPRESENTATION
 (TYPE_OF_WS                : in WS_TYPE;
 INDEX                      : in PATTERN_INDEX;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 PATTERN                    : out VARIABLE_COLOUR_
                                  _MATRIX);
```

INQUIRE COLOUR FACILITIES — Уровень 0a
УЗНАТЬ ВОЗМОЖНОСТИ ПРЕДСТАВЛЕНИЯ ЦВЕТА

```
procedure INQ_COLOUR_FACILITIES
  (TYPE_OF_WS              : in WS_TYPE;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  NUMBER_OF_COLOURS        : out NATURAL;
  AVAILABLE_COLOUR         : out COLOUR_AVAILABLE;
  NUMBER_OF_COLOUR_        : out NATURAL);
   _INDICES
```

INQUIRE PREDEFINED COLOUR REPRESENTATION — Уровень 0a
УЗНАТЬ ПРЕДСТАВЛЕНИЕ ЦВЕТА ПО УМОЛЧАНИЮ

```
procedure INQ_PREDEFINED_COLOUR_REPRESENTATION
  (TYPE_OF_WS              : in WS_TYPE;
  INDEX                    : in COLOUR_INDEX;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  RGB_COLOUR               : out COLOUR_REPRESEN-
                                          TATION);
```

INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES — Уровень 0a
УЗНАТЬ ИДЕНТИФИКАТОРЫ ДОСТУПНЫХ ОБОБЩЕННЫХ ПРИМИТИВОВ ВЫВОДА

```
procedure INQ_LIST_OF_AVAILABLE_GDP
  (TYPE_OF_WS              : in WS_TYPE;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  LIST_OF_GDP              : out GDP_IDS.LIST_OF);
```

INQUIRE GENERALIZED DRAWING PRIMITIVE — Уровень 0a
УЗНАТЬ ХАРАКТЕРИСТИКИ ОБОБЩЕННОГО ПРИМИТИВА ВЫВОДА

```
procedure INQ_GDP
  (TYPE_OF_WS              : in WS_TYPE;
  GDP                      : in GDP_ID;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  LIST_OF_ATTRIBUTES_      : out ATTRIBUTES_USED.
   _USED                                 LIST_OF);
```

INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES — Уровень 0a
УЗНАТЬ ДЛИНУ ТАБЛИЦ, ХАРАКТЕРИЗУЮЩИХ СТАНЦИЮ

```
procedure INQ_MAX_LENGTH_OF_WS_STATE_TABLES
 (TYPE_OF_WS                : in WS_TYPE;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 MAX_POLYLINE_              : out NATURAL;
  _ENTRIES
 MAX_POLYMARKER_            : out NATURAL;
  _ENTRIES
 MAX_TEXT_ENTRIES           : out NATURAL;
 MAX_FILL_AREA_             : out NATURAL;
 _ENTRIES
 MAX_PATTERN_               : out NATURAL;
  _INDICES
 MAX_COLOUR_INDICES         : out NATURAL);
```

INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED — Уровень 1a
УЗНАТЬ ДОПУСТИМОЕ ЧИСЛО ПРИОРИТЕТОВ СЕГМЕНТОВ

```
procedure INQ_NUMBER_OF_SEGMENT_PRIORITIES_
                                    _SUPPORTED
 (TYPE_OF_WS                : in WS_TYPE;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 NUMBER_OF_                 : out NATURAL);
  _PRIORITIES
```

INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES — Уровень 1a
УЗНАТЬ СПОСОБ ДИНАМИЧЕСКОЙ МОДИФИКАЦИИ АТРИБУТОВ СЕГМЕНТОВ

```
procedure INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_
                                    _ATTRIBUTES
 (TYPE_OF_WS                : in WS_TYPE;
 ERROR_INDICATOR            : out ERROR_NUMBER;
 TRANSFORMATION             : out DYNAMIC_
                                    _MODIFICATION;
 VISIBLE_TO_INVISIBLE       : out DYNAMIC_
                                    _MODIFICATION;
 INVISIBLE_TO_              : out DYNAMIC_
  _VISIBLE                          _MODIFICATION;
```

```
    HIGHLIGHTING              : out DYNAMIC_
                                    _MODIFICATION;
    PRIORITY                  : out DYNAMIC_
                                    _MODIFICATION;
    ADDING_PRIMITIVES         : out DYNAMIC_
                                    _MODIFICATION;
    DELETION_VISIBLE          : out DYNAMIC_
                                    _MODIFICATION);
```

INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES — Уровень 0b

УЗНАТЬ ЧИСЛО ДОПУСТИМЫХ УСТРОЙСТВ ВВОДА

```
    procedure INQ_NUMBER_OF_AVAILABLE_LOGICAL_
                                    _INPUT_DEVICES
    (TYPE_OF_WS               : in WS_TYPE;
    ERROR_INDICATOR           : out ERROR_NUMBER;
    LOCATOR                   : out NATURAL;
    STROKE                    : out NATURAL;
    VALUATOR                  : out NATURAL;
    CHOICE                    : out NATURAL;
    PICK                      : out NATURAL;
    STRING                    : out NATURAL);
```

INQUIRE DEFAULT LOCATOR DEVICE DATA — Уровень 0b

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ПОЗИЦИИ

```
    procedure INQ_DEFAULT_LOCATOR_DEVICE_DATA
    (TYPE_OF_WS               : in WS_TYPE;
    DEVICE                    : in LOCATOR_DEVICE_
                                    _NUMBER;
    ERROR_INDICATOR           : out ERROR_NUMBER;
    INITIAL_POSITION          : out WC.POINT;
    LIST_OF_PROMPT_           : out LOCATOR_PROMPT_
    _ECHO_TYPES                     _ECHO_TYPES.LIST_OF;
    ECHO_AREA                 : out DC.RECTANGLE_LIMITS;
    DATA_RECORD               : out LOCATOR_DATA_
                                    _RECORD);
```

INQUIRE DEFAULT STROKE DEVICE DATA — Уровень 0b

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ПОСЛЕДОВАТЕЛЬНОСТИ ПОЗИЦИЙ

```
procedure INQ_DEFAULT_STROKE_DEVICE_DATA
 (TYPE_OF_WS                 : in WS_TYPE;
 DEVICE                      : in STROKE_DEVICE_
                                  _NUMBER;
 ERROR_INDICATOR             : out ERROR_NUMBER;
 MAX_BUFFER_SIZE             : out NATURAL;
 LIST_OF_PROMPT_             : out STROKE_PROMPT_ECHO_
  _ECHO_TYPES                      _TYPES.LIST_OF;
 ECHO_AREA                   : out DC.RECTANGLE_LIMITS;
 DATA_RECORD                 : out STROKE_DATA_RECORD);
```

INQUIRE DEFAULT VALUATOR DEVICE DATA — Уровень 0b

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА ЧИСЛА

```
procedure INQ_DEFAULT_VALUATOR_DEVICE_DATA
 (TYPE_OF_WS                 : in WS_TYPE;
 DEVICE                      : in VALUATOR_DEVICE_
                                  _NUMBER;
 ERROR_INDICATOR             : out ERROR_NUMBER;
 INITIAL_VALUE               : out VALUATOR_INPUT_
                                  _VALUE;
 LIST_OF_PROMPT_             : out VALUATOR_PROMPT_
  _ECHO_TYPES                      _ECHO_TYPES.LIST_OF;
 ECHO_AREA                   : out DC.RECTANGLE_LIMITS;
 DATA_RECORD                 : out VALUATOR_DATA_
                                  _RECORD);
```

INQUIRE DEFAULT CHOICE DEVICE DATA — Уровень 0b

УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВЫБОРА

```
procedure INQ_DEFAULT_CHOICE_DEVICE_DATA
 (TYPE_OF_WS                 : in WS_TYPE;
 DEVICE                      : in CHOICE_DEVICE_
                                  _NUMBER;
 ERROR_INDICATOR             : out ERROR_NUMBER;
 MAX_CHOICES                 : out CHOICE_VALUE;
 LIST_OF_PROMPT_             : out CHOICE_PROMPT_ECHO_
  _ECHO_TYPES                      _TYPES.LIST_OF;
 ECHO_AREA                   : out DC.RECTANGLE_LIMITS;
 DATA_RECORD                 : out CHOICE_DATA_RECORD);
```

INQUIRE DEFAULT PICK DEVICE DATA Уровень 1b
УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА УКАЗАНИЯ

```
procedure INQ_DEFAULT_PICK_DEVICE_DATA
  (TYPE_OF_WS              : in WS_TYPE;
  DEVICE                   : in PICK_DEVICES_NUMBER;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  LIST_OF_PROMPT_          : out PICK_PROMPT_ECHO_
  _ECHO_TYPES                        _TYPES.LIST_OF;
  ECHO_AREA                : out DC.RECTANGLE_LIMITS;
  DATA_RECORD              : out PICK_DATA_RECORD);
```

INQUIRE DEFAULT STRING DEVICE DATA Уровень 0b
УЗНАТЬ ХАРАКТЕРИСТИКИ ПО УМОЛЧАНИЮ УСТРОЙСТВА ВВОДА СТРОКИ

```
procedure INQ_DEFAULT_STRING_DEVICE_DATA
  (TYPE_OF_WS              : in WS_TYPE;
  DEVICE                   : in STRING_DEVICE_
                                      _NUMBER;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  MAX_STRING_BUFFER_       : out NATURAL;
   _SIZE
  LIST_OF_PROMPT_          : out STRING_PROMPT_ECHO_
   _ECHO_TYPES                       _TYPES.LIST_OF;
  ECHO_AREA                : out DC.RECTANGLE_LIMITS;
  DATA_RECORD              : out STRING_DATA_RECORD);
```

INQUIRE SET OF ASSOCIATED WORKSTATIONS Уровень 1a
УЗНАТЬ СТАНЦИИ, СВЯЗАННЫЕ С СЕГМЕНТОМ

```
procedure INQ_SET_OF_ASSOCIATED_WS
  (SEGMENT                 : in SEGMENT_NAME;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  LIST_OF_WS               : out WS_IDS.LIST_OF);
```

INQUIRE SEGMENT ATTRIBUTES Уровень 1a
УЗНАТЬ АТРИБУТЫ СЕГМЕНТА

```
procedure INQ_SEGMENT_ATTRIBUTES
  (SEGMENT                 : in SEGMENT_NAME;
  ERROR_INDICATOR          : out ERROR_NUMBER;
  TRANSFORMATION           : out TRANSFORMATION_
                                      _MATRIX;
```

```
    VISIBILITY                  : out SEGMENT_VISIBILITY;
    HIGHLIGHTING                : out SEGMENT_
                                        _HIGHLIGHTING;
    PRIORITY                    : out SEGMENT_PRIORITY;
    DETECTABILITY               : out SEGMENT_
                                        _DETECTABILITY);
```

INQUIRE PIXEL ARRAY DIMENSIONS — Уровень 0a
УЗНАТЬ РАЗМЕРНОСТЬ МАТРИЦЫ ПИКСЕЛЕЙ

```
  procedure INQ_PIXEL_ARRAY_DIMENSIONS
    (WS                         : in WS_ID;
    CORNER_I_I                  : in WC.POINT;
    CORNER_DX_DY                : in WC.POINT;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    DIMENSIONS                  : out RASTER_UNIT_SIZE);
```

INQUIRE PIXEL ARRAY — Уровень 0a
УЗНАТЬ МАТРИЦУ ПИКСЕЛЕЙ

```
  procedure INQ_PIXEL_ARRAY
    (WS                         : in WS_ID;
    CORNER                      : in WC.POINT;
    DX                          : in RASTER_UNITS;
    DY                          : in RASTER_UNITS;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    INVALID_VALUES              : out INVALID_VALUES_
                                        _INDICATOR;
    PIXEL_ARRAY                 : out VARIABLE_PIXEL_
                                        _COLOUR_MATRIX);
```

INQUIRE_PIXEL — Уровень 0a
УЗНАТЬ ЦВЕТ ПИКСЕЛЯ

```
  procedure INQ_PIXEL
    (WS                         : in WS_ID;
    POINT                       : in WC.POINT;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    PIXEL_COLOUR                : out PIXEL_COLOUR_INDEX);
```

INQUIRE INPUT QUEUE OVERFLOW — Уровень 0c
УЗНАТЬ НАЛИЧИЕ ПЕРЕПОЛНЕНИЯ ОЧЕРЕДИ СОБЫТИЙ

```
  procedure INQ_INPUT_QUEUE_EVERFLOW
    (ERROR_INDICATOR            : out ERROR_NUMBER;
    WS                          : in WS_ID;
```

```
    CLASS                   : out INPUT_QUEUE_CLASS;
    DEVICE                  : out EVENT_OVERFLOW_
                                  _DEVICE_NUMBER);
```

EVALUATE TRANSFORMATION MATRIX — Уровень 1а
СФОРМИРОВАТЬ МАТРИЦУ ПРЕОБРАЗОВАНИЙ

```
  procedure EVALUATE_TRANSFORMATION_MATRIX
    (FIXED_POINT            : in WC.POINT;
    SHIFT_VECTOR            : in WC.VECTOR;
    ROTATION_ANGLE          : in RADIANS;
    SCALE_FACTORS           : in TRANSFORMATION_
                                  _FACTOR;
    TRANSFORMATION          : out TRANSFORMATION_
                                  _MATRIX);
  procedure EVALUATE_TRANSFORMATION_MATRIX
    (FIXED_POINT            : in NDC.POINT;
    SHIFT_VECTOR            : in NDC.VECTOR;
    ROTATION_ANGLE          : in RADIANS;
    SCALE_FACTORS           : in TRANSFORMATION_
                                  _FACTOR;
    TRANSFORMATION          : out TRANSFORMATION_
                                  _MATRIX);
```

ACCUMULATE TRANSFORMATION MATRIX — Уровень 1а
ВЫЧИСЛИТЬ РЕЗУЛЬТИРУЮЩУЮ МАТРИЦУ ПРЕОБРАЗОВАНИЙ

```
  procedure ACCUMULATE_TRANSFORMATION_MATRIX
    (SOURCE_TRANSFOR-       : in TRANSFORMATION_
     MATION                       _MATRIX);
    FIXED_POINT             : in WC.POINT;
    SHIFT_VECTOR            : in WC.VECTOR;
    ROTATION_ANGLE          : in RADIANS;
    SCALE_FACTORS           : in TRANSFORMATION_
                                  _FACTOR;
    RESULT_TRANSFOR-        : out TRANSFORMATION_
     MATION                       _MATRIX);
  procedure ACCUMULATE_TRANSFORMATION_MATRIX
    (SOURCE_TRANSFOR-       : in TRANSFORMATION_
     MATION                       _MATRIX);
    FIXED_POINT             : in NDC.POINT;
    SHIFT_VECTOR            : in NDC.VECTOR;
    ROTATION_ANGLE          : in RADIANS;
```

```
    SCALE_FACTORS              : in TRANSFORMATION_
                                    _FACTOR;
    RESULT_TRANSFOR-           : out TRANSFORMATION_
      MATION                        _MATRIX);
```

EMERGENCY CLOSE GKS; Уровень 0а
АВАРИЙНО ЗАКРЫТЬ ЯГС

```
  procedure EMERGENCY_CLOSE_GKS;
```

ERROR HANDLING Уровень 0а
ОБРАБОТАТЬ ОШИБКУ

```
  procedure ERROR_HANDLING
    (ERROR_INDICATOR           : in ERROR_NUMBER;
    GKS_FUNCTION               : in STRING;
    ERROR_FILE                 : in STRING := DEFAULT_
                                    _ERROR_FILE);
```

ERROR LOGGING Уровень 0а
ЗАРЕГИСТРИРОВАТЬ ОШИБКУ

```
  procedure ERROR_LOGGING
    (ERROR_INDICATOR           : out ERROR_NUMBER;
    GKS_FUNCTION               : in STRING;
    ERROR_FILE                 : in STRING := DEFAULT_
                                    _ERROR_FILE);
```

5.2. **Дополнительные функции**

5.2.1. Подпрограммы для манипуляции записями входных данных

В данном разделе определены функции и процедуры, которые необходимы для построения и запроса записей входных данных, декларированных как личные типы в данной связке, для всех шести классов устройств, определенных спецификацией ЯТС. Процедуры, представленные здесь, используют для построения записей данных для каждого зарегистрированного типа подсказки и эха. Также предоставляются соответствующие функции, позволяющие прикладным программам анализировать части записей данных, которые определены в ЯГС. Любую специфическую для реализации информацию в записях данных поддерживают личной и недоступной. Если любую из приведенных ниже процедур используют некорректно, то происходит исключительное событие GKS_ERROR. Таким образом, если недопустимый тип подсказки и эха используют для построения процедур, то в файле ошибок регистрируют ошибку номер 2500.

Данные подпрограммы требуются на уровне 0b.

Для создания зависящих от реализации и зарегистрированных элементов реализация может предоставить дополнительные совмещаемые версии процедур BUILD и дополнительные функции для выделения информации из личных записей данных.

```
— Операции над записями данных устройства ввода позиций
   procedure BUILD_LOCATOR_DATA_RECORD
    (PROMPT_ECHO_TYPE      : in LOCATOR_PROMPT_
                                    _ECHO_TYPE;
    DATA_RECORD            : out LOCATOR_DATA_
                                    _RECORD);
— Создает и возвращает записи данных устройства ввода позиций.
— Операции над записями данных устройства ввода последовательности позиций
   procedure BUILD_STROKE_DATA_RECORD
    (PROMPT_ECHO_TYPE      : in STROKE_PROMPT_ECHO_
                                    _TYPE;
    BUFFER_SIZE            : in POSITIVE;
    DATA_RECORD            : out STROKE_DATA_RECORD);
— Создает и возвращает записи данных устройства ввода последовательности позиций
```

```
   function BUFFER_SIZE (DATA_RECORD : in STROKE_DATA_
                                    _RECORD) return POSITIVE;
— Возвращает размер входного буфера устройства ввода последовательности позиций
— Операции над записями данных устройства ввода числа
   procedure BUILD_VALUATOR_DATA_RECORD
    (PROMPT_ECHO_TYPE      : in VALUATOR_PROMPT_
                                    _ECHO_TYPE;
    LOW_VALUE              : in VALUATOR_INPUT_
                                    _VALUE;
    HIGH_VALUE             : in VALUATOR_INPUT_
                                    _VALUE;
    DATA_RECORD            : out VALUATOR_DATA_
                                    _RECORD);
— Создает и возвращает записи данных устройства ввода числа.
   function HIGH_VALUE (DATA_RECORD : in VALUATOR_
                                    _DATA_RECORD)
      return VALUATOR_INPUT_VALUE;
```

— Возвращает наибольшее значение числа, запомненное в записи данных оценки.

```
function LOW_VALUE (DATA_RECORD : in VALUATOR_
                                    _DATA_RECORD)
    ruturn VALUATOR_INPUT_VALUE;
```

— Возвращает наименьшее значение числа, запомненное в записи данных оценки.

— Операции над записями данных устройства выбора

```
procedure BUILD_CHOICE_DATA_RECORD
 (PROMPT_ECHO_TYPE      : in CHOICE_PROMPT_ECHO_
                                    _TYPE;
 DATA_RECORD            : out CHOICE_DATA_RECORD);
```

— Создает и возвращает запись данных устройства выбора.

— Операции над записями данных устройства указания

```
procedure BUILD_PICK_DATA_RECORD
 (PROMPT_ECHO_TYPE      : in PICK_PROMPT_ECHO_
                                    _TYPE;
 DATA_RECORD            : out PICK_DATA_RECORD).
```

— Создает и возвращает запись данных устройства указания.

— Операции над записями данных устройства ввода строки

```
procedure BUILD_STRING_DATA_RECORD
 (PROMPT_ECHO_TYPE      : in STRING_PROMPT_ECHO_
                                    _TYPE;
 INPUT_BUFFER_SIZE      : in NATURAL;
 INITIAL_CURSOR_        : in NATURAL;
  _POSITION
 DATA_RECORD            : out STRING_DATA_RECORD);
```

— Создает и возвращает запись данных устройства ввода строки.

```
function INPUT_BUFFER_SIZE (DATA_RECORD:
                            in STRING_DATA_RECORD)
    return NATURAL;
```

— Возвращает размер буфера, используемого для запоминания вводимой строки, хранящейся в записи данных строки.

```
function INITIAL_CURSOR_POSITION (DATA_RECORD:
                            in STRING_DATA_RECORD)
    return NATURAL;
```

— Возвращает начальную позицию курсора для вводимой строки, запомненной в записи данных строки.

5.2.2. Пакет обобщенной координатной системы ЯГС

Обобщенный пакет, декларируемый в данном разделе, представляет собой спецификацию обобщенной декартовой системы

координат для GKS. Данный пакет встречается три раза в пакете GKS_TYPE для мировых координат, нормализованных координат устройства и координат устройства. Пакет определяет представление POINT (точки), POINT_ARRAY (матрицы точек), VECTOR (вектора) и RECTANGLE_LIMITS (прямоугольные ограничения) для координатной системы. Также определяется тип MAGNITUDE для измерения длин в координатной системе. Тип SIZE измеряет длины параллельно обеим осям, а тип RANGE_OF_MAGNITUDES определяет две длины внутри координатной системы: минимум и максимум для таких величин, как диапазон высот литер, доступных на устройстве. Данный обобщенный пакет включается в пакет типов GKS.

```
package
generic
 type COORDINATE_COMPONENT_TYPE is digits <>;
package GKS_COORDINAE_SYSTEM is
 type POINT is
   record
     X : COORDINATE_COMPONENT_TYPE;
     Y : COORDINATE_COMPONENT_TYPE;
   end record;
 type POINT_ARRAY is array (POSITIVE range <>)
     of POINT;
 type POINT_LIST (LENGTH : SMALL_NATURAL :=0) is
   record
     POINTS : POINT_ARRAY (1 .. LENGTH);
   end record;
type VECTOR is new POINT;
 type RECTANGLE_LIMITS is
   record
     XMIN : COORDINATE_COMPONENT_TYPE;
     XMAX : COORDINATE_COMPONENT_TYPE;
     YMIN : COORDINATE_COMPONENT_TYPE;
     YMAX : COORDINATE_COMPONENT_TYPE;
   end record;
 type MAGNITUDE_BASE_TYPE is digits PRECISION;
 subtype MAGNITUDE is MAGNITUDE_BASE_TYPE range
   COORDINATE_COMPONENT_TYPE'SAFE_SMALL ..
   COORDINATE_COMPONENT_TYPE'SAFE_LARGE;
 type SIZE is
 record
   XAXIS : MAGNITUDE;
   YAXIS : MAGNITUDE;
```

```
    end record;
    type RANGE_OF_MAGNITUDES is
    record
      MIN : MAGNITUDE;
      MAX : MAGNITUDE;
    end record;
  end GKS_COORDINATE_SYSTEM;
```

5.2.3. Общий пакет списка утилит ЯГС

Общий пакет GSK_LIST_UTILITIES встречается несколько раз в пакете GSK_TYPE для задания различных типов списков LIST_OF и подпрограмм для манипуляции ими. Каждый тип списка содержит различные значения типов элементов.

Тип списка декларируется как личный тип в GSK_LIST_UTILITIES для ограничения операций над типами списков, которые доступны внешним программным единицам. Декларация личных типов списков включает дискриминантную часть, которая определяет текущий размер списка. Объекты списков декларируются как неограниченные объекты, чтобы позволить динамическую модификацию размера списка.

Объект списка представляет собой последовательность значений типов элементов. Каждое значение типа элемента связывается с индексом. Значение индекса начинается с 1 и идет с приращением 1.

Размер объекта списка — это число значений типов элементов, запомненных в нем. Одно значение типа элемента может быть запомнено более одного раза внутри объекта списка. Объект списка может быть пустым. Максимальный размер объекта списка дан в общем параметре MAX_LIST_SIZE. Если данный параметр не задан, то используют принимаемое по умолчанию и зависящее от реализации значение.

— Подпрограммы обработки LIST_OF

```
function NULL_LIST return LIST_OF;
```

— Данная функция возвращает пустой объект LIST_OF. Данный список предназначен главным образом для тех, кто занимается реализацией GKS.

```
procedure ADD_TO_LIST
 (ELEMENT                : in ELEMENT_TYPE;
 LIST                    : in out LIST_OF);
```

— Данная процедура осуществляет запоминание значения параметра элемента в объекте параметра списка и увеличение размера списка на единицу. Индексное значение, равное увеличенному размеру списка, связывается с запомненным значением элемента.

Данная процедура порождает ошибку ЯГС 2502, если она вызывается, когда параметр списка имеет размер, равный максимальному. По желанию пользователю может гарантироваться незапоминание дублированных значений. Это реализуется вызовом ADD_TO_LIST с конкретным значением элемента, если функция IS_IN_LIST возвратила FALSE (ложь) для данного элемента.

```
procedure DELETE_FROM_LIST
(ELEMENT                    in ELEMENT_TYPE;
LIST                        : in out LIST_OF);
```

— Если объект параметра списка не содержит значения параметра элемента, данная процедура ничего не делает. В противном случае первый встретившийся элемент с данным значением удаляется. Размер объекта списка уменьшается на единицу, а индексы, связанные с оставшимися элементами, настраиваются таким образом, чтобы начинаться с единицы и идти с интервалом в единицу. При желании пользователь может удалить все элементы с данным значением. Это реализуется за счет повторения вызовов функции DELETE_FROM_LIST с конкретным значением элемента до тех пор, пока функция IS_IN_LIST возвращает TRUE (истина) для значения элемента.

```
function SIZE_OF_LIST (LIST : in LIST_OF)
    return NATURAL;
```

— Данная функция возвращает число значений типов элементов, запомненных в объекте списка.

```
function IS_IN_LIST
 (ELEMENT                   : in ELEMENT_TYPE;
 LIST                       : in LIST_OF) return BOOLEAN;
```

— Данная функция возвращает TRUE (истина), если значение параметра элемента существует в объекте списка; в противном случае возвращается FALSE.

```
function LIST_ELEMENT
 (INDEX                     : in POSITIVE;
 LIST                       : in LIST_OF) return ELEMENT
                                                 _TYPE;
```

— Данная функция возвращает значение элемента объекта списка, имеющего значение индекса, равное параметру индекса. Если параметр индекса превосходит текущий размер списка, генерируется ошибка ЯГС 2502.

```
function LIST (VALUES : in LIST_VALUES) return LIST_OF;
```

— Данная функция возвращает достоверный объект LIST_OF. Если параметр VALUE является пустой матрицей, то возвращается пустой объект LIST_OF. Если параметр не нулевой, данная

функция возвращает объект LIST_OF, содержащий все значения в параметре VALUES. Если число значений элементов превосходит максимальный размер объекта LIST_OF, генерируется ошибка ЯГС 2502.

— Спецификация обобщенного пакета

```
generic
   type ELEMENT_TYPE is private;
   MAX_LIST_SIZE : POSITIVE := implementation_defined;
package GKS_LIST_UTILITIES is
   subtype LIST_SIZE is NATURAL range 0..MAX_LIST_SIZE;
   type LIST_OF (SIZE : LIST_SIZE := 0) is private;
   type LIST_VALUES is array (POSITIVE range <>) of
                                        ELEMENT_TYPE;
function NULL_LIST return LIST_OF;
function SIZE_OF_LIST (LIST : in LIST_OF) return NATURAL;
function IS_IN_LIST          : in ELEMENT_TYPE;
    (ELEMENT
    LIST                     : in LIST_OF) return BOOLEN;
function LIST_ELEMENT        : in POSITIVE;
    (INDEX
    LIST                     : in LIST_OF) return
                                        ELEMENT_TYPE;
function LIST (VALUES        : in LIST VALUES) return
                                        LIST_OF;
procedure ADD_TO_LIST        : in ELEMENT_TYPE;
    (ELEMENT
    LIST                     : in out LIST_OF);
procedure DELETE_FROM_       : in ELEMENT_TYPE;
      LIST (ELEMENT
    LIST                     : in out LIST_OF);
private
```

— Декларация типа LIST_OF зависит от реализации. Однако операции, неявно объявленные декларацией LIST_OF, включая как присваивание, так и сравнение на равенство и неравенство, должны выполняться правильно. Данное требование предотвращает использование типов доступа для реализации типа LIST_OF. Рекомендуемая реализация представлена ниже:

```
   type LIST_OF (SIZE : LIST_SIZE := 0) is
    record
      ELEMENTS : LIST_VALUES (1..SIZE);
    end_record;
```

— Отметим, что декларирование неуточненных объектов LIST_OF с использованием значения дискриминанта, принимаемого по умолчанию, допускает динамическую модификацию размера матрицы элементов.

```
end GKS_LIST_UTILITIES;
```

5.2.4. Утилиты функций метафайла

Записи данных элементов метафайла являются сложными, для данных записей рекомендовано более 55 различных форматов. Прикладные программисты также могут определить новые форматы. Длина этих записей переменная. Записи данных могут содержать списки указателей, строки символов, матрицы индексов цветов и данные GDP и ESC. Длина записи зависит от числа элементов данных. GKS определяет, что формат зависит от реализации.

Тип записи данных должен быть личным, чтобы позволить непосредственную обработку содержимого записей.

Прикладной программист должен иметь возможность записать в метафайл нерафические данные. Это может быть предоставлено разрешением вывода символьных строк. Числовые данные могут быть преобразованы в символьные строки прикладным программистом до вызова функции BUILD_NEW_DATA_RECORD построить новую запись данных метафайла GKSM.

```
BUILD NEW GKSM DATA RECORD
procedure BUILD_NEW_GKSM_DATA_RECORD
    (TYPE_OF_ITEM                : in GKSM_ITEM_TYPE;
    ITEM_DATA                    : in STRING;
    ITEM                         : out GKSM_DATA_RECORD);
ITEM DATA RECORD STRING
function ITEM_DATA_RECORD_STRING
        (ITEM : in GKSM_DATA_RECORD) return STRING;
```

5.3. **Настраивающиеся варианты**

Так как подмножества или расширения Ады не допускаются, то нет настраивающихся вариантов ЯГС/Ада. Более того, данное связывание не требует, чтобы необходимые для поддержки свойства были независящими от реализации.

ПРИЛОЖЕНИЕ 4

СПЕЦИФИКАЦИЯ СКОМПИЛИРОВАННОГО ЯГС

(Данное приложение не является составной частью стандарта, а предоставляет дополнительную информацию)

```
with GKS_LIST_UTILITIES,
package GKS_TYPE is
```

— Данный пакет содержит все определения типов данных, используемые для задания связывания Ады с ЯГС. Данная компиляция была выполнена на компьютере MicroVax 2, используя компилятор VaxAda версии T1.4—32. Значения типов или подтипов, зависящих от реализации, были выбраны так, чтобы работать в среде 32-разрядного миникомпьютера с виртуальной памятью. Эти значения возможно потребуется изменить для микрокомпьютеров или машин с фиксированным размером памяти.

— Последующие константы являются зависящими от реализации и задают максимумы для реализации для типов ЯГС/Ада.

```
PRECISION                    : constant    : =6;
SMALL_NATURAL_MAX            : constant    : =500;
STRING_SMALL_NATURAL_
  _MAX                       : constant    : =100;
CHOICE_SMALL_NATURAL_
  _MAX                       . constant    : =5,

subtype SMALL_NATURAL is NATURAL
    range 0 . . SMALL_NATURAL_MAX;
```

— — Это зависящие от реализации подтипы, допускающие применение для объектов записей различных типов записей без возникновения прерывания STORAGE_ERROR.

```
subtype STRING_SMALL_NATURAL is NATURAL
    range 0. . STRING_SMALL_NATURAL_MAX;
```

— Это декларация зависящего от реализации подтипа, которая для объектов неуточненных записей допускает применение различных определенных ниже типов записей строк без возникновения прерывания STORAGE_ERROR.

```
subtype CHOICE_SMALL_NATURAL is NATURAL
    range 0. CHOICE_SMALL_NATURAL_MAX;
```

— Это декларация зависящего от реализации подтипа, который для объектов неуточненных записей допускает применение типа CHOICE_PROMPT_STRING_LIST без возникновения прерывания STORAGE_ERROR.

— Система координат ЯГС

```
generic
    type COORDINATE_COMPONENT_TYPE is digits <>;
package GKS_COORDINATE_SYSTEM is
    type POINT is
            record
                X : COORDINATE_COMPONENT_TYPE;
                Y : COORDINATE_COMPONENT_TYPE;
```

```
            end record;
    type POINT_ARRAY is array (POSITIVE range <>) of POINT;
    type POINT_LIST (LENGTH : SMALL_NATURAL := 0) is
            record
                POINTS : POINT_ARRAY (1..LENGTH);
            end record;
    type VECTOR is new POINT;
    type RECTANGLE_LIMITS is
            record
                XMIN : COORDINATE_COMPONENT_TYPE;
                XMAX : COORDINATE_COMPONENT_TYPE;
                YMIN : COORDINATE_COMPONENT_TYPE;
                YMAX : COORDINATE_COMPONENT_TYPE;
            end record;
    type MAGNITUDE_BASE_TYPE is digits PRECISION;
    subtype MAGNITUDE is MAGNITUDE_BASE_TYPE range
          COORDINATE_COMPONENT_TYPE'SAFE_SMALL..
          COORDINATE_COMPONENT_TYPE'SAFE_LARGE;
    type SIZE is
            record
                XAXIS : MAGNITUDE;
                YAXIS : MAGNITUDE;
            end record;
    type RANGE_OF_MAGNITUDES is
            record
                MIN : MAGNITUDE;
                MAX : MAGNITUDE;
            end record;
end GKS_COORDINATE_SYSTEM;
```

— ASF — Уровень 0а

```
type ASF is (BUNDLED, INDIVIDUAL);
```

— Данный тип определяет флаг выборки атрибутов, чье значение указывает, откуда будут устанавливаться атрибуты примитива: из таблицы связок или из индивидуального атрибута.

— ASF_LIST — Уровень 0а

```
type ASF_LIST is
      record
        TYPE_OF_LINE_ASF       : ASF;
        WIDTH_ASF              : ASF;
        LINE_COLOUR_ASF        : ASF;
        TYPE_OF_MARKER_ASF     : ASF;
        SIZE_ASF               : ASF;
        MARKER_COLOUR_ASF      : ASF;
        FONT_PRECISION_ASF     : ASF;
        EXPANSION_ASF          : ASF;
        SPACING_ASF            : ASF;
        TEXT_COLOUR_ASF        : ASF;
        INTERIOR_ASF           : ASF;
```

```
        STYLE_ASF                    : ASF;
        FILL_AREA_COLOUR_            : ASF,
          _ASF
      end record;
```

— Запись, содержащая все флаги выборки атрибутов, с компонентами, обозначающими индивидуальный флаг.

```
— ATTRIBUTES_USED_TYPE                                          Уровень 0a
type ATTRIBUTES_USED_TYPE is
      (POLYLINE_ATTRIBUTES,
       POLYMARKER_ATTRIBUTES,
       TEXT_ATTRIBUTES,
       FILL_AREA_ATTRIBUTES);
```

— Типы атрибутов, которые могут быть использованы при генерации вывода для ОПВ и при генерации подсказки и эха различных типов для различных классов входных устройств

```
— ATTRIBUTES_USED                                               Уровень 0a
package ATTRIBUTES_USED is
    new GKS_LIST_UTILITIES (ATTRIBUTES_USED_TYPE);
— Предназначен для списка используемых атрибутов
```

```
— SCALE_FACTOR                                                  Уровень 0a
package SCALE_FACTOR_TYPE is
— Данный пакет используют для выделения производного типа SCALE_FACTOR,
так как его используют как базу для ряда других производных типов. В языке
Ада, если база производного типа сама является производным типом. Этот тип
предка не может быть декларирован непосредственно в видимой части того же
самого пакета.
    type SCALE_FACTOR is digits PRECISION;
— Тип, используемый для безразмерных масштабов.
end SCALE_FACTOR_TYPE;
use SCALE_FACTOR_TYPE;
```

```
— CHAR_EXPANSION                                                Уровень 0a
type CHAR_EXPANSION is new SCALE_FACTOR range
    SCALE_FACTOR'SAFE_SMALL .. SCALE_FACTOR'LAST;
```

— Определяет масштаб расширения литер. Масштаб должен быть безразмерным и больше нуля.

```
— CHAR_SPACING                                                  Уровень 0a
type CHAR_SPACING is new SCALE_FACTOR;
```

— Определяет межлитерный просвет. Положительное значение указывает на величину дополнительного пространства между литерами в текстовой строке, а отрицательное — величину перекрытия прямоугольников литер в строке текста.

```
— DEVICE_NUMBER                                                 Уровень 0b
package DEVICE_NUMBER_TYPE is
    type DEVICE_NUMBER is new POSITIVE;
```

— Логическое устройство ввода устанавливают по номеру устройства.

```
end DEVICE_NUMBER_TYPE;
use DEVICE_NUMBER_TYPE;
```

```
— CHOICE_DEVICE_NUMBER                                          Уровень 0b
type CHOICE_DEVICE_NUMBER is new DEVICE_NUMBER;
```

— Определяет идентификатор устройства выбора.

```
— LOCATOR_DEVICE_NUMBER                                    Уровень 0b
type LOCATOR_DEVICE_NUMBER is new DEVICE_NUMBER;
— Предоставляет идентификатор устройства ввода позиции.

— PICK_DEVICE_NUMBER                                       Уровень 1b
type PICK_DEVICE_NUMBER is new DEVICE_NUMBER;
— Предоставляет идентификаторы устройства указания.

— STRING_DEVICE_NUMBER                                     Уровень 0b
type STRING_DEVICE_NUMBER is new DEVICE_NUMBER;
— Предоставляет номер устройства ввода строки.

— STROKE_DEVICE_NUMBER                                     Уровень 0b
type STROKE_DEVICE_NUMBER is new DEVICE_NUMBER;
— Предоставляет номер устройства ввода числа.

— VALUATOR_DEVICE_NUMBER                                   Уровень 0b
type VALUATOR_DEVISE_NUMBER is new DEVICE_NUMBER;
— Предоставляет идентификаторы устройства выбора.

— CHOICE_PROMPT                                            Уровень 0b
type CHOICE_PROMPT is (PFF, ON);
— Определяет, будет или нет отображаться заданная подсказка для типа под-
сказки и эха устройства выбора.

— CHOICE_PROMPTS                                           Уровень 0b
package CHOICE_PROMPTS is
   new GKS_LIST_UTILITIES (CHOICE_PROMPT);
— Предоставляет список подсказок.

— CHOICE_PROMPT_ECHO_TYPE                                  Уровень 0b
type CHOICE_PROMP_TECHO_TYPE is new INTEGER;
— Определяет тип подсказки и эха устройства выбора.

— CHOICE_PROMPT_ECHO_                                      Уровень 0b
  _TYPES
package CHOICE_PROMPT_ECHO_TYPES is
   new GKS_LIST_UTILITIES (CHOICE_PROMPT_ECHO_TYPE);
— Предоставляет список типов подсказок и эха устройства выбора.

— CHOICE_PROMPT_STRING                                     Уровень 0b
type CHOICE_PROMPT_STRING
   (LENGTH : STRING_SMALL_NATURAL := 0) is
   record
   CONTENTS : STRING (1..LENGTH);
   end record;
— Предоставляет подсказки переменной длины. Объекты данного типа должны
быть декларированы неуточненными, чтобы позволить динамическую модифика-
цию длины.

— CHOICE_PROMPT_STRING_                                    Уровень 0b
  _ARRAY
type CHOICE_PROMPT_STRING_ARRAY is array
   (POSITIVE range <>) of CHOICE_PROMPT_STRING;
— Предоставляет матрицу строк подсказок.

— CHOICE_PROMPT_STRING_                                    Уровень 0b
  _LIST
type CHOICE_PROMPT_STRING_LIST (LENGTH :
```

```
                                        CHOICE_SMALL_NATURAL := 0)
        is record
            LIST : CHOICE_PROMPT_STRING_ARRAY (1..LENGTH);
          end record;
— Предоставляет список строк подсказок.

— CHOICE_REQUEST_STATUS                                    Уровень 0b
type CHOICE_REQUEST_STATUS is (OK, NOCHOICE, NONE);
— Определяет статус для входных операций выбора для функций запроса.

— CHOICE_STATUS                                            Уровень 0b
subtype CHOICE_STATUS is CHOICE_REQUEST_STATUS range
                                                    OK..NOCHOICE;
— Указывает сделанный оператором выбор для функций опроса, получения события и справочных функций.

— CHOICE_VALUE                                             Уровень 0b
type CHOICE_VALUE is new POSITIVE;
— Определяет альтернативы, имеющиеся в реализации.

— CLIPPING_INDICATOR                                       Уровень 0a
type CLIPPING_INDICATOR is (CLIP, NOCLIP);
— Указывает на то, что будет или нет выполняться усечение.

— COLOUR_AVAILABLE                                         Уровень 0a
type COLOUR_AVAILABLE is (COLOUR, MONOCHROME);
— Указывает, имеется ли цветной вывод на станции.

— PIXEL_COLOUR_INDEX                                       Уровень 0a
type PIXEL_COLOUR_INDEX is new INTEGER
        range —1..INTEGER'LAST;
— Тип цвета пиксела, где —1 обозначает неверный индекс цвета.

— COLOUR_INDEX                                             Уровень 0a
subtype COLOUR_INDEX is PIXEL_COLOUR_INDEX
                range 0..PIXEL_COLOUR_INDEX'LAST;
— Предназначен для индексов в таблицах цветов.

— COLOUR_INDICES                                           Уровень 0a
package COLOUR_INDICES is new GKS_LIST_UTILITIES (COLOUR_INDEX);
— Предоставляет набор индексов цветов, которые имеются на конкретной станции.

— COLOUR_MATRIX                                            Уровень 0a
type COLOUR_MATRIX is array (POSITIVE range <>,
        POSITIVE range <>) of COLOUR_INDEX;
— Предоставляет матрицы, содержащие индексы цветов, соответствующие матрице ячеек или матрице шаблонов.

— INTENSITY                                                Уровень 0a
type INTENSITY is digits PRECISION range 0.0..1.0;
— Определяет область возможных интенсивностей цвета.

— COLOUR_REPRESENTATION                                    Уровень 0a
type COLOUR_REPRESENTATION is
        record
            RED         : INTENSITY;
            GREEN       : INTENSITY;
            BLUE        : INTENSITY;
```

```
    end record;
```
— Определяет представление цвета, как комбинацию интенсивностей в системе цветов красный—зеленый—голубой.

— CONTROL_FLAG Уровень 0a
type CONTROL_FLAG is (CONDITIONALLY, ALWAYS);
— Флаг управления используют, чтобы указать условия, при которых носитель изображения очищается

— DC_TYPE Уровень 0a
type DC_TYPE is digits PRECISION;
— Тип координат в системе координат устройства.

— DC Уровень 0a
package DC is new GKS_COORDINATE_SYSTEM (DC_TYPE);
— Определяет систему координат устройства.

— DC_UNITS Уровень 0a
type DC_UNITS is (METRES, OTHER);
— Единицей измерения координат устройства для конкретной станции должен быть метр, если устройство не способно порождать масштабированные образы, или зависящая от конкретной станции единица в противном случае.

— DEFFERAL_MODE Уровень 0a
type DEFFERAL_MODE is (ASAP, BNIG, BNIL, ASTI);
— Определяет четыре отложенных режима ЯГС

— DISPLAY_CLASS Уровень 0a
```
type DISPLAY_CLASS is (VECTOR_DISPLAY,
                       RASTER_DISPLAY,
                       OTHER_DISPLAY);
```
— Классификация станций категорий OUTPUT или OUTIN.

— DISPLAY_SURFACE_EMPTY Уровень 0a
type DISPLAY_SURFACE_EMPTY is (EMPTY, NOTEMPTY);
— Обозначает пуст ли носитель изображения.

— DYNAMIC_MODIFICATION Уровень 1a
type DYNAMIC_MODIFICATION is (IRG, IMM);
— Указывает, что обновление списка состояний выполняется немедленно или требует неявной повторной генерации.

— ECHO-SWITCH Уровень 0b
type ECHO_SWITCH is (ECHO, NOECHO);
— Обозначает, выполняется или нет выход эха.

— ERROR_NUMBER Уровень 0a
type is ERROR_NUMBER is new INTEGER;
— Определяет тип для значения индикатора ошибок

— INPUT_CLASS Уровень 0b
```
type INPUT_CLASS is (NONE,
                     LOCATOR_INPUT,
                     STROKE_INPUT,
                     VALUATOR_INPUT,
                     CHOICE_INPUT,
                     PICK_INPUT,
                     STRING_INPUT);
```
— Задает классификации входных устройств для станции категории IUPUT или OUTIN.

```
— EVENT_DEVICE_NUMBER                                          Уровень 0а
type EVENT_DEVICE_NUMBER (CLASS : INPUT_CLASS) : = NONE) is
    record
      case CLASS is
      when NONE                    => null;
      when LOCATOR_INPUT           => LOCATOR_EVENT_DEVICE
                                      : LOCATOR_DEVICE_NUMBER;
      when STROKE_INPUT            => STROKE_EVENT_DEVICE
                                      : STROKE_DEVICE_NUMBER;
      when VALUATOR_INPUT          => VALUATOR_EVENT_DEVICE
                                      : VALUATOR_DEVICE_
                                                      NUMBER;
      when CHOICE_INPUT            => CHOICE_EVENT_DEVICE
                                      : CHOICE_DEVICE_NUMBER;
      when PICK_INPUT              => PICK_EVENT_DEVICE
                                      : PICK_DEVICE_NUMBER
      when STRING_INPUT            => STRING_EVENT_DEVICE
                                      : STRING_DEVICE_NUMBER;
      end case;
    end record;
— Предназначен для возврата номера устройства любого класса из очереди
событий.
— INPUT_QUEUE_CLASS                                            Уровень 0а
subtype INPUT_QUEUE_CLASS is INPUT_CLASS range
        LOCATOR_INPUT . . STRING_INPUT;
— Определяет классификации устройств ввода для ситуаций, в которых класси-
фикация невозможна.
— EVENT_OVERFLOW_DEVICE_                                       Уровень 0а
_NUMBER
type EVENT_OVERFLOW_DEVICE_NUMBER
    (CLASS : INPUT_QUEUE_CLASS : = LOCATOR_INPUT) is
    record
      case CLASS is
    when LOCATOR_INPUT             => LOCATOR_EVENT_DEVICE
                                      : LOCATOR_DEVICE_NUMBER;
    when STROKE_INPUT              => STROKE_EVENT_DEVICE
                                      : STROKE_DEVICE_NUMBER;
    when VALUATOR_INPUT            => VALUATOR_EVENT_DEVICE
                                      : VALUATOR_DEVICE_
                                                      NUMBER;
    when CHOICE_INPUT              => CHOICE_EVENT_DEVICE
                                      : CHOICE_DEVICE_NUMBER;
    when PICK_INPUT                => PICK_EVENT_DEVICE
                                      : PICK_DEVICE_NUMBER;
    when STRING_INPUT              => STRING_EVENT_DEVICE
                                      : STRING_DEVICE_NUMBER;
      end case;
    end record;
— FILL_AREA_INDEX                                              Уровень 0а
type FILL_AREA_INDEX is new POSITIVE;
— Определяет индексы таблицы связок областей заполнения.
```

```
— INTERIOR_STYLE                                                    Уровень 0a
type INTERIOR_STYLE is (HOLLOW, SOLID, PATTERN, HATCH);
— Определяет вид заполнения области.

— STYLE_INDEX                                                       Уровень 0a
type STYLE_INDEX is new INTEGER;
— Индекс вида — это либо HATCH_STYLE, либо PATTERN_STYLE.

— FILL_AREA_INDICES                                                 Уровень 0a
package FILL_AREA_INDICES is
        new GKS_LIST_UTILITIES (FILL_AREA_INDEX);
— Предоставляет списки индексов таблицы связок областей заполнения.

— GDP_ID                                                            Уровень 0a
type GDP_ID is new INTEGER;
— Выбирает среди классов обобщенных примитивов вывода.

— GDP_IDS                                                           Уровень 0a
package GDP_IDS is new GKS_LIST_UTILITIES (GDP_ID);
— Предоставляет  списки идентификаторов обобщенных примитивов вывода.

— GKS_LEVEL                                                         Уровень 0a
type GKS_Level is (L0a, L0b, L0c, L1a, L1b, L1c, L2a, L2b, L2c);
— Доступные уровни ЯГС

— GKSM_ITEM_TYPE                                                    Уровень 0a
type GKSM_ITEM_TYPE is new NATURAL;
— Тип элемента, содержащегося в метафайле ЯГС

— HATCH_STYLE                                                       Уровень 0a
subtype HATCH_STYLE is STYLE_INDEX;
— Определяет вид штриховки при заполнении области.

— HATCH_STYLES                                                      Уровень 0a
package HATCH_STYLES is new GKS_LIST_UTILITIES (HATCH_STYLE);
— Предоставляет список видов штриховки.

— HORIZONTAL_ALIGNMENT                                              Уровень 0a
type HORIZONTAL_ALIGNMENT is (NORMAL, LEFT, CENTRE, RIGHT);
— Выравнивание параллелограмма текста по отношению к горизонтальному по-
ложению текста.

— IMPLEMENTATION_DEFINED_                                           Уровень 0a
  _ERROR
subtype IMPLEMENTATION_DEFINED_ERROR is ERROR_NUMBER
        range ERROR_NUMBER'FIRST.. —1;
— Определяет область номеров ошибок, чтобы указывать, что произошла за-
данная реализацией ошибка.

— INPUT_STATUS                                                      Уровень 0b
type INPUT_STATUS is (OK, NONE).
— Определяет статус операции.

— INPUT_STRING                                                      Уровень 0b
type INPUT_STRING (LENGTH : STRING_SMALL_NATURAL : = 0) is
    record
      CONTENTS : STRING (1. . LENGTH);
    end record;
```

— Предоставляет строку переменной длины. Объекты данного типа должны быть декларированы как неуточненные, чтобы позволить динамическую модификацию длины.

— INTERIOR_STYLES Уровень 0a
package INTERIOR_STYLES is
new GKS_LIST_UTILITIES (INTERIOR_STYLE);
— Предоставляет список видов заполнения.

— INVALID_VALUES_INDICATOR Уровень 0a
type INVALID_VALUES_INDICATOR is (ABSENT, PRESENT);
— Указывает, имеются ли —1 в параметре PIXEL_ARRAY, возвращаемом INQ_PIXEL_ARRAY.

— LANGUAGE_BILDING_ERROR Уровень 0a
subtype LANGUAGE_BILDING_ERROR is ERROR_NUMBER
range 2500. . 2999;
— Определяет область номеров ошибок, относящихся к ошибкам связывания с языком.

— POLYLINE_INDEX Уровень 0a
type POLYLINE_INDEX is new POSITIVE;
— Определяет область индексов ломаной.

— LINETYPE Уровень 0a
type LINETYPE is new INTEGER;
— Определяет типы линий, предоставляемых ЯГС.

— LINEWIDTH Уровень 0a
type LINEWIDTH is new SCALE_FACTOR range 0.0. .
SCALE_FACTOR'LAST;
— Ширина линии определяется масштабом толщины.

— LINETYPES Уровень 0a
package LINETYPES is new GKS_LIST_UTILITIES (LINETYPE).
— Предоставляет список типов линий.

— LOCATOR_PROMPT_ECHO_ Уровень 0b
_TYPE
type LOCATOR_PROMPT_ECHO_TYPE is new INTEGER;
— Определяет типы подсказок и эха, поддерживаемых реализацией.

— LOCATOR_PROMPT_ECHO_ Уровень 0b
_TYPES
package LOCATOR_PROMPT_ECHO_TYPES is
new GKS_LIST_UTILITIES (LOCATOR_PROMPT_ECHO_TYPE);
— Предоставляет список типов подсказок и эха определителя местоположения.

— POLYMARKER_INDEX Уровень 0a
type POLYMARKER_INDEX is new POSITIVE;
— Определяет область индексов таблицы связок полимаркеров.

— MARKER_SIZE Уровень 0a
type MARKER_SIZE is new SCALE_FACTOR range 0.0. .
SCALE_FACTOR'LAST;
— Размер маркера определяется масштабом.

— MARKER_TYPE Уровень 0b
type MARKER_TYPE is new INTEGER;
— Определяет типы маркеров, предоставляемых ЯГС.

```
— MARKER_DATA                                                    Уровень 0b
— MARKER_TYPES                                                   Уровень 0a
package MARKER_TYPES is new GKS_LIST_UTILITIES
                (MARKER_TYPE);
— Предоставляет список типов маркеров.

— MORE_EVENTS                                                    Уровень 0a
type MORE_EVENTS is (NOMORE, MORE);
— Указывает, содержатся ли еще события в очереди событий.

— NDC_TYPE                                                       Уровень ma
type NDC_TYPE is digits PRECISION;
— Определяет тип координат в нормализованной системе координат.

— NDC                                                            Уровень ma
package NDC is new GKS_COORDINATE_SYSTEM (NDC_TYPE);
   Задает нормализованную систему координат.

— NEW_FRAME_NECESSARY                                            Уровень 0a
type NEW_FRAME_NECESSARY is (NO, YES);
— Указывает, необходимы ли действия по новому коду при модификации.

— OPERATING_MODE                                                 Уровень 0b
type OPERATING_MODE is (REQUEST_MODE, SAMPLE_MODE,
                        EVENT_MODE);
— Определяет режимы работы устройства ввода.

— OPERATING_STATE                                                Уровень 0a
type OPERATING_STATE is (GKCL, GKOP, WSOP, WSAC, SGOP);
— Определяет пять состояний ЯГС.

— PATTERN_INDEX                                                  Уровень 0a
subtype PATTERN_INDEX is STYLE_INDEX range 1.. STYLE_INDEX'LAST;
— Определяет диапазон индексов таблицы шаблонов.

— PATTERN_INDICES                                                Уровень 0a
package PATTERN_INDICES is
        new GKS_LIST_UTILITIES (PATTERN_INDEX);
— Предоставляет списки индексов таблицы шаблонов.

— PICK_ID                                                        Уровень 1b
type PICK_ID is new POSITIVE;
— Определяет диапазон идентификаторов устройства указания, существующих
в реализации.

— PICK_IDS                                                       Уровень 1b
pakcage PICK_IDS is new GKS_LIST_UTILITIES (PICK_ID);
— Предоставляет списки идентификаторов устройства указания.

— PICK_PROMPT_ECHO_TYPE                                          Уровень 1b
type PICK_PROMPT_ECHO_TYPE is new INTEGER;
— Определяет тип подсказки и эха для устройства указания.

— PICK_PROMPT_ECHO_TYPES                                         Уровень 1b
package PICK_PROMPT_ECHO_TYPES is new GKS_LIST_UTILITIES
        (PICK_PROMPT_ECHO_TYPE);
   — Предоставляет списки типов подсказки и эха устройства указания.
```

— PICK_REQUEST_STATUS Уровень 1b
type PICK_REQUEST_STATUS is (OK, NOPICK, NONE);
— Определяет статус операции ввода указания для функции запроса.

— PICK_STATUS Уровень 1b
subtype PICK_STATUS is PICK_REQUEST_STATUS range OK..NOPICK;
— Определяет статус операции ввода указания для функций получения информации.

— PIXEL_COLOUR_MATRIX Уровень 0a
type PIXEL_COLOUR_MATRIX is array (POSITIVE range <>,
POSITIVE range <>) of PIXEL_COLOUR_INDEX;
— Предоставляет матрицы цветов пикселов.

— POLYLINE_INDICES Уровень 0a
package POLYLINE_INDICES is new GKS_LIST_UTILITIES
(POLYLINE_INDEX);
— Предоставляет списки индексов ломаной

— POLYMARKER_INDICES Уровень 0a
package POLYMARKER_INDICES is new GKS_LIST_UTILITIES
(POLYMARKER_INDEX);
— Предоставляет списки индексов полимаркеров.

— RADIANS Уровень 1a
type RADIANS is digits PRECISION;
— Величины, используемые в выполнении преобразования сегмента (угол вращения). Положительное значение указывает на вращение против часовой стрелки.

— RANGE_OF_EXPANSIONS Уровень 0a
type RANGE_OF_EXPANSIONS is
record
MIN : CHAR_EXPANSION;
MAX : CHAR_EXPANSION;
end record;
— Предоставляет область значений масштаба расширения литер.

— RASTER_UNITS Уровень 0a
type RASTER_UNITS is new POSITIVE;
— Определяет область единиц растра.

— RASTER_UNIT_SIZE Уровень 0a
type RASTER_UNIT_SIZE is
record
X : RASTER_UNITS;
Y : RASTER_UNITS;
end record;
— Определяет размер экрана дисплея в растровых единицах.

— REGENERATION_MODE Уровень 0a
type REGENERATION_MODE is (SUPPRESSED, ALLOWED);
— Указывает, подавлена или разрешена неявная повторная генерация.

— RELATIVE_PRIORITY Уровень 0a
type RELATIVE_PRIORITY is (HIGHER, LOWER);
— Обозначает относительный приоритет между двумя преобразованиями нормирования.

— RETURN_VALUE_TYPE Уровень 0a
type RETURN_VALUE_TYPE is (SET, REALIZED);
— Указывает, возвращаемое значение следует рассматривать как заданное программой или как действительно реализованное устройством.

— SEGMENT_DETECTABILITY Уровень 1a
type SEGMENT_DETECTABILITY is (UNDETECTABLE, DETECTABLE);
— Указывает, является или нет сегмент обнаруживаемым.

— SEGMENT_HIGHLIGHTING Уровень 1a
type SEGMENT_HIGHLIGHTING is (NORMAL, HIGHLIGHTED);
— Указывает, является или нет сегмент выделяемым.

— SEGMENT_NAME Уровень 1a
type SEGMENT_NAME is new POSITIVE;
— Определяет диапазон имен сегментов.

— SEGMENT_NAMES Уровень 1a
package SEGMENT_NAMES is new GKS_LIST_UTILITIES
(SEGMENT_NAME);
— Дает список имен сегментов.

— SEGMENT_PRIORITY Уровень 1a
type SEGMENT_PRIORITY is digits PRECISION range 0.0..0.1;
— Определяет приоритет сегмента.

— SEGMENT_VISIBILITY Уровень 1a
type SEGMENT_VISIBILITY is (VISIBLE, INVISIBLE);
— Обозначает, является ли сегмент видимым или нет.

— STRING_PROMPT_ECHO_TYPE Уровень 0b
type STRING_PROMPT_ECHO_TYPE is new INTEGER;
— Определяет типы подсказки и эха устройства ввода строки.

— STRING_PROMPT_ECHO_TYPES Уровень 0b
package STRING_PROMPT_ECHO_TYPES is
new GKS_LIST_UTILITIES (STRING_PROMPT_ECHO_TYPE);
— Предоставляет списки типов подсказок и эха устройства ввода строки.

— STROKE_PROMPT_ECHO_TYPE Уровень 0b
type STROKE_PROMPT_ECHO_TYPE is new INTEGER;
— Определяет типы подсказок и эха устройства ввода последовательности позиций.

— STROKE_PROMPT_ECHO_ Уровень 0b
TYPES
package STROKE_PROMPT_ECHO_TYPES is
new GKS_LIST_UTILITIES (STROKE_PROMPT_ECHO_TYPE);
— Предоставляет списки типов подсказок и эха ввода последовательности позиций.

— VERTICAL_ALIGNMENT Уровень 0a
type VERTICAL-ALIGNMENT is (NORMAL, TOP, CAP, HALF, BASE,
BOTTOM);
— Выравнивание параллелограмма текста по отношению к вертикальной позиции текста.

— TEXT_ALIGNMENT Уровень 0a
type TEXT_ALIGNMENT is
record

```
    HORIZONTAL                  : HORIZONTAL_ALIGNMENT;
    VERTICAL                    : VERTICAL_ALIGNMENT;
      end record;
```

— Тип атрибута, управляющего позиционированием параллелограмма текста по отношению к позиции текста, имеющего горизонтальные и вертикальные компоненты, как определено выше.

```
— WC_TYPE                                                    Уровень 0a
type WC_TYPE is digits PRECISION;
```

— Определяет точность для типа мировых координат.

```
— WC                                                         Уровень 0a
package WC is new GKS_COORDINATE_SYSTEM (WC_TYPE);
```

— Определяет мировую систему координат.

```
— TEXT_EXTENT                                                Уровень 0a
  _PARALLELOGRAM
type TEXT_EXTENT_PARALLELOGRAM is
      record
    LOWER_LEFT                  : WC.POINT;
    LOWER_RIGHT                 : WC.POINT;
    UPPER_RIGHT                 : WC.POINT;
    UPPER_LEFT                  : WC.POINT;
      end record;
```

— Определяет угловые точки параллелограмма текста по отношению к вертикальному позиционированию текста.

```
— TEXT_FONT                                                  Уровень 0a
type TEXT_FONT is new INTEGER;
```

— Определяет типы шрифтов, предоставляемых реализацией.

```
— TEXT_PRECISION                                             Уровень 0a
type TEXT_PRECISION is (STRING_PRECISION,
                        CHAR_PRECISION,
                        STROKE_PRECISION);
```

— Точность, с которой появляется текст.

```
— TEXT_FONT_PRECISION                                        Уровень 0a
type TEXT_FONT_PRECISION is
      record
    FONT                        : TEXT_FONT;
    PRECISION                   : TEXT_PRECISION;
      end record;
```

— Данный тип определяет запись, описывающую атрибут шрифта и точности текста.

```
— TEXT_FONT_PRECISIONS                                       Уровень 0a
package TEXT_FONT_PRECISIONS is
          new GKS_LIST_UTILITIES (TEXT_FONT_PRECISION);
```

— Предоставляет список пар шрифта и точности текста.

```
— TEXT_INDEX                                                 Уровень 0a
type TEXT_INDEX is new POSITIVE;
```

— Определяет диапазон значений индексов таблицы связок текста

```
— TEXT_INDICES                                               Уровень 0a
package TEXT_INDICES is new GKS_LIST_UTILITIES (TEXT_INDEX);
```

— Предоставляет списки индексов текста.

```
— TEXT_PATH                                                    Уровень 0a
type TEXT_PATH is (RIGHT, LEFT, UP, DOWN);
— Направление строки текста.

— TRANSFORMATION_FACTOR                                        Уровень 1a
type TRANSFORMATION_FACTOR is
        record
          X : NDC_TYPE;
          Y : NDC_TYPE;
        end record;
— Масштаб, используемый в матрицах преобразования для выполнения преобразования сегментов.

— TRANSFORMATION_MATRIX                                        Уровень 1a
type TRANSFORMATION_MATRIX is array (1..2, 1..3) of NDC_TYPE;
— Для преобразований сегментов, отображающихся внутрь пространства НК.

— TRANSFORMATION_NUMBER                                        Уровень 0a
type TRANSFORMATION_NUMBER is new NATURAL;
— Номер преобразования нормирования.
subtype POSITIVE_TRANSFORMATION_NUMBER is
        TRANSFORMATION_NUMBER
        range 1.. TRANSFORMATION_NUMBER'LAST;
— Номер преобразования нормирования, соответствующий установленному преобразованию.

— TRANSFORMATION_PRIORITY                                      Уровень 0a
_ARRAY
type TRANSFORMATION_PRIORITY_ARRAY is array
    (POSITIVE range <>) of TRANSFORMATION_NUMBER;
— Тип для запоминания номера преобразования.

— TRANSFORMATION_PRIORITY_                                     Уровень 0a
LIST
type TRANSFORMATION_PRIORITY_LIST (LENGTH : SMALL_NATURAL
    := 0) is
        record
        CONTENTS : TRANSFORMATION_PRIORITY_ARRAY (1..LENGTH);
        end record;
— Предоставляет упорядоченный по приоритетам список номеров преобразований.

— UPDATE_REGENERATION_                                         Уровень 0a
FLAG
type UPDATE_REGENERATION_FLAG is (PERFORM, POSTRONE);
— Флаг повторной генерации на дисплее.

— UPDATE_STATE                                                 Уровень 0a
type UPDATE_STATE is (NOTPENDING, PENDING);
— Указывает, было ли запрошено изменение преобразования станции.

— VALUATOR_INPUT_VALUE                                         Уровень 0b
type VALUATOR_INPUT_VALUE is digits PRECISION;
— Определяет точность вводимых величин в данной реализации.

— VALUATOR_PROMPT_ECHO_                                        Уровень 0b
TYPE
type VALUATOR_PROMPT_ECHO_TYPE is new INTEGER;
— Определяет возможные типы подсказок и эха устройства ввода числа.
```

— VALUATOR_PROMPT_ECHO_TYPES Уровень 0b
package VALUATOR_PROMPT_ECHO_TYPES is
new GKS_LIST_UTILITIES (VALUATOR_PROMPT_ECHO_TYPE);
— Предоставляет списки типов подсказок и эха устройства ввода числа.

— VARIABLE_COLOUR_MATRIX Уровень 0a
type VARIABLE_COLOUR_MATRIX (DX : SMALL_NATURAL : = 0;
DY : SMALL_NATURAL : = 0) is
record
MATRIX : COLOUR_MATRIX (1..DX, 1..DY);
end record;
— Предоставляет матрицы переменного размера, содержащие индексы цветов, соответствующие матрице ячеек или матрице шаблонов.

— VARIABLE_CONNECTION_ID Уровень 0a
type VARIABLE_CONNECTION_ID
(LENGTH : STRING_SMALL_NATURAL : = 0) is
record
CONNECT : STRING (1..LENGTH);
end record;
— Определяет идентификатор связи переменной длины для INQ_WS_CONNECTION_AND_TYPE.

— VARIABLE_PIXEL_COLOUR_MATRIX Уровень 0a
type VARIABLE_PIXEL_COLOUR_MATRIX (DX : SMALL_NATURAL : = 0;
DY : SMALL_NATURAL : = 0) is
record
MATRIX : PIXEL_COLOUR_MATRIX (1..DX, 1..DY);
end record;
— Предоставляет матрицы переменного размера для цветов пикселей.

— WS_CATEGORY Уровень 0a
type WS_CATEGORY is (OUTPUT, INPUT, OUTIN, WISS, MO, MI);
— Тип для категорий станций ЯГС.

— WS_ID Уровень 0a
type WS_ID in new POSITIVE;
— Определяет область идентификаторов станций.

— WS_IDS Уровень 0a
package WS_IDS is new GKS_LIST_UTILITIES (WS_ID);
— Предоставляет списки идентификаторов станций.

— WS_STATES Уровень 0a
type WS_STATES is (INACTIVE, ACTIVE);
— Состояние станции.

— WS_TYPES Уровень 0a
type WS_TYPES is new POSITIVE;
— Область значений, соответствующих достоверным типам станций. Константы, определяющие имена для различных типов станций, должны предоставляться реализацией.

— WS_TYPES Уровень 0a
package WS_TYPES is new GKS_LIST_UTILITIES (WS_TYPE);
— Предоставляет списки типов станций.

— INDIVIDUAL_ATTRIBUTE_ Уровень 0а
VALUES
type INDIVIDUAL_ATTRIBUTE_VALUES is
record
TYPE_OF_LINE : LINETYPE;
WIDTH : LINEWIDTH;
LINE_COLOUR : COLOUR_INDEX;
TYPE_OF_MARKER : MARKER_TYPE;
SIZE : MARKER_SIZE;
MARKER_COLOUR : COLOUR_INDEX;
FONT_PRECISION : TEXT_FONT_PRECISION;
EXPANSION : CHAR_EXPANSION;
SPACING : CHAR_SPACING;
TEXT_COLOUR : COLOUR_INDEX;
INTERIOR : INTERIOR_STYLE;
STYLE : STYLE_INDEX;
FILL_AREA_COLOUR : COLOUR_INDEX;
ASF : ASF_LIST;
end record;
— Запись, содержащая текущие индивидуальные атрибуты для процедуры INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES.

— PRIMITIVE_ATTRIBUTE_ Уровень 0а
_VALUES
type PRIMITIVE_ATTRIBUTE_VALUES is
record
INDEX_POLYLINE : POLYLINE_INDEX;
INDEX_POLYMARKER : POLYMARKER_INDEX;
INDEX_TEXT : TEXT_INDEX;
CHAR_HEIGHT : WC.MAGNITUDE;
CHAR_UP_VECTOR : WC.VECTOR;
CHAR_WIDTH : WC.MAGNITUDE;
CHAR_BASE_VECTOR : WC.VECTOR;
PATH : TEXT_PATH;
ALIGNMENT : TEXT_ALIGNMENT;
INDEX_FILL_AREA FILL_AREA_INDEX;
PATTERN_WIDTH_VECTOR : WC.VECTOR;
PATTERN_HEIGHT_VECTOR : WC.VECTOR;
PATTERN_REFERENCE_ : WC.POINT;
POINT
end record;
— Запись, содержащая атрибуты текущего примитива для процедуры INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES.
— Далее задается предопределенное прерывание GKS-ERROR, заданное в п. 3.2.3.
GKS_ERROR : exception;
— Далее идут декларации зависящих от реализации констант для определения типов ЯГС/Ада. Некоторые константы используют для задания значений параметров, принимаемых по умолчанию, процедурам ЯГС.
— Следующие константы определяют стандартные типы линий ЯГС:

SOLID_LINE : constant LINETYPE := 1;
DASHED_LINE : constant LINETYPE := 2;
DOTTED_LINE : constant LINETYPE := 3;
DASHED_DOTTED_LINE : constant LINETYPE := 4;

Следующие константы определяют стандартные типы маркеров ЯГС:

```
DOT_MARKER                    : constant MARKER_TYPE := 1;
PLUS_MARKER                   : constant MARKER_TYPE := 2;
STAR_MARKER                   : constant MARKER_TYPE := 3;
ZERO_MARKER                   : constant MARKER_TYPE := 4;
X_MARKER                      : constant MARKER_TYPE := 5;
```

Следующие константы определяют типы подсказок и эха, поддерживаемые ЯГС:

```
DEFAULT_LOCATOR               : constant LOCATOR_PROMPT_ECHO_TYPE := 1;
CROSS_HAIR_
LOCATOR                       : constant LOCATOR_PROMPT_ECHO_TYPE := 2;
TRACKING_CROSS_               : constant LOCATOR_PROMPT_ECHO_TYPE := 3;
LOCATOR
RUBBER_BAND_LINE
_LOCATOR                      : constant LOCATOR_PROMPT_ECHO_TYPE := 4;
RECTANGLE_LOCATOR             : constant LOCATOR_PROMPT_ECHO_TYPE := 5;
DIGITAL_LOCATOR               : constant LOCATOR_PROMPT_ECHO_TYPE := 6;

DEFAULT_STROKE                : constant STROKE_PROMPT_ECHO_TYPE := 1;
DIGITAL_STROKE                : constant STROKE_PROMPT_ECHO_TYPE := 2;
MARKER_STROKE                 : constant STROKE_PROMPT_ECHO_TYPE := 3;
LINE_STROKE                   : constant STROKE_PROMPT_ECHO_TYPE := 4;

DEFAULT_VALUATOR              : constant VALUATOR_PROMPT_
                                                  ECHO_TYPE := 1;
GRAFHICAL_                    : constant VALUATOR_PROMPT_
VALUATOR                                          ECHO TYPE := 2;
DIGITAL_VALUATOR              : constant VALUATOR_PROMPT_
                                                  ECHO_TYPE := 3;

DEFAULT_CHOICE                : constant CHOICE_PROMPT_ECHO_TYPE := 1;
PROMPT_ECHO_                  : constant CHOICE_PROMPT_ECHO_TYPE := 2;
CHOICE
STRING_PROMPT_                : constant CHOICE_PROMPT_ECHO_TYPE := 3;
CHOICE
STRING_INPUT_                 : constant CHOICE_PROMPT_ECHO_TYPE := 4;
_CHOICE
SEGMENT_CHOICE                : constant CHOICE_PROMPT_ECHO_TYPE := 5;

DEFAULT_STRING                : constant STRING_PROMPT_ECHO_TYPE := 1;

DEFAULT_PICK                  : constant PICK_PROMPT_ECHO_TYPE := 1;

GROUP-HIGHLIGHT_              : constant PICK_PROMPT_ECHO_TYPE := 2;
_PICK
SEGMENT_HIGHLIGHT_            : constant PICK_PROMPT_ECHO_TYPE := 3;
_PICK
```

— Следующие константы используют для задания принимаемых по умолчанию значений параметров процедурам ЯГС.

```
DEFAULT_MAMORY_               : constant                  := 0;
UNITS

DEFAULT_ERROR_FILE            : constant STRING           := « »;
end GKS_TYPES;
```

— ПАКЕТ ЯГС

```
with GKS_TYPES,
use GKS_TYPES,
package GKS is
```

— Пакет ЯГС содержит все процедуры, которые требуются для реализации уровня 2с ЯГС.

— Последующие типы данных являются личными и включены в пакет ЯГС для легкости манипулирования.

```
— CHOICE_DATA_RECORD                                          Уровень 0b
   type CHOICE_DATA_RECORD (PROMPT_ECHO_TYPE :
        CHOICE_PROMPT_ECHO_TYPE := DEFAULT_CHOICE) is private;
```

— Определяет запись для инициализации устройства выбора

```
— GKSM_DATA_RECORD                                            Уровень 0b
   type GKSM_DATA_RECORD (TYPE_OF_ITEM : GKSM_ITEM_TYPE := 0;
                          LENGTH : NATURAL := 0) is private;
```

— Запись данных для метафайла GKSM.

```
— LOCATOR_DATA_RECORD                                         Уровень 0b
   type LOCATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
        LOCATOR_PROMPT_ECHO_TYPE := DEFAULT_LOCATOR)
                                                    is private;
```

— Определяет запись для инициализации устройства ввода позиции.

```
— PICK_DATA_RECORD                                            Уровень 1b
   type PICK_DATA_RECORD (PROMPT_ECHO_TYPE:
     PICK_PROMPT_ECHO_TYPE := DEFAULT_PICK) is private;
```

— Определяет запись для инициализации устройства указания.

```
— STRING_DATA_RECORD                                          Уровень 0b
   type STRING_DATA_RECORD (PROMPT_ECHO_TYPE:
        STRING_PROMPT_ECHO_TYPE := DEFAULT_STRING) is private;
```

— Определяет запись для инициализации устройства ввода строки.

```
— STROKE_DATA_RECORD                                          Уровень 0b
   type STROKE_DATA_RECORD (PROMPT_ECHO_TYPE:
        STROKE_PROMPT_ECHO_TYPE := DEFAULT_STROKE) is private;
```

— Определяет запись для инициализации устройства ввода последовательности позиций.

```
— VALUATOR_DATA_RECORD                                        Уровень 0b
   type VALUATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
        VALUATOR_PROMPT_ECHO_TYPE := DEFAULT_VALUATOR)
                                                    is private;
```

— Определяет запись для инициализации устройства ввода числа.

— Подпрограммы для манипулирования записями данных ввода.

— Процедуры и функции, определенные ниже, необходимы для построения и опроса записей входных данных, декларированных как личные типы в пакете для каждого из шести классов устройств ввода, определенных спецификацией ЯГС. Процедуры, представленные ниже, используют для построения записей данных для каждого из типов подсказок и эха устройств, применяемых для инициализации конкретных устройств ввода. Также предоставлены соответствующие функции, позволяющие прикладному программному обеспечению ЯГС/Ада проанализировать части записи данных, которые определены ЯГС. Любую специфическую для реализации информацию в записях данных поддерживают личной и недоступной. Прерывания GKS_ERROR возникают, если любую из приведенных ниже процедур используют неправильно. Таким образом, если

применяют недопустимый тип подсказки и эха, в файле ошибок регистрируют ошибку номер 2500.

— Операции над записями данных устройства ввода позиций.

```
procedure BUILD_LOCATOR_DATA_RECORD
   (PROMPT_ECHO_TYPE          : in LOCATOR_PROMPT_ECHO_TYPE;
   DATA_RECORD                : out LOCATOR_DATA_RECORD);
```

— Строит и возвращает запись данных устройства ввода позиций.

```
procedure BUILD_STROKE_DATA_RECORD
   (PROMPT_ECHO_TYPE          : in STROKE_PROMPT_ECHO_TYPE;
   BUFFER_SIZE                : in POSITIVE;
   DATA_RECORD                : out STROKE_DATA_RECORD);
```

— Строит и возвращает запись данных устройства ввода последовательности позиций.

```
function BUFFER_SIZE (DATA_RECORD : in STROKE_DATA_RECORD)
   return POSITIVE;
```

— Операции над записями данных устройства ввода числа.

```
procedure BUILD_VALUATOR_DATA_RECORD
   (PROMPT_ECHO_TYPE          : in VALUATOR_PROMPT_ECHO_
                                                    TYPE;
   LOW_VALUE                  : in VALUATOR_INPUT_VALUE;
   HIGH_VALUE                 : in VALUATOR_INPUT_VALUE;
   DATA_RECORD                : out VALUATOR_DATA_RECORD);
```

— Строит и возвращает запись данных устройства ввода числа.

```
function HIGH_VALUE (DATA_RECORD : in VALUATOR_DATA_RECORD)
   return VALUATOR_INPUT_VALUE;
```

— Возвращает наибольшее число, запомненное в записи данных устройства ввода числа.

```
function LOW_VALUE (DATA_RECORD : in VALUATOR_DATA_RECORD)
   return VALUATOR_INPUT_VALUE;
```

— Возвращает наименьшее число, запомненное в записи данных устройства ввода числа.

— Операции над записями данных устройства выбора.

```
procedure BUILD_CHOICE_DATA_RECORD
   (PROMPT_ECHO_TYPE          : in CHOICE_PROMPT_ECHO_TYPE;
   DATA_RECORD                : out CHOICE_DATA_RECORD);
```

— Строит и возвращает запись данных устройства выбора.

— Операции над записями данных устройства указания.

```
procedure BUILD_PICK_DATA_RECORD
   (PROMPT_ECHO_TYPE          : in PICK_PROMPT_ECHO_TYPE;
   DATA_RECORD                : out PICK_DATA_RECORD);
```

— Строит и возвращает запись данных устройства указания.

— Операции над записями данных устройства ввода строки.

```
procedure BUILD_STRING_DATA_RECORD
   (PROMPT_ECHO_TYPE          : in STRING_PROMPT_ECHO_TYPE;
   INPUT_BUFFER_SIZE          : in POSITIVE;
```

```
    INITIAL_CURSOR_POSITION          : in NATURAL;
    DATA_RECORD                      : out STRING_DATA_RECORD);
```

— Строит и возвращает запись данных строки.

```
function INPUT_BUFFER_SIZE
    (DATA_RECORD : in STRING_DATA_RECORD) return NATURAL;
```

— Возвращает размер буфера, использованного для запоминания строки, размещенной в записи данных строки.

```
function INITIAL_CURSOR_POSITION
    (DATA_RECORD : in STRING_DATA_RECORD) return NATURAL;
```

— Возвращает начальную позицию курсора для устройства ввода строки, запомненного в записи данных устройства ввода строки.

— Процедуры ЯГС

— ФУНКЦИИ УПРАВЛЕНИЯ

```
procedure OPEN_GKS
    (ERROR_FILE                      : in STRING := DEFAULT_ERROR_
                                                          FILE;
    AMOUNT_OF_MEMORY                 : in NATURAL := DEFAULT_MEMORY_
                                                          UNITS);

procedure CLOSE_GKS;
procedure OPEN_WS
    (WS                              : in WS_ID;
    CONNECTION                       : in STRING;
    TYPE_OF_WS                       : in WS_TYPE);

procedure CLOSE_WS
    (WS : in WS_ID);

procedure ACTIVATE_WS
    (WS : in WS_ID);

procedure DEACTIVATE_WS
    (WS : in WS_ID);

procedure CLEAR_WS
    (WS : in WS_ID);
    FLAG : in CONTROL_FLAG);

procedure REDRAW_ALL_SEGMENTS_ON_WS
    (WS : in WS_ID);

procedure UPDATE_WS
    (WS                              : in WS_ID;
    REGENERATION                     : in UPDATE_REGENERATION_
                                                          FLAG);

procedure SET_DEFERRAL_STATE
    (WS                              : in WS_ID;
    DEFERRAL                         : in DEFERRAL_MODE;
    REGENERATION                     : in REGENERATION_MODE);

procedure MESSAGE
    (WS                              : in WS_ID;
    CONTENTS                         : in STRING);
```

```
— ФУНКЦИИ ВВОДА ГРАФИЧЕСКИХ ДАННЫХ
procedure POLYLINE
    (POINTS                          : in WC.POINT_ARRAY);
procedure POLYMARKER
    (POINTS                          : in WC.POINT_ARRAY);
procedure TEXT
    (POSITION                        : in WC.POINT;
    CHAR_STRING                      : in STRING);
procedure FILL_AREA
    (POINTS                          : in WC.POINT_ARRAY);
procedure CELL_ARRAY
    (CORNER_1_1                      : in WC.POINT;
    CORNER_DX_DY                     : in WC.POINT;
    CELLS                            : in COLOUR_MATRIX);
— ФУНКЦИИ ЗАДАНИЯ АТРИБУТОВ ВЫХОДНЫХ ДАННЫХ
procedure SET_POLYLINE_INDEX
    (INDEX                           : in POLYLINE_INDEX);
procedure SET_LINETYPE
    (TYPE_OF_LINE                    : in LINETYPE);
procedure SET_LINEWIDTH_SCALE_FACTOR
    (WIDTH                           : in LINEWIDTH);
procedure SET_POLYLINE_COLOUR_INDEX
    (LINE_COLOUR                     : in COLOUR_INDEX);
procedure SET_POLYMARKER_INDEX
    (INDEX                           : in POLYMARKER_INDEX);
procedure SET_MARKER_TYPE
    (TYPE_OF_MARKER                  : in MARKER_TYPE);
procedure SET_MARKER_SIZE_SCALE_FACTOR
    (SIZE                            : in MARKER_SIZE);
procedure SET_POLYMARKER_COLOUR_INDEX
    (MARKER_COLOUR                   : in COLOUR_INDEX);
procedure SET_TEXT_INDEX
    (INDEX                           : in TEXT_INDEX);
procedure SET_TEXT_FONT_AND_PRECISION
    (FONT_PRECISION                  : in TEXT_FONT_PRECISION);
procedure SET_CHAR_EXPANSION_FACTOR
    (EXPANSION                       : in CHAR_EXPANSION);
procedure SET_CHAR_SPACING
    (SPASING                         : in CHAR_SPACING);
procedure SET_TEXT_COLOUR_INDEX
    (TEXT_COLOUR                     : in COLOUR_INDEX);
procedure SET_CHAR_HEIGHT
    (HEIGHT                          : in WC.MAGNITUDE);
```

```
procedure SET_CHAR_UP_VECTOR
    (CHAR_UP_VECTOR              : in WC.VECTOR);
procedure SET_TEXT_PATH
    (PATH                        : in TEXT_PATH);
procedure SET_TEXT_ALIGNMENT
    (ALIGNMEN                    : in TEXT_ALIGNMENT);
procedure SET_FILL_AREA_INDEX
    (INDEX                       : in FILL_AREA_INDEX);
procedure SET_FILL_AREA_INTERIOR_STYLE
    (INTERIOR                    : in INTERIOR_STYLE);
procedure SET_FILL_AREA_STYLE_INDEX
    (STYLE                       : in STYLE_INDEX);
procedure SET_FILL_AREA_COLOUR_INDEX
    (FILL_AREA_COLOUR            : in COLOUR_INDEX);
procedure SET_PATTERN_SIZE
    (SIZE                        : in WC.SIZE);
procedure SET_PATTERN_REFERENCE_POINT
    (POINT                       : in WC.POINT);
procedure SET_ASF
    (ASF                         : in ASF.LIST);
procedure SET_PICK_ID
    (PICK                        : in PICK_ID);
procedure SET_POLYLINE_REPRESENTATION
    (WS                          : in WS_ID;
    INDEX                        : in POLYLINE_INDEX;
    TYPE_OF_LINE                 : in LINETYPE;
    WIDTH                        : in LINEWIDTH;
    LINE_COLOUR                  : in COLOUR_INDEX);
procedure SET_POLYMARKER_REPRESENTATION
    (WS                          : in WS-ID;
    INDEX                        : in POLYMARKER_INDEX;
    TYPE_OF_MARKER               : in MARKER_TYPE;
    SIZE                         : in MARKER_SIZE;
    MARKER_COLOUR                : in COLOUR_INDEX),
procedure SET_TEXT_REPRESENTATION
    (WS                          : in WS_ID;
    INDEX                        : in TEXT_INDEX;
    FONT_PRECISION               : in TEXT_FONT_PRECISION;
    EXPANSION                    : in CHAR_EXPANSION;
    SPACING                      : in CHAR_SPACING;
    TEXT_COLOUR                  : in COLOUR_INDEX);
procedure SET_FILL_AREA_REPRESENTATION
    (WS                          : in WS_ID;
    INDEX                        : in FILL_AREA_INDEX;
    INTERIOR                     : in INTERIOR_STYLE;
    STYLE                        : in STYLE_INDEX;
    FILL_AREA_COLOUR             : in COLOUR_INDEX);
```

```
procedure SET_PATTERN_REPRESENTATION
    (WS                              : in WS_ID;
    INDEX                            : in PATTERN_INDEX;
    PATTERN                          : in COLOUR_MATRIX);

procedure SET_COLOUR_REPRESENTATION
    (WS                              : in WS_ID;
    INDEX                            : in COLOUR_INDEX;
    RGB_COLOUR                       : in COLOUR_REPRESENTATION);

— ФУНКЦИИ ПРЕОБРАЗОВАНИЯ

procedure SET_WINDOW
    (TRANSFORMATION                  : in POSITIVE_TRANSFORMATION_
                                                          _NUMBER;
    WINDOW_LIMITS                    : in WC.RECTANGLE_LIMITS);

procedure SET_VIEWPORT
    (TRANSFORMATION                  : in POSITIVE_TRANSFORMATION_
                                                          _NUMBER;
    VIEWPORT_LIMITS                  : in NDC.RECTANGLE_LIMITS);

procedure SET_VIEWPORT_INPUT_PRIORITY
    (TRANSFORMATION                  : in TRANSFORMATION_NUMBER;
    REFERENCE_                       : in TRANSFORMATION_NUMBER;
TRANSFORMATION
    PRIORITY                         : in RELATIVE_PRIORITY);

procedure SELECT_NORMALIZATION_TRANSFORMATION
    (TRANSFORMATION                  : in TRANSFORMATION_NUMBER);

procedure SET_CLIPPING_INDICATOR
    (CLIPPING                        : in CLIPPING_INDICATOR);

procedure SET_WS_WINDOW
    (WS                              : in WS_ID;
    WS_WINDOW_LIMITS                 : in NDC.RECTANGLE_LIMITS);

procedure SET_WS_VIEWPORT
    (WS                              : in WS_ID;
    WS_VIEWPORT_LIMITS               : in DC.RESTANGLE_LIMITS);

— ФУНКЦИИ СЕГМЕНТАЦИИ

procedure CREATE_SEGMENT
    (SEGMENT                         : in SEGMENT_NAME);

procedure CLOSE_SEGMENT;
procedure RENAME_SEGMENT
    (OLD_NAME                        : in SEGMENT_NAME);
    NEW_NAME                         : in SEGMENT_NAME);

procedure DELETE_SEGMENT
    (SEGMENT                         : in SEGMENT_NAME);

procedure DELETE_SEGMENT_FROM_WS
    (WS                              : in WS_ID;
    SEGMENT                          : in SEGMENT_NAME);

procedure ASSOCIATE_SEGMENT_WITH_WS
    (WS                              : in WS_ID;
    SEGMENT                          : in SEGMENT_NAME);
```

```
procedure COPY_SEGMENT_TO_WS
    (WS                              : in WS_ID;
    SEGMENT                          : in SEGMENT_NAME);
procedure INSERT_SEGMENT
    (SEGMENT                         : in SEGMENT_NAME;
    TRANSFORMATION                   : in TRANSFORMATION_MATRIX);
procedure SET_SEGMENT_TRANSFORMATION
    (SEGMENT                         : in SEGMENT_NAME;
    TRANSFORMATION                   : in TRANSFORMATION_MATRIX);
procedure SET_VISIBILITY
    (SEGMENT                         : in SEGMENT_NAME;
    VISIBILITY                       : in SEGMENT_VISIBILITY);
procedure SET_HIGHLIGHTING
    (SEGMENT                         : in SEGMENT_NAME;
    HIGHLIGHTING                     : in SEGMENT_HIGHLIGHTING);
procedure SET_SEGMENT_PRIORITY
    (SEGMENT                         : in SEGMENT_NAME;
    PRIORITY                         : in SEGMENT_PRIORITY);
procedure SET_DETECTABILITY
    (SEGMENT                         : in SEGMENT_NAME;
    DETECTABILITY                    : in SEGMENT_DETECTABILITY);
— ФУНКЦИИ ВВОДА
procedure INITIALISE_LOCATOR
    (WS                              : in WS_ID;
    DEVICE                           : in LOCATOR_DEVICE_NUMBER;
    INITIAL_TRANSFORMATION           : in TRANSFORMATION_NUMBER;
    INITIAL_POSITION                 : in WC.POINT;
    ECHO_AREA                        : in DC.RECTANGLE_LIMITS;
    DATA_RECORD                      : in LOCATOR_DATA_RECORD);
procedure INITIALISE_STROKE
    (WS                              : in WS_ID;
    DEVICE                           : in STROKE_DEVICE_NUMBER;
    INITIAL_TRANSFORMATION           : in TRANSFORMATION_NUMBER;
    INITIAL_STROKE                   : in WC.POINT_ARRAY;
    ECHO_AREA                        : in DC.RESTANGLE_LIMITS;
    DATA_RECORD                      : in STROKE_DATA_RECORD);
procedure INITIALISE_VALUATOR
    (WS                              : in WS_ID;
    DEVICE                           : in VALUATOR_DEVICE_NUMBER;
    INITIAL_VALUE                    : in VALUATOR_INPUT_VALUE;
    ECHO_AREA                        : in DC.RECTANGLE_LIMITS;
    DATA_RECORD                      : in VALUATOR_DATA_RECORD);
procedure INITIALISE_CHOICE
    (WS                              : in WS_ID;
    DEVICE                           : in CHOICE_DEVICE_NUMBER;
    INITIAL_STATUS                   : in CHOICE_STATUS;
    INITIAL_CHOICE                   : in CHOICE_VALUE;
    ECHO_AREA                        : in DC.RESTANGLE_LIMITS;
    DATA_RECORD                      : in CHOICE_DATA_RECORD);
```

```
procedure INITIALISE_PICK
    (WS                          : in WS_ID;
    DEVICE                       : in PICK_DEVICE_NUMBER;
    INITIAL_STATUS               : in PICK_STATUS;
    INITIAL_SEGMENT              : in SEGMENT_NAME;
    INITIAL_PICK                 : in PICK_ID;
    ECHO_AREA                    : in DC.RESTANGLE_LIMITS;
    DATA_RECORD                  : in PICK_DATA_RECORD);

procedure INITIALISE_STRING
    (WS                          : in WS_ID;
    DEVICE                       : in STRING_DEVICE_NUMBER;
    INITIAL_STRING               : in INPUT_STRING;
    ECHO_AREA                    : in DC.RECTANGLE_LIMITS;
    DATA_RECORD                  : in STRING_DATA_RECORD);

procedure SET_LOCATOR_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in LOCATOR_DEVICE_NUMBER;
    MODE                         : in OPERATION_MODE;
    SWITCH                       : in ECHO_SWITCH);

procedure SET_STROKE_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in STROKE_DEVICE_NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);

procedure SET_VALUATOR_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in VALUATOR_DEVICE_NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);

procedure SET_CHOICE_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in CHOICE_DEVICE_NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);

procedure SET_PICK_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in PICK_DEVICE_NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);

procedure SET_STRING_MODE
    (WS                          : in WS_ID;
    DEVICE                       : in STRING_DEVICE_NUMBER;
    MODE                         : in OPERATING_MODE;
    SWITCH                       : in ECHO_SWITCH);

procedure REQUEST_LOCATOR
    (WS                          : in WS_ID;
    DEVICE                       : in LOCATOR_DEVICE_NUMBER;
    STATUS                       : out INPUT_STATUS;
    TRANSFORMATION               : out TRANSFORMATION_NUMBER;
    POSITION                     : out WC.POINT);
```

```
procedure REQUEST_STROKE
    (WS                          : in WS_ID;
    DEVICE                       : in STROKE_DEVICE_NUMBER;
    STATUS                       : out INPUT_STATUS;
    TRANSFORMATION               : out TRANSFORMATION_NUMBER;
    STROKE_POINTS                : out WC.POINT_LIST);

procedure REQUEST VALUATOR
    (WS                          : in WS_ID;
    DEVICE                       : in VALUATOR_DEVICE_NUMBER;
    STATUS                       : out INPUT_STATUS;
    VALUE                        : out VALUATOR_INPUT_VALUE);

procedure REQUEST_CHOICE
    (WS                          : in WS_ID;
    DEVICE                       : in CHOICE_DEVICE_NUMBER;
    STATUS                       : out CHOICE_REQUEST_STATUS;
    CHOICE_NUMBER                : out CHOICE_VALUE);

procedure REQUEST_PICK
    (WS                          : in WS_ID;
    DEVICE                       : in PICK_DEVICE_NUMBER;
    STATUS                       : out PICK_REQUEST_STATUS;
    SEGMENT                      : out SEGMENT_NAME;
    PICK                         : out PICK_ID);

procedure REQUEST_STRING
    (WS                          : in WS_ID;
    DEVICE                       : in STRING_DEVICE_NUMBER;
    STATUS                       : out INPUT_STATUS;
  CHAR_STRING                    : out INPUT_STRING);

procedure SAMPLE_LOCATOR
    (WS                          : in WS_ID;
    DEVICE                       : in LOCATOR_DEVICE_NUMBER;
    TRANSFORMATION               : out TRANSFORMATION_NUMBER;
    POSITION                     : out WC.POINT);

procedure SAMPLE_STROKE
    (WS                          : in WS_ID;
    DEVICE                       : in STROKE_DEVICE_NUMBER;
    TRANSFORMATION               : out TRANSFORMATION_NUMBER;
    STROKE_POINTS                : out WC.POINT_LIST);

procedure SAMPLE_VALUATOR
    (WS                          : in WS_ID;
    DEVICE                       : in VALUATOR_DEVICE_NUMBER;
    VALUE                        : out VALUATOR_INPUT_VALUE);

procedure SAMPLE_CHOICE
    (WS                          : in WS_ID;
    DEVICE                       : in CHOICE_DEVICE_NUMBER;
    STATUS                       : out CHOICE_STATUS;
    CHOICE_NUMBER                : out CHOICE_VALUE);

procedure SAMPLE_PICK
    (WS                          : in WS_ID;
    DEVICE                       : in PICK_DEVICE_NUMBER;
```

```
    STATUS                          : out PICK_STATUS;
    SEGMENT                         : out SEGMENT_NAME;
    PICK                            : out PICK_ID);

procedure SAMPLE_STRING
    (WS                             : in WS_ID;
    DEVICE                          : in STRING_DEVICE_NUMBER;
    CHAR_STRING                     : out INPUT_STRING);

procedure AWAIT_EVENT
    (TIMEOUT                        : in DURATION;
    WS                              : out WS_ID;
    CLASS                           : out INPUT_CLASS;
    DEVICE                          : out EVENT_DEVICE_NUMBER);

procedure FLUSH_DEVICE_EVENTS
    (WS                             : in WS_ID;
    CLASS                           : in INPUT_QUEUE_CLASS;
    DEVICE                          : in EVENT_OVERFLOW_DEVICE_
                                                        NUMBER);

procedure GET_LOCATOR
    (TRANSFORMATION                 : out TRANSFORMATION_NUMBER;
    POSITION                        : out WC.POINT);

procedure GET_STROKE
    (TRANSFORMATION                 : out TRANSFORMATION_NUMBER;
    STROKE_POINTS                   : out WC.POINT_LIST);

procedure GET_VALUATOR
    (VALUE                          : out VALUATOR_INPUT_VALUE);

procedure GET_CHOICE
    (STATUS                         : out CHOICE_STATUS;
    CHOICE_NUMBER                   : out CHOICE_VALUE);

procedure GET_PICK
    (STATUS                         : out PICK_STATUS;
    SEGMENT                         : out SEGMENT_NAME;
    PICK                            : out PICK_ID);

procedure GET_STRING
    (CHAR_STRING                    : out INPUT_STRING);
```

— ФУНКЦИИ МЕТАФАЙЛА

```
procedure WRITE_ITEM_TO_GKSM
    (WS                             : in WS_ID;
    ITEM                            : in GKSM_DATA_RECORD);

procedure GET_ITEM_TYPE_FROM_GKSM
    (WS                             : in WS_ID;
    TYPE_OF_ITEM                    : out GKSM_ITEM_TYPE;
    LENGTH                          : out NATURAL);

procedure READ_ITEM_FROM_GKSM
    (WS                             : in WS_ID;
    MAX_LENGTH                      : in NATURAL;
    ITEM                            : out GKSM_DATA_RECORD);
```

```
procedure INTERPRET_ITEM
    (ITEM                          : in GKSM_DATA_RECORD);
— СПРАВОЧНЫЕ ФУНКЦИИ
procedure INQ_OPERATING_STATE_VALUE
    (VALUE                         : out OPERATING_STATE);
procedure INQ_LEVEL_OF_GKS
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    LEVEL                          : out GKS_LEVEL);
procedure INQ_LIST_OF_AVAILABLE_WS_TYPES
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    TYPES                          : out WS_TYPES.LIST_OF);
procedure INQ_WS_MAX_NUMBERS
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    MAX_OPEN_WS                    : out POSITIVE;
    MAX_ACTIVE_WS                  : out POSITIVE;
    MAX_SEGMENT_WS                 : out POSITIVE);
procedure INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    TRANSFORMATION                 : out TRANSFORMATION_NUMBER);
procedure INQ_SET_OF_OPEN_WS
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    WS                             : out WS_IDS.LIST_OF);
procedure INQ_SET_OF_ACTIVE_WS
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    WS                             : out WS_IDS.LIST_OF);
procedure INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    ATTRIBUTES                     : out PRIMITIVE_ATTRIBUTE_
                                                    VALUES);
procedure INQ_POLYLINE_INDEX
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    INDEX                          : out POLYLINE_INDEX);
procedure INQ_POLYMARKER_INDEX
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    INDEX                          : out POLYMARKER_INDEX);
procedure INQ_TEXT_INDEX
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    INDEX                          : out TEXT_INDEX);
procedure INQ_CHAR_HEIGHT
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    HEIGHT                         : out WC.MAGNITUDE);
procedure INQ_CHAR_UP_VECTOR
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    VECTOR                         : out WC.VECTOR);
procedure INQ_CHAR_WIDTH
    (ERROR_INDICATOR               : out ERROR_NUMBER;
    WIDTH                          : out WC.MAGNITUDE);
```

```
procedure INQ_CHAR_BASE_VECTOR
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    VECTOR                          : out WC.VECTOR);

procedure INQ_TEXT_PATH
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    PATH                            : out TEXT_PATH);

procedure INQ_TEXT_ALIGNMENT
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    ALIGNMENT                       : out TEXT_ALIGNMENT);

procedure INQ_FILL_AREA_INDEX
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    INDEX                           : out FILL_AREA_INDEX);

procedure INQ_PATTERN_WIDTH_VECTOR
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    WIDTH                           : out WC.VECTOR);

procedure INQ_PATTERN_HEIGHT_VECTOR
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    VECTOR                          : out WC.VECTOR);

procedure INQ_PATTERN_REFERENCE_POINT
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    REFERENCE_POINT                 : out WC.POINT);

procedure INQ_CURRENT_PICK_ID_VALUE
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    PICK                            : out PICK_ID);

procedure INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    ATTRIBUTES                      : out INDIVIDUAL_ATTRIBUTE_
                                                        VALUES);

procedure INQ_LINETYPE
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    TYPE_OF_LINE                    : out LINETYPE);

procedure INQ_LINEWIDTH_SCALE_FACTOR
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    WIDTH                           : out LINEWIDTH);

procedure INQ_POLYLINE_COLOUR_INDEX
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    LINE_COLOUR                     : out COLOUR_INDEX);

procedure INQ_POLYMARKER_TYPE
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    TYPE_OF_MARKER                  : out MARKER_TYPE);

procedure INQ_POLYMARKER_SIZE_SCALE_FACTOR
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    SIZE                            : out MARKER_SIZE);

procedure INQ_POLYMARKER_COLOUR_INDEX
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    MARKER_COLOUR                   : out COLOUR_INDEX);
```

```
procedure INQ_TEXT_FONT_AND_PRECISION
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    FONT_PRECISION                  : out TEXT_FONT_PRECISION);

procedure INQ_CHAR_EXPANSION_FACTOR
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    EXPANSION                       : out CHAR_EXPANSION);

procedure INQ_CHAR_SPASING
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    SPACING                         . out CHAR_SPACING);

procedure INQ_TEXT_COLOUR_INDEX
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    TEXT_COLOUR                     : out COLOUR_INDEX);

procedure INQ_FILL_AREA_INTERIOR_STYLE
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    INTERIOR                        ' out INTERIOR_STYLE);

procedure INQ_FILL_AREA_STYLE_INDEX
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    STYLE                           : out STYLE_INDEX);

procedure INQ_FILL_AREA_COLOUR_INDEX
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    FILL_AREA_COLOUR                : out COLOUR_INDEX);

procedure INQ_LIST_OF_ASF
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    LIST                            : out ASF_LIST);

procedure INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    TRANSFORMATION                  ' out TRANSFORMATION_NUMBER);

procedure INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBERS
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    LIST                            : out TRANSFORMATION_PRIORITY_
                                                        _LIST);

procedure INQ_NORMALIZATION_TRANSFORMATION
    (TRANSFORMATION                 : in TRANSFORMATION_NUMBER;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    WINDOW_LIMITS                   : out WC.RESTANGLE_LIMITS;
    VIEWPORT_LIMITS                 : out NDC.RECTENGLE_LIMITS);

procedure INQ_CLIPPING
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    CLIPPING                        : out CLIPPING_INDICATOR;
    CLIPPING_RECTANGLE_             : out NDC.RECTANGLE_LIMITS);
LIMITS

procedure INQ_NAME_OF_OPEN_SEGMENT
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    SEGMENT                         : out SEGMENT_NAME);

procedure INQ_SET_OF_SEGMENT_NAMES_IN_USE
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    SEGMENTS                        : out SEGMENT_NAMES.LIST_OF);
```

```
procedure INQ_MORE_SIMULTANEOUS_EVENTS
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    EVENTS                          . out MORE EVENTS);

procedure INQ_WS_CONNECTION_AND_TYPE
    (WS                             . in WS_ID;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    CONNECTION                      : out VARIABLE_CONNECTION_ID.
    TYPE_OF_WS                      . out WS_TYPE);

procedure INQ_WS_STATE
    (WS                             . in WS_ID;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    STATE                           : out WS_STATE);

procedure INQ_WS_DEFERRAL_AND_UPDATE_STATES
    (WS                             : in WS_ID;
    ERROR_INDICATOR                 : out ERROR_NUMBER.
    DEFERRAL                        : out DEFERRAL_MODE;
    REGENERATION                    : out REGENERATION_MODE;
    DISPLAY                         : out DISPLAY_SURFASE_EMPTY;
    FRAME_ACTION                    · out NEW_FRAME_NECESSARY);

procedure INQ_LIST_OF_POLYLINE_INDICES
    (WS                             . in WS_ID;
    ERROR_INDICATOR                 . out ERROR_NUMBER;
    INDICES                         : out POLYLINE_INDICES.LIST_OF);

procedure INQ_POLYLINE_REPRESENTATION
    (WS                             : in WS_ID;
    INDEX                           : in POLYLINE_INDEX;
    RETURNED_VALUES                 : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    TYPE_OF_LINE                    · out LINETYPE;
    WIDTH                           : out LINEWIDTH;
    LINE_COLOUR                     · out COLOUR_INDEX);

procedure INQ_LIST_OF_POLYMARKER_INDICES
    (WS                             : in WS_ID;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    INDICES                         : out POLYMARKER_INDICES.LIST_
                                                              _OF);

procedure INQ_POLYMARKER_REPRESENTATION
    (WS                             · in WS_ID;
    INDEX                           . in POLYMARKER_INDEX;
    RETURNED_VALUES                 : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    TYPE_OF_MARKER                  i out MARKER_TYPE;
    SIZE                            : out MARKER_SIZE,
    MARKER_COLOUR                   : out COLOUR_INDEX);

procedure INQ_LIST_OF_TEXT_INDICES
    (WS                             : in WS_ID;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    INDICES                         : out TEXT_INDICES.LIST_OF);

procedure INQ_TEXT_REPRESENTATION
    (WS                             : in WS_ID;
```

```
    INDEX                          : in TEXT_INDEX;
    RETURNED_VALUES                : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    FONT_PRECISION                 : out TEXT_FONT_PRECISION;
    EXPANSION                      : out CHAR_EXPANSION;
    SPACING                        : out CHAR_SPACING
    TEXT_COLOUR                    : out COLOUR_INDEX);

procedure INQ_TEXT_EXTENT
    (WS                            : in WS_ID;
    POSITION                       : in WC.POINT;
    CHAR_STRING                    : in STRING;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    CONCATENATION_POINT            : out WC.POINT;
    TEXT_EXTENT                    : out TEXT_EXTENT_PARALLELO-
                                                        GRAM);

procedure INQ_LIST_OF_FILL_AREA_INDICES
    (WS                            : in WS_ID;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    INDICES                        : out FILL_AREA_INDICES.LIST_OF);

procedure INQ_FILL_AREA_REPRESENTATION
    (WS                            : in WS_ID;
    INDEX                          : in FILL_AREA_INDEX;
    RETURNED_VALUES                : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    INTERIOR                       : out INTERIOR_STYLE;
    STYLE                          : out STYLE_INDEX;
    FILL_AREA_COLOUR               : out COLOUR_INDEX);

procedure INQ_LIST_OF_PATTERN_INDICES
    (WS                            : in WS_ID;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    INDICES                        : out PATTERN_INDICES.LIST_OF);

procedure INQ_PATTERN_REPRESENTATION
    (WS                            : in WS_ID;
    INDEX                          : in PATTERN_INDEX;
    RETURNED_VALUES                : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    PATTERN                        : out VARIABLE_COLOUR_MATRIX);

procedure INQ_LIST_OF_COLOUR_INDICES
    (WS                            : in WS_ID;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    INDICES                        : out COLOUR_INDICES.LIST_OF);

procedure INQ_COLOUR_REPRESENTATION
    (WS                            : in WS_ID;
    INDEX                          : in COLOUR_INDEX;
    RETURNED_VALUES                : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    RGB_COLOUR                     : out COLOUR_REPRESENTATION);

procedure INQ_WS_TRANSFORMATION
    (WS                            : in WS_ID;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    UPDATE                         : out UPDATE_STATE;
```

```
    REQUESTED_WINDOW                : out NDC_RECTANGLE_LIMITS;
    CURRENT_WINDOW                  : out NDC_RECTANGLE_LIMITS;
    REQUESTED_VIEWPORT              : out DC_RECTANGLE_LIMITS;
    CURRENT_VIEWPORT                : out DC_RECTANGLE_LIMITS);

procedure INQ_SET_OF_SEGMENT_NAMES_ON_WS
    (WS                             : in WS_ID;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    SEGMENTS                        : out SEGMENT_NAMES_LIST_OF);

procedure INQ_LOCATOR_DEVICE_STATE
    (WS                             : in WS_ID;
    DEVICE                          : in LOCATOR_DEVICE_NUMBER;
    RETURNED_VALUES                 : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    MODE                            : out OPERATING_MODE;
    SWITCH                          : out ECHO_SWITCH;
    INITIAL_TRANSFORMATION          : out TRANSFORMATION_NUMBER;
    INITIAL_POSITION                : out WC_POINT;
    ECHO_AREA                       : out DC_RECTANGLE_LIMITS;
    DATA_RECORD                     : out LOCATOR_DATA_RECORD);

procedure INQ_STROKE_DEVICE_STATE
    (WS                             : in WS_ID;
    DEVICE                          : in STROKE_DEVICE_NUMBER;
    RETURNED_VALUES                 : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    MODE                            : out OPERATING_MODE;
    SWITCH                          : out ECHO_SWITCH;
    INITIAL_TRANSFORMATION          : out TRANSFORMATION_NUMBER;
    INITIAL_STROKE_POINTS           : out WC_POINT_LIST;
    ECHO_AREA                       : out DC_RECTANGLE_LIMITS;
    DATA_RECORD                     : out STROKE_DATA_RECORD);

procedure INQ_VALUATOR_DEVICE_STATE
    (WS                             : in WS_ID;
    DEVICE                          : in VALUATOR_DEVICE_NUMBER;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    MODE                            : out OPERATING_MODE;
    SWITCH                          : out ECHO_SWITCH;
    INITIAL_VALUE                   : out VALUATOR_INPUT_VALUE;
    ECHO_AREA                       : out DC_RECTANGLE_LIMITS;
    DATA_RECORD                     : out VALUATOR_DATA_RECORD);

procedure INQ_CHOICE_DEVICE_STATE
    (WS                             : in WS_ID;
    DEVICE                          : in CHOICE_DEVICE_NUMBER;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    MODE                            : out OPERATING_MODE;
    SWITCH                          : out ECHO_SWITCH;
    INITIAL_STATUS                  : out CHOICE_STATUS;
    INITIAL_CHOICE                  : out CHOICE_VALUE;
    ECHO_AREA                       : out DC_RECTANGLE_LIMITS;
    DATA_RECORD                     : out CHOICE_DATA_RECORD);

procedure INQ_PICK_DEVICE_STATE
    (WS                             : in WS_ID;
    DEVICE                          : in PICK_DEVICE_NUMBER;
```

```
    RETURNED_VALUES                 : in RETURN_VALUE_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    MODE                            : out OPERATING_MODE;
    SWITCH                          : out ECHO_SWITCH;
    INITIAL_STATUS                  : out PICK_STATUS;
    INITIAL_SEGMENT                 : out SEGMENT_NAME;
    INITIAL_PICK                    : out PICK_ID;
    ECHO_AREA                       : out  DC.RECTANGLE_LIMITS;
    DATA_RECORD                     : out PICK_DATA_RECORD);
procedure INQ_STRING_DEVICE_STATE
    (WS                             : in WS_ID;
    DEVICE                          : in STRING_DEVICE_NUMBER;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    MODE                            : out OPERATING_MODE;
    SWITCH                          : out ECHO_SWITCH;
    INITIAL_STRING                  : out INPUT_STRING;
    ECHO_AREA                       : out DC.RECTANGLE_LIMITS;
    DATA_RECORD                     : out STRING_DATA_RECORD);
procedure INQ_WS_CATEGORY
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    CATEGORY                        : out WS_CATEGORY);
procedure INQ_WS_CLASSIFICATION
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    CLASS                           : out DISPLAY_CLASS);
procedure INQ_DISPLAY_SPACE_SIZE
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    UNITS                           : out DC_UNITS;
    MAX_DC_SIZE                     : out DC.SIZE;
    MAX_RASTER_UNIT_SIZE            : out RASTER_UNIT_SIZE);
procedure INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    POLYLINE_REPRESENTATION         : out DYNAMIC_MODIFICATION;
    POLYMARKER_                     : out DYNAMIC_MODIFICATION;
REPRESENTATION
    TEXT_REPRESENTATION             : out DYNAMIC_MODIFICATION;
    FILL_AREA_                      : out DYNAMIC_MODIFICATION;
REPRESENTATION
    PATTERN_REPRESENTATION          : out DYNAMIC_MODIFICATION;
    COLOUR_REPRESENTATION           : out DYNAMIC_MODIFICATION;
    TRANSFORMATION                  : out DYNAMIC_MODIFICATION);
procedure INQ_DEFAULT_DEFERRAL_STATE_VALUES
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    DEFERRAL                        : out DEFERRAL_MODE;
    REGENERATION                    : out REGENERATION_MODE);
procedure INQ_POLYLINE_FACILITIES
    (TYPE_OF_WS                     : in WS_TYPE;
```

```
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    LIST_OF_TYPES                   : out LINETYPES.LIST_OF;
    NUMBER_OF_WIDTHS                : out NATURAL;
    NOMINAL_WIDTH                   : out DC.MAGNITUDE;
    RANGE_OF_WIDTHS                 : out DC.RANGE_OF_MAGNITUDES;
    NUMBER_OF_INDICES               : out NATURAL);

procedure INQ_PREDEFINED_POLYLINE_REPRESENTATION
    (TYPE_OF_WS                     : in WS_TYPE;
    INDEX                           : in POLYLINE_INDEX;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    TYPE_OF_LINE                    : out LINETYPE;
    WIDTH                           : out LINEWIDTH;
    LINE_COLOUR                     : out COLOUR_INDEX);

procedure INQ_POLYMARKER_FACILITIES
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    LIST_OF_TYPES                   : out MARKER_TYPES.LIST_OF;
    NUMBER_OF_SIZES                 : out NATURAL;
    NOMINAL_SIZE                    : out DC.MAGNITUDE;
    RANGE_OF_SIZES                  : out DC.RANGE_OF_MAGNITUDES;
    NUMBER_OF_INDICES               : out NATURAL);

procedure INQ_PREDEFINED_POLYMARKER_REPRESENTATION
    (TYPE_OF_WS                     : in WS_TYPE;
    INDEX                           : in POLYMARKER_INDEX;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    TYPE_OF_MARKER                  : out MARKER_TYPE;
    SIZE                            : out MARKER_SIZE;
    MARKER_COLOUR                   : out COLOUR_INDEX);

procedure INQ_TEXT_FACILITIES
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    LIST_OF_FONT_PRECISION_         : out
    _PAIRS                            TEXT_FONT_PRECISION.LIST_OF;
    NUMBER_OF_HEIGHTS               : out NATURAL;
    RANGE_OF_HEIGHTS                : out DC.RANGE_OF_MAGNITUDES;
    NUMBER_OF_EXPANSIONS            : out NATURAL;
    EXPANSION_RANGE                 : out RANGE_OF_EXPANSIONS;
    NUMBER_OF_INDICES               : out NATURAL);

procedure INQ_PREDEFINED_TEXT_REPRESENTATION
    (TYPE_OF_WS                     : in WS_TYPE;
    INDEX                           : in TEXT_INDEX;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    FONT_PRECISION                  : out TEXT_FONT_PRECISION;
    EXPANSION                       : out CHAR_EXPANSION;
    SPACING                         : out CHAR_SPACING;
    TEXT_COLOUR                     : out COLOUR_INDEX);

procedure INQ_FILL_AREA_FACILITIES
    (TYPE_OF_WS                     : in WS_TYPE;
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    LIST_OF_INTERIOR_STYLES         : out INTERIOR_STYLES.LIST_OF;
    LIST_OF_HATCH_STYLES            : out HATCH_STYLES.LAST_OF;
    NUMBER_OF_INDICES               : out NATURAL);
```

```
procedure INQ_PREDEFINED_FILL_AREA_REPRESENTATION
    (TYPE_OF_WS                 : in WS_TYPE;
    INDEX                       : in FILL_AREA_INDEX;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    INTERIOR                    : out INTERIOR_STYLE;
    STYLE                       : out STYLE_INDEX;
    FILL_AREA_COLOUR            : out COLOUR_INDEX);

procedure INQ_PATTERN_FACILITIES
    (TYPE_OF_WS                 : in WS_TYPE;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    NUMBER_OF_INDICES           : out NATURAL);

procedure INQ_PREDEFINED_PATTERN_REPRESENTATION
    (TYPE_OF_WS                 : in WS_TYPE;
    INDEX                       : in PATTERN_INDEX;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    PATTERN                     : out VARIABLE_COLOUR_MATRIX);

procedure INQ_COLOUR_FACILITIES
    (TYPE_OF_WS                 : in WS_TYPE;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    NUMBER_OF_COLOURS           : out NATURAL;
    AVAILABLE_COLOUR            : out COLOUR_AVAILABLE;
    NUMBER_OF_COLOUR_           : out NATURAL);
INDICES

procedure INQ_PREDEFINED_COLOUR_REPRESENTATION
    (TYPE_OF_WS                 : in WS_TYPE;
    INDEX                       : in COLOUR_INDEX;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    RGB_COLOUR                  : out COLOUR_REPRESENTATION);

procedure INQ_LIST_OF_AVAILABLE_GDP
    (TYPE_OF_WS                 : in WS_TYPE;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    LIST_OF_GDP                 : out GDP_IDS LIST_OF);

procedure INQ_GDP
    (TYPE_OF_WS                 : in WS_TYPE;
    GDP                         : in GDP_ID;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    LIST_OF_ATTRIBUTES_USED     : out ATTRIBUTES_USED.LIST_OF);

procedure ING_MAX_LENGTH_OF_WS_STATE_TABLES
    (TYPE_OF_WS                 : in WS_TYPE;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    MAX_POLYLINE_ENTRIES        : out NATURAL;
    MAX_POLYMARKER_ENTRIES      : out NATURAL;
    MAX_TEXT_ENTRIES            : out NATURAL;
    MAX_FILL_AREA_ENTRIES       : out NATURAL;
    MAX_PATTERN_INDICES         : out NATURAL;
    MAX_COLOUR_INDICES          . out NATURAL);

procedure INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED
    (TYPE_OF_WS                 : in WS_TYPE;
    ERROR_INDICATOR             : out ERROR_NUMBER;
    NUMBER_OF_PRIORITIES        : out NATURAL);
```

```
procedure INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES
    (TYPE_OF_WS                    : in WS_TYPE;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    TRANSFORMATION                 : out DYNAMIC_MODIFICATION;
    VISIBLE_TO_INVISIBLE           : out DYNAMIC_MODIFICATION;
    INVISIBLE_TO_VISIBLE           : out DYNAMIC_MODIFICATION;
    HIGHLIGHTING                   : out DYNAMIC_MODIFICATION;
    PRIORITY                       : out DYNAMIC_MODIFICATION;
    ADDING_PRIMITIVES              : out DYNAMIC_MODIFICATION;
    DELETION_VISIBLE               : out DYNAMIC_MODIFICATION);

procedure INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES
    (TYPE_OF_WS                    : in WS_TYPE;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    LOCATOR                        : out NATURAL;
    STROKE                         : out NATURAL;
    VALUATOR                       : out NATURAL;
    CHOICE                         : out NATURAL;
    PICK                           : out NATURAL;
    STRING                         : out NATURAL);

procedure INQ_DEFAULT_LOCATOR_DEVICE_DATA
    (TYPE_OF_WS                    : in WS_TYPE;
    DEVICE                         : in LOCATOR_DEVICE_NUMBER;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    INITIAL_POSITION               : out WC.POINT;
    LIST_OF_PROMPT_ECHO_
    _TYPES_                        : out LOCATOR_PROMPT_ECHO_
                                         TYPES.LIST_OF;
    ECHO_AREA                      : out DC.RESTANGLE_LIMITS;
    DATA_RECORD                    : out LOCATOR_DATA_RECORD);

procedure INQ_DEFAULT_STROKE_DEVICE_DATA
    (TYPE_OF_WS                    : in WS_TYPE;
    DEVICE                         : in STROKE_DEVICE_NUMBER;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    MAX_BUFFER_SIZE                : out NATURAL;
    LIST_OF_PROMPT_ECHO_
    _TYPES_                        : out STROKE_PROMPT_ECHO_
                                         TYPES.LIST_OF;
    ECHO_AREA                      : out DC.RESTANGLE_LIMITS;
    DATA_RECORD                    : out STROKE_DATA_RECORD);

procedure INQ_DEFAULT_VALUATOR_DEVICE_DATA
    (TYPE_OF_WS                    : in WS_TYPE;
    DEVICE                         : in VALUATOR_DEVICE_NUMBER;
    ERROR_INDICATOR                : out ERROR_NUMBER;
  INITIAL_VALUE                    : out VALUATOR_INPUT_VALUE;
    LIST_OF_PROMPT_ECHO_
    _TYPES_                        : out VALUATOR_PROMPT_ECHO_
                                         TYPES.LIST_OF;
    ECHO_AREA                      : out DC.RECTANGLE_LIMITS;
    DATA_RECORD                    : out VALUATOR_DATA_RECORD);

procedure INQ_DEFAULT_CHOICE_DEVICE_DATA
    (TYPE_OF_WS                    : in WS_TYPE;
    DEVICE                         : in CHOICE_DEVICE_NUMBER;
    ERROR_INDICATOR                : out ERROR_NUMBER;
    MAX_CHOICES                    : out CHOICE_VALUE;
```

```
    LIST_OF_PROMPT_ECHO_          : out CHOICE_PROMPT_ECHO_
    _TYPES_                                    _TYPES.LIST_OF;
    ECHO_AREA                     : out DC.RESTANGLE_LIMITS;
    DATA_RECORD                   : out CHOICE_DATA_RECORD);

procedure INQ_DEFAULT_PICK_DEVICE_DATA
    (TYPE_OF_WS                   : in WS_TYPE;
    DEVICE                        : in PICK_DEVICE_NUMBER;
    ERROR_INDICATOR               : out ERROR_NUMBER;
    LIST_OF_PROMPT_ECHO_          : out PICK_PROMPT_ECHO_
    _TYPES_                                    _TYPES.LIST_OF;
    ECHO_AREA                     : out DC.RECTANGLE_LIMITS;
    DATA_RECORD                   : out PICK_DATA_RECORD);

procedure INQ_DEFAULT_STRING_DEVICE_DATA
    (TYPE_OF_WS                   : in WS_TYPE;
    DEVICE                        : in STRING_DEVICE_NUMBER;
    ERROR_INDICATOR               : out ERROR_NUMBER;
    MAX_STRING_BUFFER_SIZE        : out NATURAL;
    LIST_OF_PROMPT_ECHO_          : out STRING_PROMPT_ECHO_
    _TYPES_                                    _TYPES.LIST_OF;
    ECHO_AREA                     : out DC_RECTANGLE_LIMITS;
    DATA_RECORD                   : out STRING_DATA_RECORD);

procedure INQ_SET_OF_ASSOCIATED_WS
    (SEGMENT                      : in SEGMENT_NAME;
    ERROR_INDICATOR               : out ERROR_NUMBER;
    LIST_OF_WS                    : out WS_IDS.LIST_OF);

procedure INQ_SEGMENT_ATTRIBUTES
    (SEGMENT                      : in SEGMENT_NAME;
    ERROR_INDICATOR               : out ERROR_NUMBER;
    TRANSFORMATION                : out TRANSFORMATION_MATRIX;
    VISIBILITY                    : out SEGMENT_VISIBILITY;
    HIGHLIGHTING                  : out SEGMENT_HIGHLIGHTING;
    PRIORITY                      : out SEGMENT_PRIORITY;
    DETECTABILITY                 : out SEGMENT_DETECTABILITY);

procedure INQ_PIXEL_ARRAY_DIMENSIONS
    (WS                           : in WS_ID;
    CORNER_1_1                    : in WC.POINT;
    CORNER_DX_DY                  : in WC.POINT;
    ERROR_INDICATOR               : out ERROR_NUMBER;
    DIMENSIONS                    : out  RASTER_UNIT_SIZE);

procedure INQ_PIXEL_ARRAY
    (WS                           : in WS_ID;
    CORNER                        : in WC.POINT;
    DX                            : in RASTER_UNITS;
    DY                            : in RASTER_UNITS;
    ERROR_INDICATOR               : out ERROR_NUMBER;
    INVALID_VALUES                : out INVALID_VALUES_INDICATOR;
    PIXEL_ARRAY                   : out VARIABLE_PIXEL_COLOUR_
                                                    MATRIX);

procedure INQ_PIXEL
    (WS                           : in WS_ID;
    POINT                         : in WC.POINT;
```

```
    ERROR_INDICATOR                 : out ERROR_NUMBER;
    PIXEL_COLOUR                    : out PIXEL_COLOUR_INDEX);
procedure INQ_INPUT_QUEUE_OVERFLOW
    (ERROR_INDICATOR                : out ERROR_NUMBER;
    WS                              : out WC_ID;
    CLASS                           : out INPUT_QUEUE_CLASS;
    DEVICE                          : out EVENT_OVERFLOW_DEVICE_
                                                    _NUMBER);
```

— ВСПОМОГАТЕЛЬНЫЕ ФУНКЦИИ

```
procedure EVALUATE_TRANSFORMATION_MATRIX
    (FIXED_POINT                    : in WC.POINT;
    SHIFT_VECTOR                    : in WC.VECTOR;
    ROTATION_ANGLE                  : in RADIANS;
    SCALE_FACTORS                   : in TRANSFORMATION_FACTOR;
    TRANSFORMATION                  : in TRANSFORMATION_MATRIX);
procedure EVALUATE_TRANSFORMATION_MATRIX
    (FIXED_POINT                    : in NDC.POINT;
    SHIFT_VECTOR                    : in NDC.VECTOR;
    ROTATION_ANGLE                  : in RADIANS;
    SCALE_FACTORS                   : in TRANSFORMATION_FACTOR;
    TRANSFORMATION                  : out TRANSFORMATION_MATRIX);
procedure ACCUMULATE_TRANSFORMATION_MATRIX
    (SOURCE_TRANSFORMATION          : in TRANSFORMATION_MATRIX);
    FIXED_POINT                     : in WC.POINT;
    SHIFT_VECTOR                    : in WC.VECTOR;
    ROTATION_ANGLE                  : in RADIANS;
    SCALE_FACTORS                   : in TRANSFORMATION_FACTOR;
    RESULT_TRANSFORMATION           : out TRANSFORMATION_MATRIX);
procedure ACCUMULATE_TRANSFORMATION_MATRIX
    (SOURCE_TRANSFORMATION          : in TRANSFORMATION_MATRIX);
    FIXED_POINT                     : in NDC.POINT;
    SHIFT_VECTOR                    : in NDC.VECTOR;
    ROTATION_ANGLE                  : in RADIANS;
    SCALE_FACTORS                   : in TRANSFORMATION_FACTOR;
    RESULT_TRANSFORMATION           : out TRANSFORMATION_MATRIX);
```

— ФУНКЦИИ ОБРАБОТКИ ОШИБОК

```
procedure ERROR_LOGGING
    (ERROR_INDICATOR                : in ERROR_NUMBER;
    GKS_FUNCTION                    : in STRING;
    ERROR_FILE                      : in STRING := DEFAULT_ERROR_
                                                    _FILE);
procedure EMERGENCY_CLOSE_GKS;
```

— Утилиты функций метафайла

— Элементы записей метафайла могут содержать списки указателей, строки символов, матрицы индексов цветов, данные GDP и ESC. Длина записи зависит от числа элементов данных. ЯГС определяет, что формат зависит от реализации.

— Тип элемента записи данных должен быть личным, позволяющим манипулировать содержимым записей, чтобы повысить эффективность их обработки.

— Прикладной программист должен быть способен записывать не графические данные в метафайл. Этого можно достигнуть, разрешив вывод литер. Числен-

ные данные должны быть преобразованы в строки прикладным программистом до вызова процедуры BUILD_NEW_GKSM_DATA_RECORD.

```
procedure BUILD_NEW_GKSM_DATA_RECORD
    (TYPE_OF_ITEM                    : in GKSM_ITEM_TYPE;
    ITEM_DATA                        : in STRING;
    ITEM                             : out GKSM_DATA_RECORD);

function ITEM_DATA_RECORD_STRING
    (ITEM : in GKSM_DATA_RECORD) return STRING;
private
— Следующие типы определяют спецификации для личных записей данных.
type GKSM_DATA_RECORD                : GKSM_ITEM_TYPE :=0;
    (TYPE_OF_ITEM
LENGTH                               : NATURAL        :=0) is
    record
      null;
    end record;

type CHOICE_DATA_RECORD (PROMPT_ECHO_TYPE:
                    CHOICE_PROMPT_ECHO_TYPE : = DEFAULT_CHOICE is
    record
      null;
    end record;

type LOCATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
                LOCATOR_PROMPT_ECHO_TYPE : = DEFAULT_LOCATOR) is
    record
      null;
    end record;

type STRING_DATA_RECORD (PROMPT_ECHO_TYPE:
                    STRING_PROMPT_ECHO_TYPE : = DEFAULT_STRING) is
    record
      null;
    end record;

type STROKE_DATA_RECORD (PROMPT_ECHO_TYPE:
                    STROKE_PROMPT_ECHO_TYPE : = DEFAULT_STROKE) is
    record
      null;
    end record;

type VALUATOR_DATA_RECORD (PROMPT_ECHO_TYPE:
                VALUATOR_PROMPT_ECHO_TYPE :=DEFAULT_VALUATOR) is
    record
      null;
    end record;

type PICK_DATA_RECORD (PROMPT_ECHO_TYPE:
                        PICK_PROMPT_ECHO_TYPE :=DEFAULT_PICK) is
    record
      null;
    end record;
end GKS;
```

— ФУНКЦИЯ ОБРАБОТКИ ОШИБОК

— Функция обработки ошибок является отдельным библиотечным блоком и не компилируется как часть пакета ЯГС.

```
procedure ERROR_HANDLING
    (ERROR_INDICATOR                : in ERROR_NUMBER;
    GKS_FUNCTION                    : in STRING;
    ERROR_FILE                      : in STRING :=DEFAULT_ERROR_
                                                        FILE);
```

```
with GKS_TYPES;
use GKS_TYPES;
package GKS_GDP is
```

— Пакет ОПВ является отдельным библиотечным блоком и не компилируется как часть ЯГС.

ОПВ связывается с отдельными процедурами, реализованными для каждого обобщенного примитива вывода, каждая со своим собственным интерфейсом. Имена ОПВ и параметры регистрируют в Международном журнале графических записей ИСО, который ведется органом регистрации.

Каждая незарегистрированная процедура ОПВ, поддерживаемая реализацией, будут содержаться в отдельном библиотечном пакете. При этом используют следующие соглашения по наименованию:

```
    package GKS_UGDP_< имя процедуры ОПВ>is
    procedure GDP;
```

— Код на языке Ада для процедуры ОПВ;

— Единственным именем процедуры, используемым в пакете, будет ОПВ.

— Для того, чтобы поддержать возможность записывать не реализованные ОПВ в метафайл, могут быть привлечены зарегистрированные ОПВ в форме процедуры GENERALIZED_GDP, которая имеет спецификацию, показанную ниже:

```
type GDP_FLOAT is digits PRECISION;
type GDP_INTEGER_ARRAY is array (SMALL_NATURAL range <>)
        of INTEGER;
type GDP_FLOAT_ARRAY is array (SMALL_NATURAL range <>)
        of GDP FLOAT;
type GDP_STRING_ARRAY is array (SMALL_NATURAL range <>)
        of STRING (1..80);
type GDP_DATA_RECORD (NUM_OF_INTEGERS : SMALL_NATURAL : = 0;
                      NUM_OF_REALS    : SMALL_NATURAL : = 0;
                      NUM_OF_STRINGS : SMALL_NATURAL : = 0) is
record
    INTEGER_ARRAY : GDP_INTEGER_ARRAY (1..NUM_OF_INTEGERS);
    REAL_ARRAY : GDP_FLOAT_ARRAY (1..NUM_OF_REALS);
    GDP_STRINGS : GDP_STRING_ARRAY (1..NUM_OF_STRINGS);
end record;
procedure GENERALIZED_GDP           : in GDP_ID;
    (GDP_NAME
                          POINT     : in WC.POINT_LIST;
                        GDP-DATA    : out GDP_DATA_RECORD);
end GKS_GDP;
with GKS_TYPES;
use GKS_TUPES;
package GKS_ESCAPE is
```

— Пакет ESCAPE является отдельным библиотечным блоком и не компилируется как часть ЯГС.

— Функции расширения связываются в Аде как отдельные процедуры для каждого уникального типа расширения, предоставляемого реализацией, каждая со списком формальных параметров, соответствующим реализованной процедуре. Имена ESCAPE и параметры регистрируют в Международном журнале графических записей ИСО, который ведется органом регистрации.

— Каждая незарегистрированная процедура ESCAPE будет отдельным библиотечным пакетом, использующим следующие соглашения по наименованию:

package GKS_UESC_<имя процедуры ESCAPE> is
procedure ESC;
—— Код на языке Ада процедуры UESC;
end GKS_UESC <имя процедуры ESCAPE>;

—— Единственным именем процедуры, используемым в пакете, является ESC.

—— Для того чтобы поддержать возможность записывать не реализованные ESCAPE в метафайл могут быть привлечены зарегистрированные процедуры ESCAPES в форме процедуры GENERALIZED_ESC, которая имеет спецификации, показанные ниже:

```
type ESCAPE_ID in new INTEGER;
type ESCAPE_FLOAT is digits PRECISION;
type ESC_INTEGER_ARRAY is array (SMALL_NATURAL range <>)
      of INTEGER;
type ESC_FLOAT_ARRAY is array (SMALL_NATURAL range <>)
      of ESCAPE_FLOAT;
type ESC_STRING_ARRAY is array (SMALL_NATURAL range <>)
      of STRING (1..80);

type ESC_DATA_RECORD                : SMALL_NATURAL :=0;
   (NUM_OF_INTEGERS
                NUM_OF_REALS    : SMALL_NATURAL :=0;
              NUM_OF_STRINGS    : SMALL_NATURAL :=0) is

record
  INTEGER_ARRAY : ESC_INTEGER_ARRAY   (1..NUM_OF_INTEGERS);
  REAL_ARRAY    : ESC_FLOAT_ARRAY     (1..NUM_OF_REALS);
  ESC_STRING    : ESC_STRING_ARRAY    (1..NUM_OF_STRINGS);

end record;
procedure GENERALIZED_ESC              : in ESCAPE_ID;
   (ESCAPE_NAM
                ESC_DATA_IN       : in ESC_DATA_RECORD;
               ESC_DATA_OUT       : out ESC_DATA_RECORD;

end GKS_ESCAPE;
```

ПРИЛОЖЕНИЕ Б

СПИСОК ССЫЛОК НА ОПРЕДЕЛЕННЫЕ РЕАЛИЗАЦИЕЙ ЗАПИСИ

(Это приложение не является составной частью стандарта, но дает дополнительную информацию)

Элемент	Пункт
CHOICE_DATA_RECORD	4 2 3
LOCATOR_DATA_RECORD	4 2 3
PICK_DATA_RECORD	4 2 3
STRING_DATA_RECORD	4 2 3
STROKE_DATA_RECORD	4 2 3
VALUATOR_DATA_RECORD	4 2 3
DEFAULT_MEMORY_UNITS	4 2 4
DEFAULT_ERROR_FILE	4 2 4
PRECISION	4 2 4
MAX_LIST_SIZE	5 2 3
SMALL_NATURAL_MAX	4 2 4
CHOICE_SMALL_NATURAL_MAX	4 2 4
STRING_SMALL_NATURAL_MAX	4 2 4

ПРИЛОЖЕНИЕ В

ПРИМЕРЫ ПРОГРАММ

(Это приложение не является составной частью стандарта, но дает дополнительную информацию).

В настоящем приложении даны программы, использующие связь, определенную в стандарте

В 1. Пример программы 1: STAR

```
— ПРОГРАММА STAR
—
— ОПИСАНИЕ:
— Эта программа рисует желтую звезду на голубом фоне и пишет зеленым цветом STAR под звездой.
— СОГЛАШЕНИЯ:
— Уровень ЯГС 0a
— Реализация должна поддерживать, по крайней мере, работу одной станции категории вывода или ввода/вывода.

with GKS;
with GKS_TYPES;

use GKS;
use GKS_TYPES;

procedure STAR is

   — Определяет переменные станции и файл регистрации ошибок.

   MY_WS_ID              : constant WS_ID          := 1;
   SOME_CONNECTION       : constant STRING         := «UNIT_1»
   SOME_OUTPUT_TYPE      : constant WS_TYPE        := 1;
   ERROR_FILE            : constant STRING
                         := «MY_ERROR_FILE»;

— Определяет точки звезды.
STAR_POINTS : constant WS.POINT_ARRAY := (
                        (0.951057, 0.309017),
                        (—0.951057, 0.309017),
                        (0.587785, —0.951057),
                        (0.0, 1.0),
                        (—0.587785, — 0.951057));

— Определяет окно мировой системы и различные атрибуты.
WINDOW:WC.RECTANGLE_LIMITS: =
                        (XMIN=>—1.25, XMAX=>1.25,
                         YMIN=>—1.25, YMAX=>1.25);

TEXT_POSITION.WC.POINT:=(0.0, —1.0);
begin

— Открыть ЯГС и активизировать станцию.
OPEN_GKS (ERROR_FILE);
OPEN_WS   (MY_WS_ID.SOME_CONNECTION.SOME_OUTPUT_TYPE);
ACTIVATE_WS (MY_WS_ID);
```

```
— Центрировать окно вокруг начала координат.
SET WINDOW (1, WINDOW);
SELECT_NORMALIZATION_TRANSFORMATION (1);

— Задать цвета.
SET_COLOUR_REPRESENTATION (MY_WS_ID,
                           INDEX=>0,
                           RGB_COLOUR=>(0.0, 0.0, 1.0));
SET_COLOUR_REPRESENTATION (MY_WS-ID,
                           INDEX=>1,
                           RGB_COLOUR=>(1.0, 1 0, 0 0));
SET_COLOUR-REPRESENTATION (MY_WS_ID, .
                           INDEX=>2,
                           RGB_COLOUR=>(1.0, 1.0, 1.0));

— Установить атрибуты области заполнения.
SET_FILE_AREA_INTERIOR_STYLE (SOLID);
SET_FILE_AREA_COLOUR_INDEX (1);

— Нарисовать звезду.
FILL_AREA (STAR_POINTS);

— Выбрать заглавные буквы, позиционированные под звездой.
SET_CHAR_HEIGHT              (HEIGHT=>0 15);
SET_TEAX_ALIGNMENT           (ALIGNMENT=>(CENTRE, HALF));
SET_TEXT_COLOUR_INDEX        (TEXT_COLOUR=>2);

— Нарисовать заголовок.
TEXT (TEXT_POSITION, «STAR»);

— Закрыть станцию и ЯГС.
DEACTIOVATE_WS (MY_WS_ID);
CLOSE_WS (MY_WS_ID),
CLOSE_GKS;

end STAR;
```

B.2. Пример программы 2: IRON

```
— ПРОГРАММА IRON
—
— ОПИСАНИЕ:
— Данная программа рисует горизонтальную гистограмму, иллюстрирующую
цены в металлургической промышленности.
— Пользователь может выбрать данные для отображения, используя устройство
выбора альтернативы ЯГС. График адаптирован из журнала «Scientic
American», май 1984 г., стр. 139.
— СОГЛАШЕНИЕ:
— Уровень ЯГС 2b
with GKS;
with GKS_TYPES;

use GKS;
use GKS_TYPES;

procedure IRON is
    — Задает переменные станции и файл регистрации ошибок.
    MY_WS_ID                : constant WS_ID        :=1;
```

```
SOME_CONNECTION      : constant STRING    :=«TTY»;
SOME_OUTIN_TYPE      : constant WS_TYPE   :=2;
ERROR_FILE           : constant STRING    := «MY_ERROR_FILE»;
```

— Описать и инициировать флаги выборки атрибутов (используют связки для области заполнения, в противном случае устанавливают индивидуальный).

```
ASF_SETTINGS:ASF_LIST: =
  (TYPE_OF_LINE_ASF          =>INDIVIDUAL,
  WIDTH_ASF                  =>INDIVIDUAL,
  LINE_COLOUR_ASF            =>INDIVIDUAL,
  TYPE_OF_MARKER_ASF         =>INDIVIDUAL,
  SIZE_ASF                   =>INDIVIDUAL,
  MARKER_COLOUR_ASF          =>INDIVIDUAL,
  FONT_PRECISION_ASF         =>INDIVIDUAL,
  EXPANSION_ASF              =>INDIVIDUAL,
  SPACING_ASF                =>INDIVIDUAL,
  TEXT_COLOUR_ASF            =>INDIVIDUAL,
  INTERIOR_ASF               =>BUNDLED,
  STYLE_ASF                  =>BUNDLED,
  FILL_AREA_COLOUR_ASF       =>INDIVIDUAL);
```

— Описать и инициировать объекты для устройства выбора альтернативы.

```
CHOICE_STRING_COUNT          : constant :=3;
CHOICE_DEVICE                : constant CHOICE_DEVICE_
                                        _NUMBER :=1;
CHOICE_ERROR                 : ERROR_NUMBER;
CHOICE_MODE                  : OPERATING_MODE;
CHOICE_ECHO_SWITCH           : ECHO_SWITCH;
INITIAL_CHOICE               : CHOICE_VALUE;
CHOICE_INPUT_RECORD          : CHOICE_DATE_RECORD;
CHOICE_ECHO_AREA             : DC.RECTANGLE_LIMITS;
PROMPT_ECHO_TYPE             : constant CHOICE_PROMPT_ECHO_
                                        TYPE :=3;
CHOICE_STRINGS               : constant CHOICE_PROMPT_
                                  STRING_LIST := (LENGTH=>3,
                                                  LIST =>
                                  ((4, «U. S.»), (9, «W.GERMANY»),
                                   (5, «JAPAN»)));
CHOICE_RECORD                : CHOICE_DATA_RECORD;
CHOICE_                      : CHOICE_VALUE;
CHOICE_REQUEST               : CHOICE_REQUEST_STATUS;
INITIAL_STATUS               : CHOICE_STATUS;
```

— Описать объекты цвета.

```
RGB_COLOUR                   : COLOUR_REPRESENTATION;
WHITE                        : constant COLOUR_INDEX :=0;
BLACK                        : constant COLOUR_INDEX :=1;
RED                          : constant COLOUR_INDEX :=2;
```

— Описать и инициировать объекты окна.

```
WINDOW_1                     : TRANSFORMATION_NUMBER :=1;
WINDOW_LIMITS                : WC.RECTANGLE_LIMITS :=
                               (XMIN=>—100.0, XMAX=>175.0,
                               YMIN=>—2.0, YMAX=>13.0);
```

— Описать и инициировать объекты данных гистограммы

```
MAX_DATA. constant: =6;
```

```
type IRON_DATA is array (1 . .MAX_DATA) of WS_TYPE;
US_DATA_1                          : IRON_DATA := (69.0, 50.0, 15.0,
                                                   53.0, 57.0, 150.0);
US_DATA_2                          : IRON_DATA := (72.0, 50.0, 103.0, 0.0,
                                                   0.0, 56.0);
GERMANY_DATA_1                     : IRON_DATA := (65.0, 42.0, 3.0, 89.0,
                                                   52.0, 93.0);
GERMANY_DATA_2                     : IRON_DATA := (70.0, 53.0, 102.0, 0.0,
                                                   0.0, 49.0);
JAPAN_DATA_1                       : IRON_DATA := (65.0, 47.0, 2.0, 60.0,
                                                   52.0, 55.0);
JAPAN_DATA_2                       : IRON_DATA := (70.0, 57.0, 105.0, 0.0,
                                                   0.0, 41.0);

procedure BARS (LENGTH             : WC_TYPE;
                POSITION           : WC.POINT) is
LEFT_HALF                          : TEXT_ALIGNMENT := (LEFT,
                                                        HALF);
BAR_POINTS                         : WC.POINT_ARRAY(1 . . 4);

begin if LENGTH = 0.0 then
        SET_TEXT_ALIGNMENT (LEFT_HALF);
        TEXT (POSITION, «0»);
    else
      BAR_POINTS := ((X=>0.0,      Y=>POSITION.Y+0.4),
                     (X=>LENGTH,   Y=>POSITION.Y+0.4),
                     (X=>LENGTH,   Y=>POSITION.Y—0.4),
                     (X=>0.0,      Y=>POSITION.Y—0.4));

        FILL_AREA(BAR_POINTS),
    end if;
end BARS;
procedure TICKS (TICK_MARK_POSITION : in out WC.POINT_ARRAY) is
TICK_MARK_LABEL_POSITION:WC.POINT;
begin
for I in 1 . . 4 loop
    POLYLINE (TICK_MARK_POSITION);
    TICK_MARK_POSITION (1).X:=TICK_MARK_POSITION(1).X+50.0;
    TICK_MARK_POSITION (2).X:=TICK_MARK_POSITION(2).X+50.0;
end loop;

— Нарисовать маскировочные метки.
TICK_MARK_LABEL_POSITION.X:=0.0;
TICK_MARK_LABEL_POSITION.Y:=WC_TYPE
                              (TICK_MARK_POSITION(1).Y);
TEXT (TICK_MARK_LABEL_POSITION, «0»);
TICK_MARK_LABEL_POSITION.X:=50.0;
TEXT (TICK_MARK_LABEL_POSITION, «50»);
TICK_MARK_LABEL_POSITION.X:=100.0;
TEXT (TICK_MARK_LABEL_POSITION, «100»);
TICK_MARK_LABEL_POSITION.X:=150.0;
TEXT (TICK_MARK_LABEL_POSITION, «150»);
end TICKS;
procedure BORDER is
```

```
— Нарисовать границу, окружающую данные.
LABEL_POSITION                          WC.POINT;
TITLE_POSITION                          WC.POINT  := (37.5, -2.9);
HEIGHT                                  WC.MAGNITUDE  := 0.5;
LEFT_HALF                               TEXT_ALIGNMENT  := (LEFT,
                                                            HALF);
CENTRE_BOTTOM                           : TEXT_ALIGNMENT  := (CENTRE,
                                                            BOTTOM);
CENTRE_CAP                              : TEXT_ALIGNMENT  := (CENTRE,
                                                            CAP);
ONLY_IF_NOT_EMPTY                       CONTROL_FLAG
                                                        := CONDITIONALLY;
BOX_POINTS                              : constant WC.POINT_ARRAY :=
                                    ((0.0, 0.0), (150.0, 0.0), (150.0, 12.0), (0.0, 12.0),
                                                                ((0.0, 0.0));
type LABELS is array (1..6) of INPUT_STRING;
BAR_LABELS : constant LABELS:=
                ((5, «LABOR»), (8, «IRON ORE»), (12, «COKE OR COAL»),
                 (15, «PURCHASED SCRAP»), (11, «OTHER COSTS»),
                 (12, «OTHER ENERGY»));
TOP_TICK_MARK_START : WC.POINT_ARRAY (1..2) := ((0.0, 12.0),
                                                        (0.0, 11.9));
BOTTOM_TICK_MARK_START : WC.POINT_ARRAY (1..2):= ((0.0, 0.0),
                                                        (0.0, 0.1));
begin
CLEAR_WS (MY_WS_ID, ONLY_IF_NOT_EMPTY);

— Нарисовать квадрат, ограниченный областью диаграммы.
POLYLINE (BOX_POINTS);

— Нарисовать метки гистограммы, центрированные по полоскам.
SET_TEXT_ALIGNMENT (LEFT_HALF);
SET_CHAR_HEIGHT (HEIGHT);
SET_TEXT_COLOUR_INDEX (BLACK);

LABEL_POSITION.X:= -99.0;
for I in 1..6 loop
    LABEL_POSITION.Y:=WC_TYPE(2.0*(FLOAT(I)-1.0)+1.2)
    TEXT (LABEL_POSITION, BAR_LABELS(INTEGER(I)).CONTENTS);
end loop;

— Нарисовать верхние и нижние черточки (красное основание).
SET_TEXT_ALIGNMENT (CENTRE_BOTTOM);

— Вызвать процедуры для рисования черточек.
TICKS (TOP_TICK_MARK_START);

SET_TEXT_ALIGNMENT (CENTRE_CAP);
SET_TEXT_COLOUR_INDEX(RED);
TICKS (BOTTOM_TICK_MARK_START);

— Нарисовать заголовок.
SET_TEXT_COLOUR_INDEX (BLACK);
SET_TEXT_ALIGNMENT (CENTRE_BOTTOM);
TEXT (TITLE_POSITION, «PRODUCTION COST»);
end BORDER;
```

```
procedure DRAW                          (DATA1 : in out IRON_DATA;
                                        DATA2 : in out IRON_DATA)is
FILL_INDEX                              : FILL_AREA_INDEX := 1;
POSITION                                : WC.POINT;
begin
— Нарисовать границу.
BORDER;
— Нарисовать черные полосы.
SET_FILL_AREA_COLOUR_INDEX (BLACK);
SET_TEXT_COLOUR_INDEX (BLACK);
SET_FILL_AREA_INDEX (FILL_INDEX);
for I in 1 . . 6 loop
    POSITION Y:=2.0*(WC_TYPE(1)—1.0)+1.6;
    — Вызвать процедуру, которая вычеркивает гистограммы.
    BARS(DATA1(INTEGER(1)), POSITION);
end loop;
— Вычеркивает красные гистограммы.
SET_FILL_AREA_COLOUR_INDEX(RED);
SET_TEXT_COLOUR_INDEX(RED);
FILL_INDEX:=2;
    SET_FILL_AREA_INDEX (FILL_INDEX);
    for I in 1 . . 6 loop
      POSSITION.Y:=2.0*(WS_TYPE(1)—1.0)+1.6;
    — Вызвать процедуру, которая вычерчивает гистограммы.
      BARS(DATA2(INTEGER(1), POSITION),
    end loop;
end DRAW;
begin
    — Открыть ЯГС и активировать станцию.
    OPEN_GKS(ERROR_FILE);
    OREN_WS(MY_WS_ID, SOME_CONNECTION.SOME_OUTIN_TYPE);
    ACTIVATE_WS(MY_WS_ID);
— Задать окно на диаграмме.
    SET_WINDOW (WINDOW_1, WINDOW_LIMITS);
    SELECT_NORMALIZATION_TRANSFORMATION (WINDOW_1);
— Задать цвета, которые будут использоваться
    SET_COLOUR_REPRESENTATION (MY_WS_ID,
                                        INDEX=>WHITE,
                                        RGB_COLOUR=>(1.0, 1.0, 1.0));
    SET_COLOUR_REPRESENTATION (MY_WS-ID,
                                        INDEX=>BLACK,
                                        RGB_COLOUR=>(0.0, 0.0, 0.0));
    SET_COLOUR_REPRESENTATION (MY_WS-ID,
                                        INDEX=>RED,
                                        RGB_COLOUR=>(1 0, 0 0, 0.0));
— Использовать связанные атрибуты, исключая цвет
    SET ASF (ASF_SETTINGS);
```

```
— Инициировать устройство выбора
   INQ_CHOICE_DEVICE_STATE (MY_WS_ID,
                            CHOICE_DEVICE,
                            CHOICE_ERROR,
                            CHOICE_MODE,
                            CHOICE_ECHO_SWITCH,
                            INITIAL_STATUS,
                            INITIAL_CHOICE,
                            CHOICE_ECHO_AREA,
                            CHOICE_RECORD);
   BUILD_CHOICE_DATA_RECORD (PROMPT_ECHO_TYPE,
                             CHOICE_STRINGS,
                             CHOICE_RECORD),
   INITIALISE_CHOICE (MY_WS_ID,
                      CHOICE_DEVICE,
                      INITIAL_STATUS,
                      INITIAL_CHOICE,
                      CHOICE_ECHO_AREA,
                      CHOICE_RECORD);
— Получить выбор пользователя (U. S., W. GERMANY, or JAPAN).
   loop
     REQUEST_CHOICE(MY_WS_ID,
                    CHOICE_DEVICE,
                    INITIAL_STATUS,
                    CHOICE);
     if INITIAL_STATUS=OK then
        case CHOICE is
          when 1 => DRAW (US_DATA_1, US_DATA_2);
          when 2 => DRAW (GERMANY_DATA_1, GERMANY_DATA_2);
          when 3 => DRAW (JAPAN_DATA_1, JAPAN_DATA_2);
          when others => exit;
        end case;
     else
        exit;
     end if;
   end loop;
— Закрыть станцию и ЯГС.
   DEACTIVATE_WS                (MY_WS_ID),
   CLOSE_WS                     (MY_WS_ID),
   CLOSE_GKS,
end IRON;
```

В. 3. Пример программы 3: MAP

```
— ПРОГРАММА MAP
—
— ОПИСАНИЕ:
— Эта программа считывает метафайл ЯГС для вычерчивания карты. Примитивы каждой области заключены в отдельных сегментах. Пользователь может применить для указания области устройство выбора. Опрашиваемое устройство выбора задает операции с выбранными примитивами области.
— СОГЛАШЕНИЕ:
— Уровень ЯГС 1c
```

— Реализация должна поддерживать, по крайней мере, одну станцию категории ввод/вывод и одну категорию метафайла ввода. Устройство выбора по умолчанию должно поддерживать, по крайней мере, пять альтернатив.

```
with GKS;
with GKS_TYPE;
use GKS;
use GKS_TYPES;
procedure METAFILE is
— Задать станцию метафайла.
   METAFILE_WS_ID                : constant WS_ID :=1;
   METAFILE_CONNECTION           : constant STRING :=«METAFILE_
                                                _INPUT_FILE»;
   METAFILE_TYPE                 : constant WS_TYPE :=2;
   METAFILE_ITEM_TYPE            : GKSM_ITEM_TYPE;
   METAFILE_DATA_RECORD          : GKSM_DATA_RECORD;
   LENGTH, MAX_LENGTH            : NATURAL :=500;
— Задать станцию ввода/вывода.
   MY_WS_ID                      : constant WS_ID :=2;
   SOME_CONNECTION               : constant STRING :=«UNIT_2»;
   SOME_OUTIN_TYPE               : constant WS_TYPE :=1;
   SOME_CHOICE_DEVICE            : constant CHOICE_DEVICE_
                                                _NUMBER :=1;
   CSTATUS                       : CHOICE_STATUS;
   CHOICE_NUMBER                 : CHOICE_VALUE;
   SOME_PICK_DEVICE              : constant PICK_DEVICE_NUMBER
                                                :=1;
   PSTATUS                       : PICK_REQUEST_STATUS;
   PICK                          : PICK_ID;
   SEGMENT                       : SEGMENT_NAME;
— Задать файл регистрации ошибок.
   ERROR_FILE                    : constant STRING := «MY_ERROR_
                                                _FILE»;
begin
— Открыть ЯГС и активизировать станции.
OPEN_GKS(ERROR_FILE);
OPEN_WS(METAFILE_WS_ID, METAFILE_CONNECTION, METAFILE_TYPE);
OPEN_WS(MY_WS_ID, SOME_CONNECTION, SOME_OUTIN_TYPE);
ACTIVATE_WS(MY_WS_ID);
— Установить устройство выбора в режим выборки.
SET_CHOICE_MODE (MY_WS_ID, SOME_CHOICE_DEVICE,
                 SAMPLE_MODE, NOECHO);
— Интерпретировать элементы метафайла до считывания конца метафайла.
loop
   GET_ITEM_TYPE_FROM_GKSM (METAFILE_WS_ID,
                            METAFILE_ITEM_TYPE, LENGTH);
   if METAFILE_ITEM_TYPE=0 then
     exit;
   end if;
   READ_ITEM_FROM_GKSM (METAFILE_WS_ID, MAX_LENGTH,
METAFILE_DATA_RECORD);
```

```
    INTERPRENT_ITEM (METAFILE_DATA_RECORD);
end loop;
— Закрыть станцию метафайла.
    CLOSE_WS (METAFILE_WS_ID);
— Позволить пользователю выбирать состояния до выбора EXIT.
loop
    REQUEST_PICK
    (MY_WS_ID, SOME_PICK_DEVICE, PSTATUS, SEGMENT, PICK);
    if PSTATUS=OK then
      SAMPLE_CHOICE  (MY_WS-ID,  SOME_CHOICE_DEVICE,  CSTATUS,
                      CHOICE_NUMBER);
    if CSTATUS=OK then
      case CHOICE_NUMBER is
      when 1=>SET_HIGHLIGHTING  (SEGMENT, HIGHLIGHTED);
      when 2=>SET_HIGHLIGHTING (SEGMENT, NORMAL);
      when 3=>SET_VISIBILITY (SEGMENT, INVISIBLE);
      when 4=>SET_VISIBILITY (SEGMENT, VICIBLE)
      when others=>null;
    end case;
  end if;
end if;
end loop;
    — Закрыть станцию и ЯГС.
    DEACTIVATE_WS   (MY_WS_ID);
    CLOSE_WS        (MY_WS_ID);
  CLOSE_GKS;
end METAFILE;
```

В. 4. Пример программы 4: MANIPULATE

```
— ПРОГРАММА MANIPULATE
—
— ОПИСАНИЕ:
— Данная программа позволяет пользователю создать объект и затем манипу-
лировать им, изменяя преобразование сегмента.
— СОГЛАШЕНИЯ:
— Уровень ЯГС: 2b
with GKS;
with GKS_TYPES;
use GKS;
use GKS_TYPES;
procedure POLYGON is
    POINTS                  : WC.POINT_ARRAY(1..500);
    POINT_1                 : WC.POINT := (0.6, 0.4);
    POINT_2                 : WC.POINT := (0.4, 0.3);
    POLIGON_SEGMENT         : SEGMENT_NAME := 1;
    SHIFT                   : constant CHOICE_VALUE := 1;
    ZOOM                    : constant CHOICE_VALUE := 2;
    ROTATE                  : constant CHOICE_VALUE := 3;
    NEXT                    : POSITIVE := 1
    TRANSFORMATION          : TRANSFORMATION_NUMBER;
    TRANSFORMATION_1        : TRANSFORMATION_NUMBER;
    TRANSFORMATION_2        : TRANSFORMATION_NUMBER;
```

```
    RED                          : constant COLOUR_INDEX :=2;
    AXIC_CHARACTER_HEIGHT        : constant WC.MAGNITUDE .=0.02;
    CHOICE                       : CHOICE_VALUE;
    CHOICE_STATUS                : CHOICE_REQUEST_STATUS;
    LOCATOR_STATUS               : INPUT_STATUS;
    MATRIX                       : TRANSFORMATION_MATRIX;
    MATRIX_RESULT                : TRANSFORMATION_MATRIX;
— Задать переменные станции и файл регистрации ошибок.
DISPLAY              : constant WS_ID      : =1;
DISPLAY_CONNECTION   : constant STRING     : =«DDDIS»;
DISPLAY_TYPE         : constant WS_TYPE    : =3;
ERROR_FILE           : constant STRING     : =«MY_ERROR_FILE»,
— Задать переменные сегмента станции.
SEGSTORE             : constant WS_ID      : =2;
SEG_CONNECTION       : constant STRING     : =«DDSEG»;
SEG_TYPE             : constant WS_TYPE    : =4;
— Задать переменные станции графопостроителя.
PLOTTER              : constant WS_ID      : =6;
PLOT_CONNECTION      : constant STRING     : =«PLOT»;
PLOT_TYPE            : constant WS_TYPE    : =5;
— Задайте окно мировой системы координат и другие атрибуты.
WINDOW_BOUNDS                : WC.RECTANGLE_LIMITS :=
                                   (XMIN=>0.0, XMAX=>1.0,
                                   YMIN=>0.0, YMAX=>1.0);
VIEWPORT_BOUNDS              : NDC.RECTANGLE_LIMITS :=
                                   (XMIN=>0.0, XMAX=>1.0,
                                   YMIN=>0.0, YMAX=>1.0);
TEXT_POSITION                : WC.POINT :=(0.5, 0.5);
begin
— Открыть ЯГС и активировать станцию.
OPEN_GKS (ERROR_FILE);
OPEN_WS (DISPLAY, DISPLAY_CONNECTION, DISPLAY_TYPE);
ACTIVATE_WS (DISPLAY);
OPEN_WS (SEGSTORE, SEG_CONNECTION, SEG_TYPE);
ACTIVATE_WS (SEGSTORE);
SET_WINDOW (1, WINDOW_BOUNDS);
SET_VIEWPORT (1, VIEWPORT_BOUNDS);
SET_VIEWPORT_INPUT_PRIORITY (1, 0, HIGHER);
— Построение сегмента POLYGON_SEGMENT
CREATE_SEGMENT (POLYGON_SEGMENT);
SET_POLYLINE_INDEX(3);
REQUEST_LOCATOR(DISPLAY, LOCATOR_STATUS,
                TRANSFORMATION, POINTS (NEXT));
SELECT_NORMALIZATION_TRANSFORMATION (TRANSFORMATION, -1);
loop
    NEXT:=NEXT+1;
    REQUEST_LOCATOR (DISPLAY, 1, LOCATOR_STATUS,
    TRANSFORMATION, POINTS (NEXT));
```

```
    exit when LOCATOR_STATUS=NONE or
        TRANSFORMATION/=TRANSFORMATION_1 or
        NEXT=500;
end loop;
POINTS (NEXT):=POINTS(1);
POLYLINE(POINTS);
CLOSE_SEGMENT;
EVALUATE_TRANSFORMATION_MATRIX (WC.POINT((0.0, 0.0)),
                                WC. VECTOR' ((0.0, 0.0)), 0.0, (1.0, 1.0), MATRIX;
— инициализировать матрицу преобразования
loop
    REQUEST_CHOICE (DISPLAY, 1, CHOICE_STATUS, CHOICE);
    exit when CHOICE_STATUS=NONE or CHOICE_STATUS=NOCHOICE;
case CHOICE is
    —— сдвинуть многоугольник в данную позицию
      when SHIFT=>
        REQUEST_LOCATOR (DISPLAY, 1; LOCATOR_STATUS,
                         TRANSFORMATION_2, POINT_1);
    exit when LOCATOR_STATUS=NONE;
        REQUEST_LOCATOR (DISPLAY, 1, LOCATOR_STATUS,
                         TRANSFORMATION, POINT_2);
    exit when LOCATOR_STATUS=NONE or
                        TRANSFORMATION/=TRANSFORMATION_2);
    SELECT_NORMALIZATION_TRANSFORMATION (TRANSFORMATION_2);
    ACCUMULATE_TRANSFORMATION_MATRIX(
      SOURCE_TRANSFORMATION      =>MATRIX,
      FIXED_POINT                =>WC.POINT'((0.0, 0.0)),
      SHIFT_VECTOR               =>WC.VECTOR'(
                                    (POINT_1.X—POINT_2.X,
                                    POINT_1.Y—POINT_2.Y)),
      ROTATION_ANGLE'            =>0.0,
      SCALE_FACTORS              =>(1.0, 1.0),
      RESULT_TRANSFORMATION      =>MATRIX_RESULT);
    SET_SEGMENT_TRANSFORMATION
    (POLYGON_SEGMENT, MATRIX_RESULT);
      when ZOOM                  =>null;
      when ROTATE                =>null;
      when others                =>exit;
    end case;
    UPDATE_WS(DISPLAY, PERFORM);
end loop;
— теперь полигон прорисован
DEACTIVATE_WS(DISPLAY);
DEACTIVATE_WS(SEGSTORE);
OPEN_WS(PLOTTER, PLOT_CONNECTION, PLOT_TYPE);
ACTIVATE_WS(PLOTTER);
— установить представления для этой станции
SET_COLOUR_REPRESENTATION (PLOTTER, RED, (1.0, 0.0, 0.0));
```

```
SET_POLYLINE_REPRESENTATION (PLOTTER, 3, 1, 1.5, RED);
SET_TEXT_REPRESENTATION (PLOTTER, 2, (0, STRING_PRECISION),
                                                    1.0, 0.0, RED);
SET_WS_VIEWPORT (PLOTTER, (0.0, 0.0, 0.5));
COPY_SEGMENT_TO_WS (PLOTTER, POLYGON_SEGMENT);
SET_TEXT_INDEX (2);
SET_CHAR_HEIGHT (AXIS_CHARACTER_HEIGHT);
TEXT ((0.5, 0.5), «This is polygon»);

    DEACTIVATE_WS (PLOTTER);
    CLOSE_WS (PLOTTER);
    CLOSE_WS (DISPLAY);
    CLOSE_WS (SEGSTORE);
    CLOSE_GKS;
end POLYGON;
```

В.5. Пример программы 5. PROGRAM SHWLN

```
— ПРОГРАММА SHOWLN
—
— ОПИСАНИЕ:
— Данная программа иллюстрирует существующие типы линий на выбранной
пользователем станции. Она содержит типичную подпрограмму инициализации
ЯГС и демонстрирует подпрограммы, которые не меняют каких-либо элементов
списка состояний.
— СОГЛАШЕНИЯ:
— Уровень ЯГС 0a

with GKS_TYPES;
with GKS;
with TEXT_IO;
use GKS_TYPES;
use GKS;
use TEXT_IO;
procedure SHOWLN is

TYPE_OF_WS                      : WS_TYPE;
ERROR_IND                       : ERROR_NUMBER :=0;
WORKSTATION                     : WS_ID :=1;
OP_STATE                        : OPERATING_STATE;
package WS_TYPE_IO is new INTEGER_IO (WS_TYPE);

procedure INIT_GKS (WTYPE : in out WS_TYPE;
        ERRIND : in out ERROR_NUMBER) is

    — Последовательность инициализации ЯГС.

    ERROR_FILE : constant STRING : = 'SHOWLN_ERR_FILE';
    GKS_WS_TYPES                : WS_TYPES.LIST_OF;
    CATEGORY                    : WS_CATEGORY;
    CONNECTION                  : STRING (1 .. 20);
    CONN_LENGTH                 : NATURAL;
begin
    OPEN_GKS (ERROR_FILE);

    — Запросить доступные типы станций и распечатать их.

    INQ_LIST_OF_AVAILABLE_WS_TYPES (ERRIND, GKS_WS_TYPES);
    if ERRIND /=0 then
```

```
      return;
    end if;
    PUT_LINE («The available output and outin workstation types are:»);
    for I in 1 .. WS_TYPES.SIZE_OF_LIST (GKS_WS_TYPES) loop
      INQ_WS_CATEGORY
    (WS_TYPES.LIST_ELEMENT (I, GKS_WS_TYPES), ERRIND, CATEGORY);
    if (CATEGORY = OUTPUT or CATEGORY =OUTIN) then
      WS_TYPE_IO.PUT (WS_TYPES.LIST_ELEMENT (I, GKS_WS_TYPES));
      PUT ("");
    end if;
  end loop;
  NEW_LINE;
```

— Выберите одну станцию, чтобы открыть и активировать ее.

```
    PUT_LINE («Please enter connection identifier and workstation type»)
    GET_LINE (CONNECTION, CONN_LENGTH);
    WS_TYPE_IO.GET (WTYPE);
    OPEN_WS (WORKSTATION, CONNECTION (1 .. CONN_LENGTH),
                                                        WTYPE);
    ACTIVATE_WS (WORKSTATION);
```

— Проверьте режим работы для гарантирования успешных открытий и активирования.

```
    INQ_OPERATING_STATE_VALUE (OP_STATE);
    if OP_STATE /=WSAC then
      ERRIND :=3;
      return;
    end if;

    ERRIND :=0;
  end INIT_GKS;
  procedure LINE_DEMO (WTYPE          : in out WS_TYPE;
                       ERRIND         : in out ERROR_NUMER) is
    STATUS                          : UPDATE_STATE;
    REQ_WINDOW                      : NDC.RECTANGLE_LIMITS;
    CUR_WINDOW                      : NDC.RECTANGLE_LIMITS;
    REQ_VIEWPORT                    : DC.RECTANGLE_LIMITS;
    CUR_VIEWPORT                    : DC RECTANGLE_LIMITS;
    LINETYPE_LIST                   : LINETYPES.LIST_OF;
    NUM_WIDTHS                      . NATURAL;
    NOMINAL_WIDTH                   ; DC.MAGNITUDE;
    RANGE_OF_WIDTHS                 : DC.RANGE_OF_MAGNITUTES;
    NUM_INDICES                     : NATURAL;
    LIST_OF_ASF                     : ASF_LIST := (others
                                                  =>INDIVIDUAL);
    SAVED_XFORM_NUM                 . TRANSFORMATION_NUMBER;
    SAVED_PRIM_ATTR                 : PRIMITIVE_ATTRIBUTE_VALUES;
    SAVED_INDV_ATTR                 : INDIVIDUAL_ATTRIBUTE_VALUES;
    DISTANCE                        : NDC_TYPE;
    PTS                             : WC.POINT_ARRAY ! 1 .. 2);
  begin
```

— Проверить режим работы.

```
    INQ_OPERATING_STATE_VALUE (OP_STATE);
    if (OP_STATE /= WSAC and OP_STATE /=SGOP) then
```

```
      ERRIND :=5;
      return;
    end if;
— Узнать преобразование станции.
    INQ_WS_TRANSFORMATION (WORKSTATION, ERRIND, STATUS,
        REQ_WINDOW, CUR_WINDOW, REQ_VIEWPORT, CUR_VIEWPORT);
                                          if ERRIND /=0 then
                                            return;
                                          end if;
— Узнать возможности ломаной.
    INQ_POLYLINE_FACITIIES (WS_TYPE, ERRIND, LINETYPE_LIST,
                  NUM_WIDTHS, NOMINAL_WIDTH, NOMINAL_WIDTH,
                                  RANGE_OF_WIDTHS, NUM_INDICES);
    if ERRIND /=0 then
      return;
    end if;
INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER
                                          (ERRIND, SAVED_PRIM_ATTR);
INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES
                                          (ERRIND, SAVE_INDV_ATTR);
  — Установить номер преобразования нормирования, флаги выборки индивидуальных атрибутов, масштаб толщины линии (1.0), индекс цвета ломаной (1) и атрибуты приемлемого текста.
SELECT_NORMALIZATION_TRANSFORMATION (0);
SET_ASF (LIST_OF_ASF);
SET_LINEWIDTH_SCALE_FACTOR (1.0);
SET_POLYLINE_COLOUR_INDEX (1);
SET_CHAR_UP_VECTOR ((0.0, 1.0));
SET_TEXT+PATH (RIGHT);
SET_TEXT_ALIGNMENT ((LEFT, HALF));
SET_TEXT_FONT_AND_PRECISION ((1, STRING_PRECISION));
SET_CHAR_EXPANSION_FACTOR (1.0);
SET_CHAR_SPACING (0.0);
SET_TEXT_COLOUR_INDEX (1);
— Вычислить расстояние между линиями.
DISTANCE := (CUR_WINDOW.YMAX — CUR_WINDOW.YMIN) / NDC_TYPE
                          (LINETYPES.SIZE_OF_LIST (LINETYPE_LIST));
— Установить высоту литер, равную половине расстояния между линиями, но не более, 1/20 высоты текущей станции.
if (DISTANCE/2.0) < ((CUR_WINDOW.YMAX — CUR_WINDOW.YMIN)/20.0
  then SET_CHAR_HEIGHT (WC.MAGNITUDE (DISTANCE/2.0));
else
    SET_CHAR_HEIGHT
      (WC.MAGNITUDE ((CUR_WINDOW.YMAX — CUR_WINDOW.YMIN)
                                                    /20.0));
end if;
— Отступление линии от левой границы к середине окна текущей станции.
PTS (1).X : = WC_TYPE (CUR_WINDOW.XMIN);
```

```
PTS (1).Y : = WC_TYPE (CUR_WINDOW.YMAX — DISTANCE/2.0);
PTS (2).X : = WC_TYPE (CUR_WINDOW.XMIN + CUR_WINDOW.XMAX
                                                    /2.0);
— Цикл по существующим типам линий.
for I in 1.. LINETYPES.SIZE_OF_LIST (LINETYPE_LIST) loop
    SET_LINETYPE (LINETYPES.LIST_ELEMENT (I, LINETYPE_LIST));
    PTS (2).Y : = PTS (1). Y;
    POLYLINE (PTS);
    PTS (1).Y : = PTS (1).Y — WC_TYPE (DISTANCE);
— Аннотировать тип линии.
TEXT (PTS(2,
      INTEGER' IMAGE (INTEGER (LINETYPES.LIST_ELEMENT
                               (I, LINETYPE.LIST) ) ) );
end loop;
— Восстановить номер преобразования нормирования и атрибуты.
SELECT_NORMALIZATION_TRANSFORMATION (0);
SET_ASF                          (SAVED_INDU_ATTR.ASF);
SET_LINEWIDTH_SCALE_FACTOR       (SAVED_INDV_ATTR.WIDTH);
SET_POLYLINE_COLOUR_INDEX        (SAVED_INDV_ATTR.LINE_COLOUR);
SET_CHAR_UP_VECTOR               (SAVED_PRIM_ATTR.CHAR_UP_
                                                    _VECTOR);
SET_TEXT_PATH                    (SAVED_PRIM_ATTR.PATH);
SET_TEXT_ALIGNMENT               (SAVED_PRIM_ATTR.ALIGHMENT);
SET_TEXT_FONT_AND_               (SAVED_INDV_ATTR.FONT_
PRECISION                                          _PRECISION);

SET_CHAR_EXPANSION_FACTOR        (SAVED_INDV_ATTR.EXPANSION);
SET_CHAR_SPACING                 (SAVED_INDV_ATTR.SPACING);
SET_TEXT_COLOUR_INDEX            (SAVED_INDV_ATTR.TEXT_COLOUR);
    ERRIND : =0;
end LINE_DEMO;
— Главная процедура SHOWLN.
begin
    — Вызов процедуры инициализации.
    INIT_GKS (TYPE_OF_WS, ERROR_IND)
    if ERROR_IND =0 then
    — Вызов подпрограммы демонстрации типов линий.
    LINE_DEMO (TYPE_OF_WS, ERROR_IND);
    end if;
    — Все закрыть.
    EMERGENCY_CLOSE_GKS;
end SHOWLN;
```

ПРИЛОЖЕНИЕ Г

МНОГОЗАДАЧНЫЙ РЕЖИМ РАБОТЫ ЯГС

(Это приложение не является составной частью стандарта, но дает дополнительную информацию).

Встраивание функций ЯГС как подпрограмм в пакет Ада реализует естественным образом: данные «состояния» ЯГС декларируют как переменные, локальные для тела пакета; они непосредственно доступны и модифицируются из подпрограмм ЯГС. Такой подход принимают, когда прикладные программы используют только последовательно управляемые структуры. Проблема состоит в том, что одновременные вызовы программ ЯГС могут испортить переменные состояния; например, при одновременной попытке записать в них. Данная проблема существует, где имеется истинная параллельность (множество процессов), или она моделируется (использование одного процессора с мультиплексированием).

Далее приводится метод реализации, который позволяет преодолеть данную трудность без изменения интерфейса с ЯГС со стороны прикладных программ на языке Ада. Коротко идея состоит в том, чтобы защитить данные тела пакета (например, переменные состояния), локализуя их в задаче, содержащейся в теле пакета. Для каждой подпрограммы, которая обращается к данным, будут существовать соответствующие входы, декларированные в задаче. Одно и то же имя может быть использовано для входа и подпрограммы, применяющей средства совмещения в языке Ада. Тело задачи похоже на монитор, то есть, это цикл, содержащий селективное ожидание с переходом для каждого входа. Предложение приема выполняет действительное считывание или запись информации по состоянию, как требуется соответствующими подпрограммами ЯГС. Тело каждой подпрограммы ЯГС сжимается просто до вызова входа задачи. Таким образом, даже если две задачи из пользовательского прикладного программного обеспечения одновременно вызывают подпрограммы, которые модифицируют или обращаются к переменным состояния, это будет приводить к вызовам входа задачи, которые ставятся в очередь и обслуживаются в порядке поступления. В данном случае опасность порчи переменных состояния отсутствует.

Для иллюстрации данного метода в следующем примере показано, как может быть написан скелет пакета ЯГС.

```
with GKS_TYPES;
use GKS_TYPES;
package GKS is;
   procedure OPEN_GKS
     (ERROR_FILE                 : in STRING :=DEFAULT_ERROR_
                                                  _FILE;
     AMOUNT_OF_MEMORY            : in NATURAL :=DEFAULT_MEMORY
                                                  _UNITS);
   procedure OPEN_WS
     (WS                         : in WS_ID;
     CONNECTION                  : in STRING;
     TYPE_OF_WS                  : in WS_TYPE);
   procedure CLOSE_GKS;
```

```
. . .
end GKS;
```

— Версия для последовательных прикладных программ:

```
with ERROR_HANDLING;
package body GKS is
```

— Переменные состояния:

```
  CURRENT_OPERATING_STATE : OPERATING_STATE : =GKCL;
SET_OF_OPEN_WORKSTATIONS : WS_IDS.LIST_OF : =WS_IDS.NULL_LIST;
procedure OPEN_GKS
(ERROR_FILE                    : in STRING :=DEFAULT_ERROR_
                                                  _FILE;
AMOUNT_OF_MEMORY               : in NATURAL :=DEFAULT_
                                          _MEMORY_UNITS) is
begin
. . .
if CURRENT_OPERATING_STATE/=GKCL then
    ERROR_HANDLING (1, «OPEN_GKS»);
else
    CURRENT_OPERATING_STATE : = GKOP;
end if;
. . .
end OPEN_GKS;
procedure OPEN_WS
      (WS                      : in WS_ID;
      CONNECTION               : in STRING;
      TYPE_OF_WS               : in WS_TYPE) is
begin
. . .
if CURRENT_OPERATING_STATE not in GKOP .. SGOP then
    ERROR_HANDLING (8, «OPEN_WS»);
else
    CURRENT_OPERATING_STATE : = WSOP;
    WS_IDS.ADD_TO_LIST(WS, SET_OF_OPEN_WORKSTATIONS);
end if;
. . .
end OPEN_WS;
    procedure CLOSE_GKS is
    begin
. . .
    if CURRENT_OPERATING_STATE / = GKOP then
      ERROR_HANDLING (2, «CLOSE_GKS»);
    else
      CURRENT_OPERATING_STATE : = GKCL;
    end if;
    . . .
    end CLOSE_GKS;
    . . .
end GKS;
```

— Версия для прикладных программ, использующих многозадачный режим работы:

```
with ERROR_HANDLING;
package body GKS is
task MONITOR is
entry OPEN_GKS
    (ERROR_FILE                    : in STRING :=DEFAULT_ERROR_
                                                    _FILE;
    AMOUNT_OF_MEMORY               : in NATURAL :=DEFAULT_MEMORY
                                                    _UNITS);
entry OPEN_WS
    (WS                            : in WS_ID;
    CONNECTION                     : in STRING;
    TYPE_OF_WS                     : in WS_TYPE);
entry CLOSE_GKS;
...
    end MONITOR;
    task body MONITOR is

— Переменные состояния:
CURRENT_OPERATING_STATE            :=OPERATING_STATE =GKCL;
SET_OF_OPEN_WORKSTATIONS           :=WS_IDS.LIST_OF
                                          :=WS_IDS.NULL_LIST;
begin
  loop
    begin
      select
        accept OPEN_GKS
          (ERROR_FILE              : in STRING :=DEFAULT_ERROR_
                                                    _FILE;
          AMOUNT_OF_MEMORY         : in NATURAL :=DEFAULT_MEMORY
                                                    _UNITS);

          do
          ...
          if CURRENT_OPERATING_STATE/=GKCL then
            ERROR_HANDLING (1, «OPEN_GKS»);
          else
            CURRENT_OPERATING_STATE : = GKOP;
          end if;
          ...
        end OPEN_GKS;
      or
        accept OPEN_WS
          (WS                      : in WS_ID;
          CONNECTION               : in STRING;
          TYPE_OF_WS               : in WS_TYPE) do
    ...
          If CURRENT_OPERATING_STATE /= not in GKOP..SGOP then
            ERROR_HANDLING (8, «OPEN_WS»);
          else
            CURRENT_OPERATING_STATE : = WSOP;
            WS_IDS.ADD_TO_LIST (WS, SET_OF_OPEN_WORKSTATIONS);
          end if;
          ...
```

```
            end OPEN_WS;
          or
            accept CLOSE_GKS do
              if CURRENT_OPERATING_STATE /= GKOP then
                ERROR_HANDLING (2, «CLOSE_GKS»);
              else
                CURRENT_OPERATING_STATE := GKCL;
              end if;
            . . .
            end CLOSE_GKS;
          . . .
        or
            terminate;
            end select;
            exception
              when others => null;
            end;
    end loop;
end MONITOR;
procedure OPEN_GKS
    (ERROR_FILE                    : in STRING := DEFAULT_ERROR_
                                                    _FILE;
    AMOUNT_OF_MEMORY               : in NATURAL := DEFAULT_MEMORY
                                                    _UNITS) is
begin
    MONITOR.OPEN_GKS(ERROR_FILE, AMOUNT_OF_MEMORY);
end OPEN_GKS;
      (WS                          : in WS_ID;
      CONNECTION                   : in STRING;
      TYPE_OF_WS                   : in WS_TYPE) is
begin
    MONITOR.OPEN_WS(WS, CONNECION, TYPE_OF_WS);
end OPEN_WS;
procedure CLOSE_GKS is
begin
    MONITOR.CLOSE_GKS;
end CLOSE_GKS;
. . .
end GKS;
```

Несколько замечаний по взаимодействию с обработкой прерываний событий в случае, когда процедура ERROR_HANDLING вызывает GKS_ERROR. Отметим, что в обеих версиях тела пакета вызывается процедура ERROR_-HANDLING. Предположим, что прикладная программа вызывает OPEN_GKS, когда ЯГС уже открыт. В последовательной версии вызов ERROR_HANDLING из тела процедуры OPEN_GKS вызовет распространение GKS_ERROR назад к прикладной программе, которая вызвала OPEN_GKS. В версии для многозадачного режима работы тот же самый эффект будет достигнут следующим образом. В течение выполнения предложения приема для входа OPEN_GKS будет вызвана процедура ERROR_HANDLING и произойдет GKS_ERROR. В соответствии с семантикой языка Ада, так как это исключение не обрабатывается локальным обработчиком при приеме, оно распространяется (1) в точку, следующую за приемом, и (2) в точку вызова входа. В первом случае оно обрабатывается в

блоке, включающем в себя предложение выбора; таким образом задача монитор может перейти к следующей итерации без разрыва. Для второго случая точка находится в теле процедуры OPEN_GKS. Так как здесь нет обработчика исключительных событий, то GKS_ERROR распространяется, как это и хотелось, назад к точке вызова в прикладной программе

Завершение выполнения задачи MONITOR будет реализовано выбором альтернативы завершения, когда задачи прикладной программы закончатся.

Существует ряд вариаций метода, рассмотренного и проиллюстрированного выше, которые при реализации могут представлять интерес с точки зрения повышения потенциального параллелизма прикладных программ ЯГС, использующих многозадачный режим работы. С помощью только что описанного метода единый набор переменных состояния защищается одной задачей. Если информация о состоянии может быть разделена на независимые наборы с одной задачей монитором на набор, то задача прикладной программы, считывающая/записывающая переменные в один набор, может выполнять это одновременно с задачей, которая считывает/записывает в другой набор.

Другой вариант защиты переменных состояния состоит в том, чтобы различить функции ЯГС, просто считывающие значения данных, от тех, которые записывают их. Метод, описанный выше, рассматривает «читателей» и «писателей» одинаково; таким образом он запрещает двум задачам прикладной программы одновременно считывать информацию о состоянии. Можно написать программу монитор таким образом, чтобы допускалось одновременное считывание задачами прикладной программы и запрещались одновременное записывание и одновременное считывание и записывание. Существует множество различных подходов к решению данной задачи, зависящих от учета программистом таких факторов, как:

— возможность задачи прикладной программы быть преждевременно прерванной;

— возможность задачи прикладной программы «голодать» по обслуживанию из-за того, как запрограммирован прием входов.

Реализации ЯГС, которые поддерживают многозадачные программы на языке Ада, могут быть использованы для программ, которые являются полностью последовательными, хотя эффективность при этом может несколько снизиться. При реализации может также возникнуть идея предоставить два тела пакета ЯГС: одно для последовательных применений, другое — для многозадачных.

ПРИЛОЖЕНИЕ Д

НЕПОДДЕРЖИВАЕМЫЕ ОБОБЩЕННЫЕ ПРИМИТИВЫ ВЫВОДА И РАСШИРЕНИЯ

(Это приложение не является составной частью стандарта, но дает дополнительную информацию.)

В настоящем приложении проясняются взаимоотношения между метафайлом GKSM и обобщенными примитивами вывода (ОПВ) и расширениями (ESC). Каждая функция ОПВ и ESC, являющаяся зарегистрированной, доступна для прикладных программ как отдельная процедура со своими собственными формальными параметрами и именем подпрограммы, как описано в п. 5.1.

Реализация ЯГС/Ада должна предоставлять возможность записывать и считывать зарегистрированные ОПВ и ESC в метафайл, даже если реализация не поддерживает функций ОПВ и ESC. Следовательно, чтобы данная возможность поддерживалась, форматы записей данных для зарегистрированных ОПВ и ESC должны существовать между реализациями.

Например, пусть метафайл А генерируется в реализации ЯГС/Ада, которая поддерживает ОПВ-окружность. Метафайл А теперь имеет запись данных, содержащую идентификатор окружности, точку центра и радиус. Далее метафайл А переносится в другую реализацию, которая не поддерживает ОПВ-окружность. В новой среде будет генерироваться метафайл В, содержащий весь метафайл А и дополнительные графические данные. И важно, чтобы ОПВ-окружность из метафайла А был включен в метафайл В, даже если ни одна станция в этой среде не способна генерировать изображение окружности.

Для иллюстрации метода в следующем примере показаны фрагмент прикладного кода и реализация функции метафайла
INTERPRET_ITEM.

```
with GKS;
use GKS;
with GKS_TYPES;
use GKS_TYPES;
— Данная прикладная программа передает данные из метафайла А в метафайл
В.
procedure TRANSFER_METAFILE is

— описание переменных
    INPUT_METAFILE                  : constant WS_ID := 1;
    OUTPUT_METAFILE                 : constant WS_ID := 2;

    INPUT_METAFILE_TYPE             : constant WS_TYPE := 2;
    OUTPUT_METAFILE_TYPE            : constant WS_TYPE := 3;
    INPUT_METAFILE_CONNECTION_ID
                                    : constant STRING := «METAFILE_A»;
    OUTPUT_METAFILE_CONNECTION_ID
                                    : constant STRING := «METAFILE_B»;

    METAFILE_DATA_RECORD            : GKSM_DATA_RECORD;
    METAFILE_ITEM_TYPE              : GKSM_ITEM_TYPE;
    LENGTH, MAX_LENGTH              : NATURAL := 500;
    ERROR_FILE : constant STRING := «MY_ERROR_FILE»;
begin
```

```
— Открыть ЯГС.
   OPEN_GKS (ERROR_FILE);

— Открыть метафайлы ввода и вывода.
   OPEN_WS (INPUT_METAFILE,
            INPUT_METAFILE_CONNECTION_ID,
            INPUT_METAFILE_TYPE);

   OPEN_WS (OUTPUT_METAFILE,
            OUTPUT_METAFILE_CONNECTION_ID,
            OUTPUT_METAFILE_TYPE);

— Активизировать только метафайл ввода.

   ACTIVATE_WS (OUTPUT_METAFILE);

— В данном цикле каждый элемент метафайла А считывается и передается в
ЯГС посредством вызова функции INTERPRET_ITEM. Помните, что метафайл
А содержит ОПВ-окружности, обрабатываемые функцией INTERPRET_ITEM,
которая представлена далее в примере.
loop
   GET_ITEM_TYPE_FROM_GKSM_(INPUT_METAFILE,
                            METAFILE_ITEM_TYPE, LENGTH);
    if METAFILE_ITEM_TYPE=0 then
    exit;      — выйти из цикла, метафайл полон.
    end if;
    READ_ITEM_FROM_GKSM (INPUT_METAFILE, MAX_LENGTH,
                         METAFILE_DATA_RECORD);
    INTERPRET_ITEM (METAFILE_DATA_RECORD);
end loop;
— Деактивируется только выходной файл.
DEAKTIVATE_WS (OUTPUT_METAFILE);
— Закрыть метафайлы ввода и вывода.
CLOSE_WS (INPUT_METAFILE);
CLOSE_WS (OUTPUT_METAFILE);
— Закрыть GKS.
CLOSE_GKS;
end TRANSFER_METAFILE;

— Пример прикладной программы опирается на реализацию функции
INTERPRET_ITEM для распознавания ОПВ-окружности и передачи ОПВ в
выходной метафайл. Это легко может быть выполнено следующим сегментом
кода.
— Предположим, что личный тип GKSM_DATA_RECORD декларирован как
тип записи дискриминанты с различными компонентами, основанными на типе
элементов. Когда GKSM_DATA_RECORD содержит ОПВ, имеются различные
поля, содержащие всю существующую информацию о ОПВ.
type GKSM_DATA_RECORD          : GKSM_ITEM_TYPE :=0;
                (TYPE_OF_ITEM  : NATURAL :=0) is
                    LENGTH

record
   case TYPE_OF_ITEM is
     when OPEN_GKS     =>...
     when POLYLINE     =>...
     when GDP          =>
```

```
      ID                          : GDP_ID;
      NUM_PTS                     : POSITIVE;
      INTEGER_DATA_LENGTH         : NATURAL;
      REAL_DATA_LENGTH            : NATURAL;
      LIST_OF_POINTS              : WC.POINT_ARRAY (1..NUM_PTS);
      INTEGER_DATA                : INTEGER_ARRAY
                                    (1..INTEGER_DATA_LENGTH);
      REAL_DATA                   : REAL_ARRAY (1..REAL_DATA_
                                                  _LENGTH);
    when...
  end case;
end record;
```

— Пример того, как реализация могла бы поддержать передачу нереализованного ОПВ через функцию INTERPRET_ITEM.

```
procedure INTERPRET_ITEM (ITEM : in GKSM_DATA_RECORD) is
   REGISTERED_GDP_CIRCLE : constant GDP_ID:=1;
begin
   case ITEM.TYPE_OF_ITEM is
     when OPEN_GKS =>...
     when POLYLINE =>...
     when GDP =>
       case ITEM.ID is
         when REGISTERED_GDP_SPLINE =>...
         when REGISTERED_GDP_ELLIPSE =>...
         when REGISTERED_GDP_CIRCLE =>
```

— Вызвать генератор метафайла, в котором запись данных ITEM как параметр записывается во все открытые и активные метафайлы.

```
         when...
     end case;
   end case;
end INTERPRET_ITEM;
```

ПРИЛОЖЕНИЕ Е

ТИПЫ ЭЛЕМЕНТОВ МЕТАФАЙЛА

(Это приложение не является составной частью стандарта, но дает дополнительную информацию.)

Функция GET ITEM TYPE FROM GKSM возвращает тип следующего элемента метафайла, однако значение данного типа может изменяться в зависимости от реализации метафайла. Для того, чтобы можно было писать программы, независимые от реализации метафайла, предлагаются имена в Аде (см. таблицу). В реализации следует задать эти имена со значениями, которые соответствуют значениям, возвращаемым процедурой GET ITEM TYPE FROM GKSM.

Тип элемента GKSM	Имя в Аде
ФЛАГИ ВЫБОРКИ АТРИБУТОВ	GKSM_ASF
МАТРИЦА ЯЧЕЕК	GKSM_CELL_ARRAY
МАСШТАБ РАСШИРЕНИЯ ЛИТЕР	GKSM_CHAR_EXPANSION_FACTOR
МЕЖЛИТЕРНЫЙ ПРОСВЕТ	GKSM_CHAR_SPACING
ВЕКТОРА ЛИТЕРЫ	GKCM_CHAR_VECTORS
ОЧИСТИТЬ СТАНЦИЮ	GKSM_CLEAR_WS
ПРЯМОУГОЛЬНИК ОТСЕЧЕНИЯ	GKSM_CLIPPING_RECTANGLE
ЗАКРЫТЬ СЕГМЕНТ	GKSM_CLOSE_SEGMENT
ПРЕДСТАВЛЕНИЕ ЦВЕТА	GKSM_COLOUR_REPRESENTATION
СОЗДАТЬ СЕГМЕНТ	GKSM_CREATE_SEGMENT
РЕЖИМ ЗАДЕРЖКИ	GKSM_DEFERRAL_STATE
УНИЧТОЖИТЬ СЕГМЕНТ	GKSM_DELETE_SEGMENT
КОНЕЦ ЗАПИСИ	GKSM_END_ITEM
РАСШИРЕНИЕ	GKSM_ESCAPE
ПОЛИГОНАЛЬНАЯ ОБЛАСТЬ	GKSM_FILL_AREA
ИНДЕКС ЦВЕТА ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	GKSM_FILL_AREA_COLOUR_INDEX
ИНДЕКС ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	GKSM_FILL_AREA_INDEX
ВИД ЗАПОЛНЕНИЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	GKSM_FILL_AREA_INTERIOR_STYLE
ПРЕДСТАВЛЕНИЕ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	GKSM_FILL_AREA_REPRESENTATION
ИНДЕКС ЗАПОЛНИТЕЛЯ ПОЛИГОНАЛЬНОЙ ОБЛАСТИ	GKSM_FILL_AREA_STYLE_INDEX
ОБОБЩЕННЫЙ ПРИМИТИВ ВЫВОДА (ОПВ)	GKSM_GDP
ТИП ЛИНИИ	GKSM_LINETYPE
МАСШТАБ ШИРИНЫ ЛИНИИ	GKSM_LINEWIDTH_SCALE_FACTOR

Тип элемента GKSM	Имя в Аде
МАСШТАБ РАЗМЕРА МАРКЕРА	GKSM_MARKER_SIZE_SCALE_FACTOR
ТИП МАРКЕРА	GKSM_MARKER_TYPE
СООБЩЕНИЕ	GKSM_MESSAGE
ОПОРНАЯ ТОЧКА ШАБЛОНА	GKSM_PATTERN_REFERENCE_POINT
ПРЕДСТАВЛЕНИЕ ШАБЛОНА	GKSM_PATTERN_REPRESENTATION
ВЕКТОРА ШАБЛОНА	GKSM_PATTERN_VECTORS
ИДЕНТИФИКАТОР ВЫБОРА	GKSM_PICK_ID
ЛОМАНАЯ	GKSM_POLYLINE
ИНДЕКС ЦВЕТА ЛОМАНОЙ	GKSM_POLYLINE_COLOUR_INDEX
ИНДЕКС ЛОМАНОЙ	GKSM_POLYLINE_INDEX
ПРЕДСТАВЛЕНИЕ ЛОМАНОЙ	GKSM_POLYLINE_REPRESENTATION
ПОЛИМАРКЕР	GKSM_POLYMARKER
ИНДЕКС ЦВЕТА ПОЛИМАРКЕРА	GKSM_POLYMARKER_COLOUR_INDEX
ПРЕДСТАВЛЕНИЕ ПОЛИМАРКЕРА	GKSM_POLYMARKER_REPRESENTATION
ПЕРЕРИСОВАТЬ ВСЕ СЕГМЕНТЫ НА СТАНЦИИ	GKSM_REDRAW_ALL_SEGMENT_WS
ПЕРЕИМЕНОВАТЬ СЕГМЕНТЫ	GKSM_RENAME_SEGMENT
ЗАДАТЬ ЧУВСТВИТЕЛЬНОСТЬ К УКАЗАНИЮ	GKSM_SET_DETECTABILITY
ЗАДАТЬ ВЫДЕЛЕНИЕ	GKSM_SET_HIGHLIGHTING
ЗАДАТЬ ПРИОРИТЕТ СЕГМЕНТА	GKSM_SET_SEGMENT_PRIORITY
ЗАДАТЬ ПРЕОБРАЗОВАНИЕ СЕГМЕНТА	GKSM_SET_SEGMENT_TRANSFORMATION
ЗАДАТЬ ВИДИМОСТЬ	GKSM_SET_VISIBLITY
ТЕКСТ	GKSM_TEXT
ВЫРАВНИВАНИЕ ТЕКСТА	GKSM_TEXT_ALIGNMENT
ИНДЕКС ЦВЕТА ТЕКСТА	GKSM_TEXT_COLOUR_INDEX
ШРИФТ И ТОЧНОСТЬ ТЕКСТА	GKSM_TEXT_FONT_AND_PRECISION
ИНДЕКС ТЕКСТА	GKSM_TEXT_INDEX
НАПРАВЛЕНИЕ ТЕКСТА	GKSM_TEXT_PATH
ПРЕДСТАВЛЕНИЕ ТЕКСТА	GKSM_TEXT_REPRESENTATION
ОБНОВИТЬ ИЗОБРАЖЕНИЕ НА СТАНЦИИ	GKSM_UPDATE_WS
НАЧАЛО ЗАПИСИ ПОЛЬЗОВАТЕЛЯ	GKSM_USER_ITEM
ПОЛЕ ВЫВОДА СТАНЦИИ	GKSM_WS_VIEWPORT
ОКНО СТАНЦИИ	GKSM_WS_WINDOW

ПРИЛОЖЕНИЕ Ж

ИНДЕКСЫ ФУНКЦИЙ ЯГС

(Это приложение не является составной частью стандарта, но дает дополнительную информацию.)

Ж.1. Функции ЯГС

ACCUMULATE TRANSFORMATION MATRIX . . . 93
ACTIVATE WORKSTATION . . . 52
ASSOCIATE SEGMENT WITH WORKSTATION . . . 62
AWAIT EVENT . . . 69

CELL ARRAY . . . 55
CLEAR WORKSTATION . . . 53
CLOSE GKS . . . 52
CLOSE SEGMENT . . . 62
CLOSE WORKSTATION . . . 52
COPY SEGMENT TO WORKSTATION . . . 63
CREATE SEGMENT . . . 62

DEACTIVATE WORKSTATION . . . 53
DELETE SEGMENT . . . 62
DELETE SEGMENT FROM WORKSTATION . . . 62

EMERGENCY CLOSE GKS . . . 94
ERROR HANDLING . . . 4, 5, 94
ERROR LOGGING . . . 5, 94
ESCAPE . . . 53
EVALUATE TRANSFORMATION MATRIX . . . 93

FILL AREA . . . 55
FLUSH DEVICE EVENTS . . . 69
GENERALIZED DRAWING PRIMITIVE . . . 55
GET CHOICE . . . 70
GET ITEM TYPE FROM GKSM . . . 71
GET LOCATOR . . . 70
GET PICK . . . 70
GET STRING . . . 70
GET STROKE . . . 70
GET VALUATOR . . . 70

INITIALISE CHOICE . . . 65
INITIALISE LOCATOR . . . 64
INITIALISE PICK . . . 65
INITIALISE STRING . . . 65
INITIALISE STROKE . . . 64
INITIALISE VALUATOR . . . 64
INQUIRE CHOICE DEVICE STATE . . . 81
INQUIRE CLIPPING . . . 81
INQUIRE COLOUR FACILITIES . . . 87
INQUIRE COLOUR REPRESENTATION . . . 79
INQUIRE CURRENT INDIVIDUAL ATTRIBUTE VALUES . . . 4, 73
INQUIRE CURRENT NORMALIZATION TRANSFORMATION NUMBER . . . 73
INQUIRE CURRENT PICK_IDENTIFIER VALUE . . . 73

INQUIRE CURRENT PRIMITIVE ATTRIBUTE VALUES 4, 72
INQUIRE DEFAULT CHOICE DEVICE DATA 90
INQUIRE DEFAULT DEFERRAL STATE VALUES 84
INQUIRE DEFAULT LOCATOR DEVICE DATA 89
INQUIRE DEFAULT PICK DEVICE DATA 91
INQUIRE DEFAULT STRING DEVICE DATA 91
INQUIRE DEFAULT STROKE DEVICE DATA 90
INQUIRE DEFAULT VALUATOR DEVICE DATA 90
INQUIRE DISPLAY SPACE SIZE 83
INQUIRE DYNAMIC MODIFICATION OF SEGMENT ATTRIBUTES . . . 88
INQUIRE DYNAMIC MODIFICATION OF WORKSTATION ATTRIBUTES . 83
INQUIRE FILL AREA FACILITIES 86
INQUIRE FILL AREA REPRESENTATION 79
INQUIRE GENERALIZED DRAWING PRIMITIVE 87
INQUIRE INPUT QUEUE OVERFLOW 92
INQUIRE LEVEL OF GKS 71
INQUIRE LIST OF AVAILABLE GENERALIZED DRAWING PRIMITIVES . 87
INQUIRE LIST OF AVAILABLE WORKSTATION TYPES 71
INQUIRE LIST OF COLOUR INDICES 79
INQUIRE LIST OF FILL AREA INDICES 78
INQUIRE LIST OF NORMALIZATION TRANSFORMATION NUMBERS . . 75
INQUIRE LIST OF PATTERN INDICES 79
INQUIRE LIST OF POLYLINE INDICES 77
INQUIRE LIST OF POLYMARKER INDICES 77
INQUIRE LIST OF TEXT INDICES 78
INQUIRE LOCATOR DEVICE STATE 80
INQUIRE MAXIMUM LENGTH OF WORKSTATION STATE TABLES . . . 88
INQUIRE MAXIMUM NORMALIZATION TRANSFORMATION NUMBER . . 72
INQUIRE MORE SIMULTANEOUS EVENTS 76
INQUIRE NAME OF OPEN SEGMENT 75
INQUIRE NORMALIZATION TRANSFORMATION 75
INQUIRE NUMBER OF AVAILABLE LOGICAL INPUT DEVICES . . . 89
INQUIRE NUMBER OF SEGMENT PRIORITIES SUPPORTED 88
INQUIRE OPERATING STATE VALUE 71
INQUIRE PATTERN FACILITIES 86
INQUIRE PATTERN REPRESENTATION 79
INQUIRE PICK DEVICE STATE 82
INQUIRE PIXEL 92
INQUIRE PIXEL ARRAY DIMENSION 92
INQUIRE PIXEL ARRAY 92
INQUIRE POLYLINE FACILITIES 84
INQUIRE POLYLINE REPRESENTATION 77
INQUIRE POLYMARKER FACILITIES 84
INQUIRE POLYMARKER REPRESENTATION 77
INQUIRE PREDEFINED COLOUR REPRESENTATION 87
INQUIRE PREDEFINED FILL AREA REPRESENTATION 86
INQUIRE PREDEFINED PATTERN REPRESENTATION 86
INQUIRE PREDEFINED POLYLINE PERPESENTATION 84
INQUIRE PREDEFINED POLYMARKER REPRESENTATION 85
INQUIRE PREDEFINED TEXT REPRESENTATION 85
INQUIRE SEGMENT ATTRIBUTES 91
INQUIRE SET OF ACTIVE WORKSTATIONS 72
INQUIRE SET OF ASSOCIATED WORKSTATIONS 91

INQUIRE SET OF OPEN WORKSTATIONS 72
INQUIRE SET OF SEGMENT NAMES IN USE 76
INQUIRE SET OF SEGMENT NAMES ON WORKSTATION 80
INQUIRE STRING DEVICE STATE 82
INQUIRE STROKE DEVICE STATE 81
INQUIRE TEXT EXTENT 78
INQUIRE TEXT FACILITIES 83
INQUIRE TEXT REPRESENTATION 78
INQUIRE VALUATOR DEVICE STATE 81
INQUIRE WORKSTATION CATEGORY 82
INQUIRE WORKSTATION CLASSIFICATION 83
INQUIRE WORKSTATION CONNECTION AND TYPE 76
INQUIRE WORKSTATION DEFERRAL AND UPDATE STATES 76
INQUIRE WORKSTATION MAXIMUM NUMBERS 71
INQUIRE WORKSTATION STATE 78
INQUIRE WORKSTATION TRANSFORMATION 80

INSERT SEGMENT 63
INTERPRET ITEM 71

MESSAGE 53

OPEN GKS 52
OPEN WORKSTATION 52

POLYLINE 55
POLYMARKER 55

READ ITEM FROM GKSM 71
REDRAW ALL SEGMENTS ON WORKSTATION 53
RENAME SEGMENT 63
REQUEST CHOICE 67
REQUEST LOCATOR 66
REQUEST PICK 67
REQUEST STRING 68
REQUEST STROKE 67
REQUEST VALUATOR 67

SAMPLE CHOICE 69
SAMPLE LOCATOR 68
SAMPLE PICK 69
SAMPLE STRING 69
SAMPLE STROKE 68
SAMPLE VALUATOR 68
SELECT NORMALIZATION TRANSFORMATION 61

SET ASPECT SOURCE FLAGS 59
SET CHARACTER EXPANSION FACTOR 58
SET CHARACTER HEIGHT 58
SET CHARACTER SPACING 58
SET CHARACTER UP VECTOR 58
SET CHOICE MODE 66
SET CLIPPING INDICATOR 61
SET COLOUR REPRESENTATION 60

SET DEFERRAL STATE 53
SET DETECTABILITY 63
SET FILL AREA COLOUR INDEX 56

SET FILL AREA INDEX 58
SET FILL AREA INTERIOR STYLE 59
SET FILL AREA REPRESENTATION 60
SET FILL AREA STYLE INDEX 59
SET HIGHLIGHTING 63
SET LINETYPE 57
SET LINEWIDTH SCALE FACTOR 57
SET LOCATOR MODE 65
SEI MARKER SIZE SCALE FACTOR 57
SET MARKER TYPE 57
SET PATTERN REFERENCE POINT 59
SET LINEWIDTH SCALE FACTOR 57
SET LOCATOR MODE 65
SET MARKER SIZE SCALE FACTOR 57
SET MARKER TYPE 57
SET PATTERN REFERENCE POINT 59
SET PATTERN REPRESENTATION 60
SET PATTERN SIZE 59
SET PICK IDENTIFIER 59
SET PICK MODE 66
SET POLYLINE COLOUR INDEX 57
SET POLYLINE INDEX 57
SET POLYLINE REPRESENTATION 59
SET POLYMARKER COLOUR INDEX 57
SET POLYMARKER INDEX 57
SET POLYMARKER REPRESENTATION 60
SET SEGMENT PRIORITY 63
SET SEGMENT TRANSFORMATION 63
SET STRING MODE 66
SET STROKE MODE 65
SET TEXT ALIGNMENT 58
SET TEXT FONT AND PRECISION 58
SET TEXT INDEX 57
SET TEXT PATH 58
SET TEXT REPRESENTATION 60
SET VALUATOR MODE 66
SET VIEWPORT 61
SET VIEWPORT INPUT PRIORITY 61
SET VISIBILITY 63
SET WINDOW 61
SET WORKSTATION VIEWPORT 62
SET WORKSTATION WINDOW 61
TEXT 55
UPDATE WORKSTATION 53
WRITE ITEM TO GKSM 70

Ж.2. **Процедуры языка Ада**

ACCUMULATE_TRANSFORMATION_MATRIX 93
ACTIVATE_WS 52
ASSOCIATE_SEGMENT_WITH_WS 62
AWAIT_EVENT 69

CELL_ARRAY 55
CLEAR_WS 53
CLOSE_GKS 52
CLOSE_SEGMENT 62
CLOSE_WS 52
COPY_SEGMENT_TO_WS 63
CREATE_SEGMENT 62

DEACTIVATE_WS 53
DELETE_SEGMENT 62
DELETE_SEGMENT_FROM_WS 62

EMERGENCY_CLOSE_GKS 94
ERROR_HANDLING 4, 5, 94
ERROR_LOGGING 5, 94
ESCAPE 53
EVALUATE_TRANSFORMATION_MATRIX 93

FILL_AREA 55
FLUSH_DEVICE_EVENTS 69

GDP 55
GET_CHOICE 70
GET_ITEM_TYPE_FROM_GKSM 71
GET_LOCATOR 70
GET_PICK 70
GET_STRING 70
GET_STROKE 70
GET_VALUATOR 70

INITIALISE_CHOICE 64
INITIALISE_LOCATOR 64
INITIALISE_PICK 65
INITIALISE_STRING 65
INITIALISE_STROKE 64
INITIALISE_VALUATOR 64

INQ_CHAR_BASE_VECTOR 130
INQ_CHAR_EXPANSION_FACTOR 131
INQ_CHAR_HEIGHT 129
INQ_CHAR_SPACING 131
INQ_CHAR_UP_VECTOR 129
INQ_CHAR_WIDTH 129
INQ_CHOICE_DEVICE_STATE 81
INQ_CLIPPING 73
INQ_COLOUR_FACILITIES 79
INQ_COLOUR_REPRESENTATION 4, 73
INQ_CURRENT_INDIVIDUAL_ATTRIBUTE_VALUES 74
INQ_CURRENT_NORMALIZATION_TRANSFORMATION_NUMBER 75
INQ_CURRENT_PICK_ID_VALUE 73
INQ_CURRENT_PRIMITIVE_ATTRIBUTE_VALUES 4, 72

INQ_DEFAULT_CHOICE_DEVICE_DATA 90
INQ_DEFAULT_DEFERRAL_STATE_VALUE 84
INQ_DEFAULT_LOCATOR_DEVICE_DATA 89
INQ_DEFAULT_PICK_DEVICE_DATA 91
INQ_DEFAULT_STRING_DEVICE_DATA 91

INQ_DEFAULT_STROKE_DEVICE_DATA 90
INQ_DEFAULT_VALUATOR_DEVICE_DATA 90
INQ_DISPLAY_SPACE_SIZE 83
INQ_DYNAMIC_MODIFICATION_OF_SEGMENT_ATTRIBUTES 88
INQ_DYNAMIC_MODIFICATION_OF_WS_ATTRIBUTES 83
INQ_FILL_AREA_COLOUR_INDEX 78
INQ_FILL_AREA_FACILITIES 86
INQ_FILL_AREA_INDEX 78
INQ_FILL_AREA_INTERIOR_STYLE 131
INQ_FILL_AREA_REPRESENTATION 133
INQ_FILL_AREA_STYLE_INDEX 133
INQ_GDP 87
INQ_INPUT_QUEUE_OVERFLOW 140
INQ_LEVEL_OF_GKS 71
INQ_LINETYPE 130
INQ_LINEWIDTH_SCALE_FACTOR 130
INQ_LIST_OF_ASF 131
INQ_LIST_OF_AVAILABLE_GDP 137
INQ_LIST_OF_AVAILABLE_WS_TYPES 129
INQ_LIST_OF_COLOUR_INDICES 133
INQ_LIST_OF_FILL_AREA_INDICES 133
INQ_LIST_OF_NORMALIZATION_TRANSFORMATION_NUMBERS 131
INQ_LIST_OF_PATTERN_INDICES 79
INQ_LIST_OF_POLYLINE_INDICES 77
INQ_LIST_OF_POLYMARKER_INDICES 77
INQ_LIST_OF_TEXT_INDICES 78
INQ_LOCATOR_DEVICE_STATE 80
INQ_MAX_LENGTH_OF_WS_STATE_TABLES 88
INQ_MAX_NORMALIZATION_TRANSFORMATION_NUMBER 72
INQ_MORE_SIMULTANEOUS_EVENTS 76
INQ_NAME_OF_OPEN_SEGMENT 75
INQ_NORMALIZATION_TRANSFORMATION 75
INQ_NUMBER_OF_AVAILABLE_LOGICAL_INPUT_DEVICES 88
INQ_NUMBER_OF_SEGMENT_PRIORITIES_SUPPORTED 88
INQ_OPERATING_STATE_VALUE 71
INQ_PATTERN_FACILITIES 86
INQ_PATTERN_HEIGHT_VECTOR 73
INQ_PATTERN_REFERENCE_POINT 73
INQ_PATTERN_REPRESENTATION 79
INQ_PATTERN_WIDTH_VECTOR 73
INQ_PICK_DEVICE_STATE 73
INQ_PIXEL 92
INQ_PIXEL_ARRAY 92
INQ_PIXEL_ARRAY_DIMENSION 92
INQ_POLYLINE_COLOUR_INDEX 74
INQ_POLYLINE_FACILITIES 84
INQ_POLYLINE_INDEX 72
INQ_POLYLINE_REPRESENTATION 77
INQ_POLYMARKER_COLOUR_INDEX 74
INQ_POLYMARKER_FACILITIES 136
INQ_POLYMARKER_INDEX 129
INQ_POLYMARKER_REPRESENTATION 77
INQ_POLYMARKER_SIZE_SCALE_FACTOR 74
INQ_POLYMARKER_TYPE 74

INQ_PREDEFINED_COLOUR_REPRESENTATION 87
INQ_PREDEFINED_FILL_AREA_REPRESENTATION 86
INQ_PREDEFINED_PATTERN_REPRESENTATION 86
INQ_PREDEFINED_POLYLINE_REPRESENTATION 84
INQ_PREDEFINED_POLYMARKER_REPRESENTATION 85
INQ_PREDEFINED_TEXT_REPRESENTATION 85
INQ_SEGMENT_ATTRIBUTES 91
INQ_SET_OF_ACTIVE_WS 72
INQ_SET_OF_ASSOCIATED_W 91
INQ_SET_OF_OPEN_WS 72
INQ_SET_OF_SEGMENT_NAMES_IN_USE 76
INQ_SET_OF_SEGMENT_NAMES_ON_WS 80
INQ_STRING_DEVICE_STATE 82
INQ_STROKE_DEVICE_STATE 81
INQ_TEXT_ALIGNMENT 73
INQ_TEXT_COLOUR_INDEX 74
INQ_TEXT_EXTENT 78
INQ_TEXT_FACILITIES 85
INQ_TEXT_FONT_AND_PRECISION 74
INQ_TEXT_INDEX 72
INQ_TEXT_PATH 73
INQ_TEXT_REPRESENTATION 78
INQ_VALUATOR_DEVICE_STATE 81
INQ_WS_CATEGORY 82
INQ_WS_CLASSIFICATION 83
INQ_WS_CONNECTION_AND_TYPE 76
INQ_WS_DEFERRAL_AND_UPDATE_STATES 76
INQ_WS_MAX_NUMBERS 76
INQ_WS_STATE 76
INQ_WS_TRANSFORMATION 80

INSERT_SEGMENT 63
INTERPRET_ITEM 71
MESSAGE 53

OPEN_GKS 52
OPEN_WS 52

POLYLINE 55
POLYMARKER 55

READ_ITEM_FROM_GKSM 71
REDRAW_ALL_SEGMENTS_ON_WS 53
RENAME_SEGMENT 62
REQUEST_CHOICE 67
REQUEST_LOCATOR 66
REQUEST_PICK 66
REQUEST_STRING 68
REQUEST_STROKE 67
REQUEST_VALUATOR 67

SAMPLE_CHOICE 69
SAMPLE_LOCATOR 68

SAMPLE_PICK 69

SAMPLE_STRING 69
SAMPLE_STROKE 68

SAMPLE_VALUATOR 68
SELECT_NORMALIZATION_TRANSFORMATION 61
SET_ASF 59
SET_CHAR_EXPANSION_FACTOR 58
SET_CHAR_HEIGHT 58
SET_CHAR_SPACING 58
SET_CHAR_UP_VECTOR 58
SET_CHOICE_MODE 66
SET_CLIPPING_INDICATOR 61
SET_COLOUR_REPRESENTATION 60
SET_DEFERRAL_STATE 53
SET_DETECTABILITY 63
SET_FILL_AREA_COLOUR_INDEX 59
SET_FILL_AREA_INDEX 58
SET_FILL_AREA_INTERIOR_STYLE 59
SET_FILL_AREA_REPRESENTATION 60
SET_FILL_AREA_STYLE_INDEX 59
SET_HIGHLIGHTING 63
SET_LINETYPE 57
SET_LINEWIDTH_SCALE_FACTOR 57
SET_LOCATOR_MODE 65
SET_MARKER_SIZE_SCALE_FACTOR 57
SET_MARKER_TYPE 57
SET_PATTERN_REFERENCE_POINT 59
SET_LINEWIDTH_SCALE_FACTOR 57
SET_LOCATOR_MODE 65
SET_MARKER_SIZE_SCALE_FACTOR 57
SET_MARKER_TYPE 57
SET_PATTERN_REFERENCE_POINT 59
SET_PATTERN_REPRESENTATION 60
SET_PATTERN_SIZE 59
SET_PICK_ID 59
SET_PICK_MODE 66
SET_POLYLINE_COLOUR_INDEX 57
SET_POLYLINE_INDEX 57
SET_POLYLINE_REPRESENTATION 59
SET_POLYMARKER_COLOUR_INDEX 57
SET_POLYMARKER_INDEX 57
SET_POLYMARKER_REPRESENTATION 60
SET_SEGMENT_PRIORITY 63
SET_SEGMENT_TRANSFORMATION 63
SET_STRING_MODE 66
SET_STROKE_MODE 65
SET_TEXT_ALIGNMENT 58
SET_TEXT_COLOUR_INDEX 58
SET_TEXT_FONT_AND_PRECISION 58
SET_TEXT_INDEX 57
SET_TEXT_PATH 58

SET_TEXT_REPRESENTATION 60
SET_VALUATOR_MODE 66
SET_VIEWPORT 61
SET_VIEWPORT_INPUT_PRIORITY 61
SET_VISIBILITY 63
SET_WINDOW 61
SET_WS_VIEWPORT 63
SET_WS_WINDOW 63
TEXT 55
UPDATE_WS 53
WRITE_ITEM_TO_GKSM 70

ИНФОРМАЦИОННЫЕ ДАННЫЕ

1. ПОДГОТОВЛЕН И ВНЕСЕН ТК 22 «Информационная технология»

2. УТВЕРЖДЕН И ВВЕДЕН В ДЕЙСТВИЕ Постановлением Госстандарта России от 28.12.92. № 1577
Настоящий стандарт подготовлен методом прямого применения международного стандарта ИСО 8651—3—88 «Системы обработки информации. Машинная графика. Связь ядра графической системы с языком программирования Ада»

3. ВВЕДЕН ВПЕРВЫЕ

4. ССЫЛОЧНЫЕ НОРМАТИВНО-ТЕХНИЧЕСКИЕ ДОКУМЕНТЫ

Обозначение НТД, на который дана ссылка	Номер пункта
ГОСТ 27817—88	Введение, 2, Приложение
ГОСТ 27831—88	2, Приложение

СОДЕРЖАНИЕ

0. ВВЕДЕНИЕ . . . 2
1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ . . . 2
2. ССЫЛКИ . . . 3
3. СВЯЗЬ ЯДРА ГРАФИЧЕСКОЙ СИСТЕМЫ С ЯЗЫКОМ АДА . . . 3
3.1. Условия соответствия стандарту . . . 3
3.2. Включение в язык . . . 4
3.2.1. Отображение функций . . . 4
3.2.2. Реализация и зависимость от компьютера . . . 4
3.2.3. Обработка ошибок . . . 4
3.2.4. Отображение данных . . . 5
3.2.5. Многозадачность . . . 6
3.2.6. Пакетирование . . . 6
3.2.7. Среда прикладных программ . . . 7
3.2.8. Регистрация . . . 8
4. ТАБЛИЦЫ . . . 8
4.1. Процедуры . . . 8
4.2. Определение типов данных . . . 25
4.2.1. Сокращения, используемые в определениях типов данных . . . 25
4.2.2. Определение типов в алфавитном порядке . . . 25
4.2.3. Список определений личных типов . . . 46
4.2.4. Список деклараций констант . . . 48
4.3. Коды ошибок . . . 50
4.3.1. Задание кодов ошибок . . . 51
4.3.2. Коды устраняемых ошибок . . . 51
5. ФУНКЦИИ В АДЕ, СВЯЗАННЫЕ С ЯДРОМ ГРАФИЧЕСКОЙ СИСТЕМЫ . . . 52
5.1. Функции ЯГС . . . 52
5.2. Дополнительные функции . . . 94
5.2.1. Подпрограммы для манипулирования записями входных данных . . . 94
5.2.2. Пакет обобщенной координатной системы ЯГС . . . 96
5.2.3. Общий пакет списка утилит ЯГС . . . 98
5.2.4. Утилиты функций метафайла . . . 101
5.3. Настраивающиеся варианты . . . 101
Приложение А. Спецификация скомпилированного ЯГС . . . 102
Приложение Б. Список ссылок на определенные реализацией записи . . . 144
Приложение В. Примеры программ . . . 145
Приложение Г. Многозадачный режим работы ЯГС . . . 160
Приложение Д. Неподдерживаемые обобщенные примитивы вывода и расширения . . . 165
Приложение Е. Типы элементов метафайла . . . 168
Приложение Ж. Индексы функций ЯГС . . . 170
Информационные данные . . . 179

Редактор *Р. С. Федорова*
Технический редактор *В. Н. Прусакова*
Корректор *М. С. Кабанова*

Сдано в набор 22.01.93. Подп. в печ. 31.05.93. Усл. печ. л. 10,69 Усл. кр-отт. 10,82. Уч.-изд. л. 13,66. Тир. 341. С 231.

Ордена «Знак Почета» Издательство стандартов, 107076. Москва, Колодезный пер., 14.
Калужская типография стандартов, ул. Московская, 256. Зак. 207

к ГОСТ Р 34.1702.3—92 (ИСО 8651-3—88) Информационная технология. Машинная графика. Связь ядра графической системы с языком программирования Ада

В каком месте	Напечатано	Должно быть
Обложка, титульный лист	ГОСТ 34.1702.3—92	ГОСТ Р 34.1702.3—92

(ИУС № 12 2007 г.)

gost.e-ecolog.ru