

Irbis Forth

Версия документа 1.0, апрель 2022

Версия программы 0.1 build 5

Введение

Irbis (Ирбис-Форт) - графическое приложение 64-разрядного транслятора языка Форт для ОС Windows. Программа реализует виртуальную стековую машину, позволяющую в интерактивном режиме выполнять команды, набор которых основан на стековом языке программирования Форт, а также компилировать новые команды («слова»). Основной идеей данной реализации языка Форт является добавление форт-машины к заранее определенному набору визуальных компонентов графического интерфейса для быстрой настройки внешнего вида приложения и выполнения динамически компилируемого кода на Форте, запускаемого этими визуальными компонентами.

Ключевые особенности:

- 64-разрядные целочисленные данные;
- поддержка вычислений с плавающей точкой;
- трехмерная графика на базе OpenGL;
- поддержка MIDI, последовательных портов и др. (всего более 20 классов компонентов);
- программный интерфейс для подключения внешних dll и импорта функций;
- свободная лицензия.

Внешний вид стартового экрана показан на рис. 1.

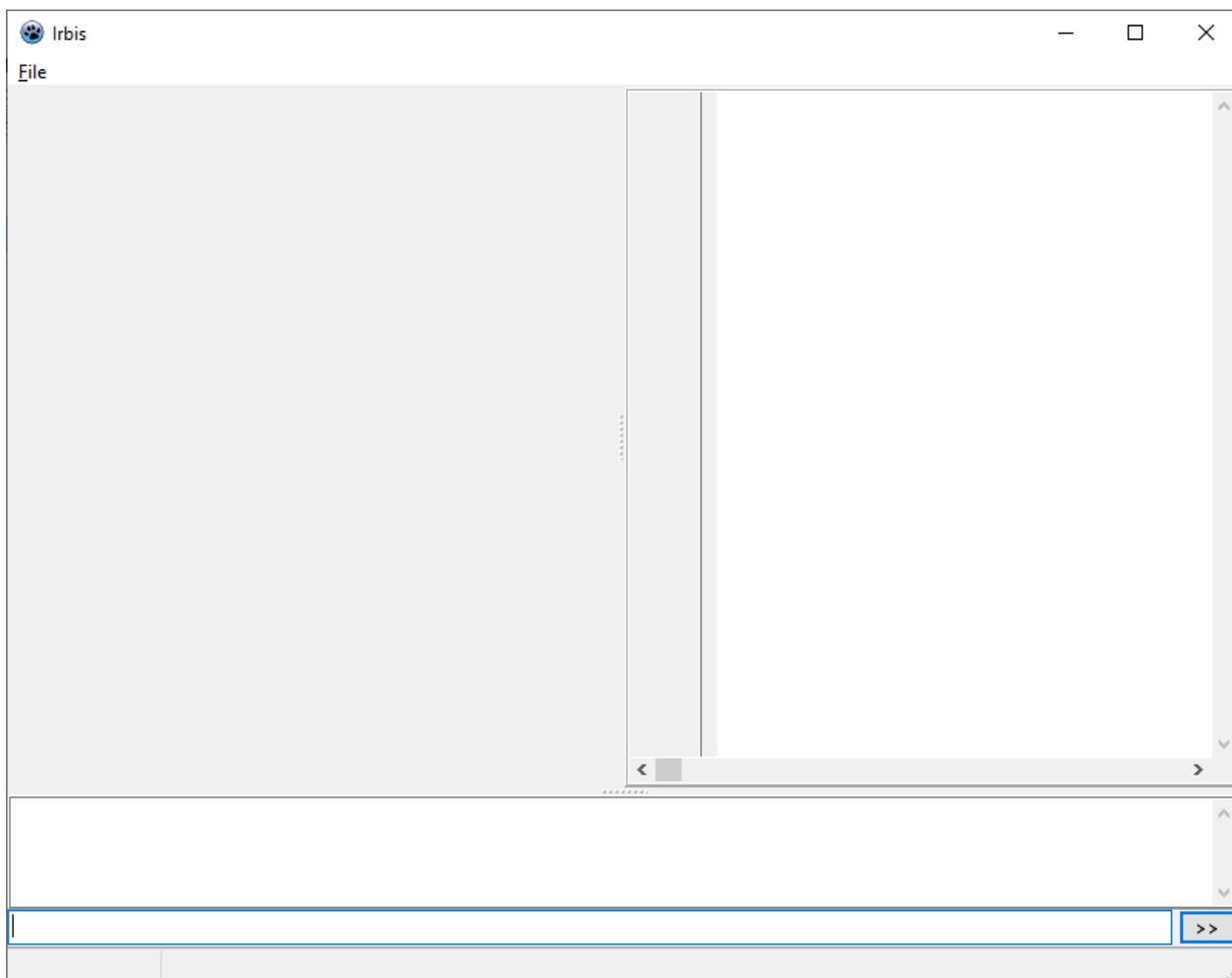


Рис. 1. Внешний вид стартового экрана Irbis

1. Установка

Программа реализована в виде единственного исполняемого файла `irbis.exe` и не требует для работы внешних библиотек. Установка выполняется путем копирования исполняемого файла в выбранный пользователем каталог.

Программа работает под управлением 64-разрядной ОС Windows. Тестирование выполнялось на базе MS Windows 10.

Требования к аппаратным средствам:

- процессор с архитектурой x86-64;
- дисплей с разрешением не хуже 1024x768;
- видеокарта с поддержкой OpenGL;
- мышь или аналогичное устройство.

2. Архитектура программы и основные характеристики.

Программа основана на IDE Lazarus и написана на языке Object Pascal с применением соответствующих классов для графических объектов. В программу встроена Форт-машина, позволяющая управлять поведением этих объектов в реальном времени, в режиме интерпретации передаваемых строковых команд, или путем исполнения скомпилированного кода.

Форт-машина использует вариант свернутого шитого кода – указание на слова Форты реализуется 32-разрядными токенами, представляющими собой идентификаторы отдельных подпрограмм. В текущей версии доступно 1 Мбайт памяти для хранения программ, представленных токенами. Для данных выделяется 128 Мбайт памяти. Эти значения будут изменены в последующих версиях, в том числе с введением динамического управления размером памяти.

Основным форматом данных являются 64-битные целые числа. Также поддерживаются 64-битные числа с плавающей точкой (double precision), реализованные на отдельном стеке. Строки реализуются как байтовые массивы в формате ASCIIZ (строка, завершаемая нулевым байтом).

Поддерживается подключение внешних динамических библиотек (dll). Подключение произвольной функции выполняется с помощью слов LoadLibrary и GetProcessAddress. В текущей версии имеются ограничения на передачу параметров, поддерживается вызов функций с 0 – 6 целочисленными аргументами. При передаче адресов необходима коррекция на абсолютное положение массива данных Форт-машины (словом DATA[+]).

Назначением программы является быстрое прототипирование алгоритмов и приложений с использованием динамически создаваемых графических интерфейсов пользователя. Элементы интерфейса предварительно создаются при запуске программы и находятся в скрытом состоянии. Управление параметрами отдельных объектов производится с помощью слов Форты, определенных в базовом словаре. В настоящее время

поддерживаются следующие типы объектов (с ограниченным набором функций):

- TButton
- TBitBtn
- TLabel
- TEdit
- TPanel
- TSplitter
- TCheckBox
- TListBox
- TComboBox
- TImage
- TProgressBar
- TTrackBar
- TMemo
- TStringGrid
- TSynEdit
- TRadioGroup
- TChart
- TTimer
- TOpenGL
- MMSystem
- COM (UART)
- text file
- file of byte

Программа запускается в режиме графического приложения, как показано на рис. 1. В правой части экрана показано окно редактора, текст из которого можно запустить на трансляцию. Если в командной строке при запуске передано имя файла, этот файл загружается в окно редактора и выполняется. Окно ввода в консоли также предназначена для выполнения

отдельных строк. Окно вывода консоли по умолчанию используется для вывода сообщений из Форт-машины (в том числе словами . F. PRINT).

Окно редактора и консоль могут быть скрыты и показаны программным способом (исполнением соответствующих слов Форта). Также реализованы горячие клавиши Ctrl-E для переключения состояния окна редактора и Ctrl-Q для переключения состояния консоли.

Вводимые команды оказывают немедленный эффект на определенные в программе объекты. Например, выполнение строки 0 button.show приведет к появлению на экране кнопки (где 0 - индекс кнопки в массиве предопределенных кнопок типа TButton). Команды управления могут входить в состав скомпилированных определений. Кроме того, ряд элементов управления (кнопки, Trackbar, некоторые события клавиатуры и мыши) приводят к вызову соответствующих слов Форт-машины, которые могут быть динамически переопределены для реализации действий, необходимых пользователю.

На рис. 2 показан вид программы при реализации простого примера из набора RosettaCode.

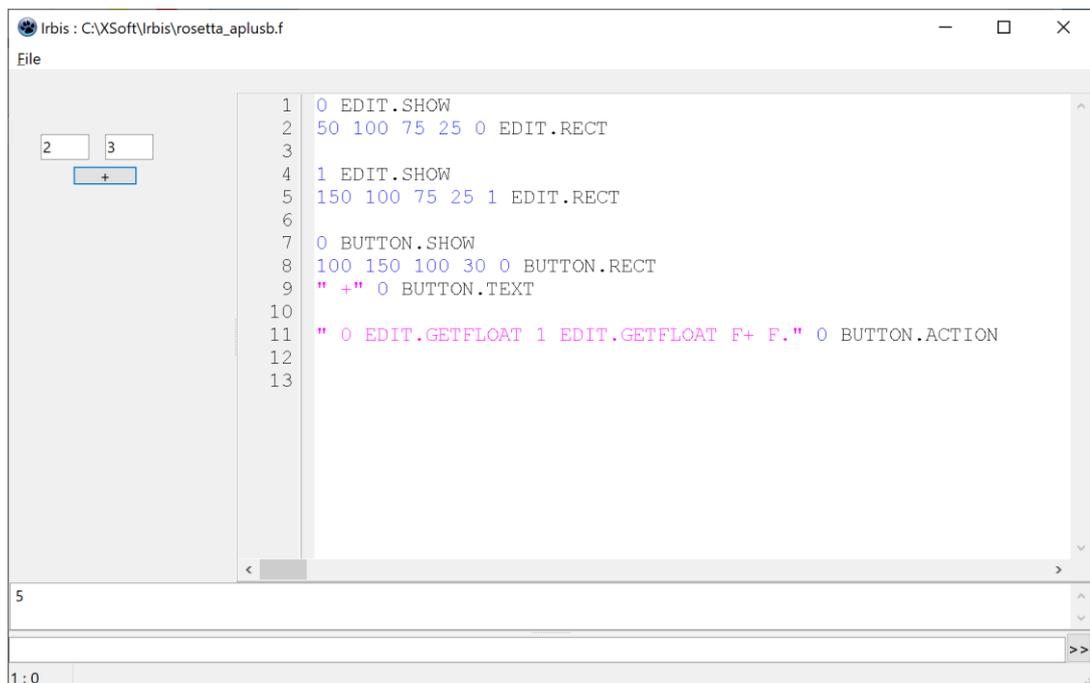


Рис. 2. Пример реализации сложения двух чисел (из набора RosettaCode)

На рисунке видно, что программа на Irbis Forth показывает на экране два объекта типа TEdit. При этом команда Edit.Show вызывает появление окна ввода на экране, а команда Edit.Rect задает его координаты и размеры (в формате x y width height). Аналогичная пара команд показывает на экране кнопку и задает ее координаты. Команда Button.Text задает текст на поверхности кнопки, а команда button.action определяет текстовую строку, которая должна быть выполнена форт-машиной при нажатии на кнопку. При этом в строке видно, что в процессе работы из текстовых полей считывается содержимое, которое преобразуется в формат с плавающей точкой (командами edit.getfloat). Вывод результата словом F. производится в консоль.

3. Описание слов.

Реализованные слова образуют подмножество слов языка Форт с добавлением слов для работы с графическими и другими объектами (файлами, мультимедиа, последовательными портами). В текущей версии поиск слов ведется без учета регистра (т.е. слово может быть записано как DUP dup Dup).

Комментарии задаются словом // (двойной слеш, за которым следует минимум один пробел) и действуют до конца строки. Вложенные и многострочные комментарии не используются.

Целочисленные литералы используют формат [-][0-9]. Поддерживается шестнадцатеричный формат представления, начинающийся с символа х (например, хFF). Другие форматы целочисленного представления в текущей версии не поддерживаются, переменная BASE определена, но не используется (это планируется привести в соответствие с традиционным поведением языка Форт).

Литералы с плавающей точкой используют формат с точкой и/или символом e.

Строки задаются символом “ (двойная кавычка), за которым следует минимум один пробел. Строка завершается двойной кавычкой. В текущей версии строковые переменные содержат 256 символов. Проверка выхода за

пределы строки в текущей версии не реализована. Конец строки идентифицируется нулевым байтом, счетчик строки не используется. Определение длины строки выполняется с помощью соответствующего слова, определенного в базовом наборе. Все слова, использующие строковые операнды, ориентированы на использование приведенного формата строк.

Список слов, их стековые нотации и описания приведены ниже. Используется общепринятый формат стековой нотации, где состояние стека до выполнения слова приводится слева, далее идет символ --, после которого приводится состояние стека после выполнения. Если числа находятся на стеке чисел с плавающей точкой, перед списком ставится обозначение f:

NOP – нет операции

DUP // x – x, x дублирование числа на вершине стека данных

DROP // x – удаление числа с вершины стека данных

SWAP // a, b – b, a - обмен местами двух верхних чисел на стеке данных

OVER // a, b – a, b, a - копирование на вершину второго сверху числа

ROT // a, b, c – b, c, a - «вращение» трех верхних значений

+ // a, b – a+b – сложение

- // a, b – a-b – вычитание

* // a, b – a*b – умножение

/ // a, b – a/b – целочисленное деление

MOD // a, b – (a mod b) – остаток от целочисленного деления

/MOD // a, b – a/b, (a mod b) – частное и остаток от целочисленного деления

ABS // x – |x| - абсолютное значение (модуль) числа

NEGATE // x – -x - смена знака числа

AND // a, b – a and b – поразрядное логическое И

OR // a, b – a or b – поразрядное логическое ИЛИ

XOR // a, b – a xor b – поразрядное логическое ИСКЛЮЧАЮЩЕЕ ИЛИ

= // a, b – t/f – результат сравнения чисел, остается ИСТИНА, если числа

равны

$\langle \rangle$ // a, b – t/f – результат сравнения чисел, остается ИСТИНА, если числа не равны

!= - то же, что $\langle \rangle$

$<$ // a, b – t/f – результат сравнения чисел, остается ИСТИНА, если $a < b$

$>$ // a, b – t/f – результат сравнения чисел, остается ИСТИНА, если $a > b$

@ // addr – data – чтение 64-разрядного числа из адреса, задаваемого addr

! // data, addr – запись 64-разрядного числа data в память данных по адресу addr

C@ // addr – data – чтение 8-разрядного числа из адреса, задаваемого addr

C! // data, addr – запись 8-разрядного числа data в память данных по адресу addr

CELLS // x – $x*8$ – получить размер в байтах, требуемый для размещения x целых чисел

FLOATS // x – $x*8$ – получить размер в байтах, требуемый для размещения x чисел с плавающей точкой

-TH // addr, i – $addr+i*8$ – получить адрес i-го элемента массива, начинающегося с адреса addr

S>F // x – f:x – преобразовать целое число в число с плавающей точкой

F>S // f:x – x – преобразовать число с плавающей точкой в целое число

FLIT – служебное слово, реализующее помещение литерала с плавающей точкой на стек

FDUP // f:x – f: x, x - дублирование числа на вершине стека чисел с плавающей точкой

FDROP // f:x – удаление числа с вершины стека чисел с плавающей точкой

FSWAP // f:a, b – f:b, a - обмен местами двух верхних чисел на стеке с плавающей точкой

FOVER // f:a, b – f: a, b, a - копирование на вершину второго сверху числа на стеке с плавающей точкой

FROT // f: a, b, c – f: b, c, a - «вращение» трех верхних значений на стеке чисел с плавающей точкой

F+ // f: a, b – f: a+b – сложение чисел с плавающей точкой

F- // f: a, b – f: a-b – вычитание чисел с плавающей точкой

F*// f: a, b – f: a*b – умножение чисел с плавающей точкой

F/ // f: a, b – f: a/b – деление чисел с плавающей точкой

FABS // f: x – f: |x| - абсолютное значение (модуль) числа с плавающей точкой

FNEGATE // f: x – f: -x - смена знака числа с плавающей точкой

F@ // addr – f: data - чтение 64-разрядного числа с плавающей точкой из адреса, задаваемого addr

F! // addr, f :data – запись 64-разрядного числа с плавающей точкой data в память данных по адресу addr

FSIN // f: x – f: sin(x) – вычисление синуса от аргумента с плавающей точкой (аргумент в радианах)

FCOS // f: x – f: cos(x) – вычисление косинуса от аргумента с плавающей точкой (аргумент в радианах)

FEXP // f: x – f: exp(x) – вычисление экспоненты от аргумента с плавающей точкой

" // -- addr – начать определение строки, строка размещается в HERE

// - начало комментария, комментарий действует до конца текущей строки

NOT // x – t|f – логическое отрицание для числа на вершине стека

DEPTH // -- depth – положить на стек текущую глубину стека данных

FDEPTH // -- fdepth – положить на стек текущую глубину стека чисел с плавающей точкой

RDEPTH // -- rdepth – положить на стек текущую глубину стека возвратов, // x – перенести число со стека данных в память данных, переместив указатель свободного места

HERE // -- here – положить на стек текущий указатель первого свободного байта в массиве памяти данных

[C]HERE // [c]here – положить на стек текущий указатель первого свободного байта в массиве памяти программ

RANDOM // -- random – положить на стек псевдослучайное число

RANDOMIZE // -- выполнить инициализацию («перемешивание») генератора псевдослучайных чисел

DELAY // time – выполнить задержку на time миллисекунд

DATA[]+ // addr -- addr+adr0 – получить из относительного адреса массива данных форт-машины абсолютный адрес в терминах ОС

[X*Y] // addr1, addr2, n – y – вычислить свертку массивов с целочисленными данными (сумму произведений элементов с одинаковыми индексами), адреса массивов задаются addr1, addr2, количество элементов - n

[FX*FY] // addr1, addr2, n – f:y – вычислить свертку массивов с плавающей точкой (сумму произведений элементов с одинаковыми индексами), адреса массивов задаются addr1, addr2, количество элементов - n

[X*FY] // addr1, addr2, n – f:y – вычислить свертку массивов, где первый массив содержит числа int64, а второй – числа с плавающей точкой (сумму произведений элементов с одинаковыми индексами), адреса массивов задаются addr1, addr2, количество элементов - n

F, // f:x -- перенести число со стека с плавающей точкой в память данных, переместив указатель свободного места

SWAP32 // ху – ух – выполнить обмен старшей и младшей 32-разрядных частей верхнего числа на стеке данных (0x12345678_abcdef01 преобразуется в 0xabcdef01_12345678)

CREATE – создать новое слово в словаре, взяв имя из входного потока, действием которого будет помещение на стек адреса первого свободного байта в памяти данных на момент создания слова

: - начать создание нового слова в словаре, взяв имя из входного потока

PROC – то же, что :

; - завершить определение текущего слова

ENDPROC – то же, что ;

ALLOT // x – переместить указатель первого свободного байта в памяти данных на x элементов

VARIABLE – создать целочисленную переменную, взяв имя из входного потока и выделив для нее 8 байт

CONSTANT // x -- - создать константу, взяв имя из входного потока и присвоить ей значение, снятое со стека данных

FLOAT – создать переменную с плавающей точкой, взяв имя из входного потока и выделив для нее 8 байт, фактическое действие эквивалентно слову VARIABLE, создаваемые таким образом переменные неразличимы средствами языка

Конструкции управления языка Форт:

IF THEN ELSE BEGIN UNTIL WHILE REPEAT DO LOOP +LOOP

I – положить на стек значение счетчика цикла

J – положить на стек значение счетчика объемлющего цикла

K – положить на стек значение счетчика следующего по отношению к J объемлющего цикла

Определено в словаре, но не реализовано в текущей версии:

CASE ENDCASE OF ENDOF BREAK

DOES> - завершение конструкции CREATE DOES>

USE // -- x – найти идентификатор определения, взяв имя из входного потока (слово немедленного исполнения)

VIRTUAL – создать виртуальное слово, выполняющее в процессе работы вызов другого слова, идентификатор которого определяется динамически, используется в сочетании со словами USE и TO

TO // x -- - записать число с вершины стека в слово, имя которого выбирается из входного потока

QUAN // определить целочисленную переменную, взяв имя из входного потока, действием созданного слова является помещение на стек своего значения, запись нового значения выполняется конструкцией x TO name

INT // то же, что `quan`

STRING // определить строковую переменную, взяв имя из входного потока и выделив 256 байт

L // `name` – интерпретировать файл с именем, заданным в виде адреса строки

INCLUDE // - то же, что L

S! // `str_dest`, `str_source` – записать строку, заданную адресом `str_source` по адресу `str_dest`

S+ // `str_dest`, `str_source` – добавить строку, заданную адресом `str_source` к строке по адресу `str_dest`

>STR // `x`, `str` -- - преобразовать число `x` в строковое представление, записав его в строку по адресу `str`

F>STR // `str`, `f`: `x` --- преобразовать число с плавающей точкой `x` в строковое представление, записав его в строку по адресу `str`

%D // `str`, `x` -- - добавить строковое представление числа `x` к строке, заданной адресом `str`

%F // `str`, `f`: `x` -- - - добавить строковое представление числа с плавающей точкой `x` к строке, заданной адресом `str`

STRLEN // `str` – `len` – вернуть на стеке длину строки в байтах, заданной адресом начального символа

S= // `str1`, `str2` – `t|f` – сравнить строки, заданные начальными адресами, вернув флаг ИСТИНА, если содержимое строк равно

STR>INT // `str` – `x` – попытаться преобразовать строку, заданную начальным адресом, в целочисленную величину

STR>FLOAT // `str` – `f`:`x` – попытаться преобразовать строку, заданную начальным адресом, в величину с плавающей точкой

CONVEERROR // - `error_code` – положить на стек код ошибки при преобразовании чисел (0, если ошибки нет)

LOADLIBRARY // `str` – `handle` – загрузить динамическую библиотеку, имя которой задано строкой `str`, вернув указатель

GETPROCADDRESS // handle, str – addr – получить адрес функции из ранее загруженной динамической библиотеки, заданной указателем handle, где имя функции задается строкой str, вернув в случае успеха адрес функции addr

INVOKE // addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr, получив на стеке результат ее выполнения

INVOKE0 – то же, что INVOKE

INVOKE1 // arg1, addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr, получив на стеке результат ее выполнения, при этом функции передается целочисленный аргумент arg1

INVOKE2 // arg1, arg2, addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr, получив на стеке результат ее выполнения, при этом функции передаются целочисленные аргументы arg1, arg2

INVOKE3 // arg1, arg2, addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr, получив на стеке результат ее выполнения, при этом функции передаются целочисленные аргументы arg1 – arg3

INVOKE4 // arg1, arg2, addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr, получив на стеке результат ее выполнения, при этом функции передаются целочисленные аргументы arg1 – arg4

INVOKE5 // arg1, arg2, addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr, получив на стеке результат ее выполнения, при этом функции передаются целочисленные аргументы arg1 – arg5

INVOKE6 // arg1, arg2, addr – result – вызвать ранее загруженную функцию из динамической библиотеки, адрес которой задан значением addr,

получив на стеке результат ее выполнения, при этом функции передаются целочисленные аргументы `arg1 – argb`

`CURRENT // -- addr` – положить на стек адрес текущего словаря (словарь, в который добавляются новые слова)

`CONTEXT // -- addr` – положить на стек адрес контекстного словаря (словарь, в котором выполняется поиск слов)

`FORTH` – сделать словарь `FORTH` контекстным

`VOCABULARY` – создать новый словарь, взяв его имя из входного потока

`DEFINITIONS` – сделать контекстный словарь текущим

[- переключиться в режим интерпретации

] – переключиться в режим компиляции

`VLIST` – вывести в консоль список слов, доступных при поиске

`$` - выполнить обработку сообщений операционной системы

`APPLICATION.PROCESSMESSAGES` – то же, что `$`

`TIMER` – виртуальное слово, выполняющееся регулярно при обработке системного таймера

Текстовые и двоичные файлы описаны в виде заранее подготовленных структур данных, сгруппированных в массивы. Системный идентификатор файла хранится внутри этой структуры и недоступен форт-машине. Вместо этого для работы используются индексы структур, начинающиеся с 0.

`FILE.TEXT.OPEN // str, index – result` – попытка открыть текстовый файл в режиме чтения, файл задан именем `str`, привязав его к внутренней структуре данных `index` (начинается с 0), код ошибки возвращается на стеке (0 – файл открыт успешно)

`FILE.TEXT.OPENW // str, index – result` – попытка открыть текстовый файл в режиме записи, файл задан именем `str`, привязав его к внутренней структуре данных `index` (начинается с 0), код ошибки возвращается на стеке (0 – файл открыт успешно)

`FILE.TEXT.CLOSE // index – result` – закрыть текстовый файл

FILE.TEXT.READ – не реализовано

FILE.TEXT.WRITE – не реализовано

FILE.TEXT.READLINE // index, str – прочитать из файла с индексом index строку, разместив ее в строковой переменной с адреса str

FILE.TEXT.WRITELINE // index, str – записать в файл с индексом index строку с адреса str

FILE.TEXT.EOF // index – t|f – поместить на стек ИСТИНУ, если достигнут конец файла

FILE.TEXT.X // index, addr – n – попытаться заполнить массив целых чисел, начинающийся с адреса addr, из текстового файла с индексом index, в каждой строке которого записано одно число, на стеке возвращается количество n прочитанных таким способом чисел

FILE.TEXT.XY // index, addr – n – попытаться заполнить массив целых чисел, начинающийся с адреса addr, из текстового файла с индексом index, в каждой строке которого записано два числа, разделенных хотя бы одним пробелом, на стеке возвращается количество n прочитанных таким способом пар чисел

FILE.TEXT.FX // index, addr – n – попытаться заполнить массив чисел с плавающей точкой, начинающийся с адреса addr, из текстового файла с индексом index, в каждой строке которого записано одно число, на стеке возвращается количество n прочитанных таким способом чисел

FILE.TEXT.FXY // index, addr – n – попытаться заполнить массив чисел с плавающей точкой, начинающийся с адреса addr, из текстового файла с индексом index, в каждой строке которого записано два числа, разделенных хотя бы одним пробелом, на стеке возвращается количество n прочитанных таким способом пар чисел

FILE.BIN.OPEN // str, index – result – попытка открыть двоичный файл в режиме чтения, файл задан именем str, привязав его к внутренней структуре данных index (начинается с 0), код ошибки возвращается на стеке (0 – файл открыт успешно)

`FILE.BIN.OPENW // str, index – result` – попытка открыть двоичный файл в режиме записи, файл задан именем `str`, привязав его к внутренней структуре данных `index` (начинается с 0), код ошибки возвращается на стеке (0 – файл открыт успешно)

`FILE.BIN.CLOSE // index – result` – закрыть двоичный файл

`FILE.BIN.READ // index, addr, size – result` – прочитать `size` байтов из двоичного файла, заданного индексом `index`, и поместить их начиная с адреса `addr`, код ошибки возвращается на стеке (0 – операция выполнена успешно)

`FILE.BIN.WRITE // index, addr, size – result` – записать `size` байтов из двоичного файла, заданного индексом `index`, взяв их начиная с адреса `addr`, код ошибки возвращается на стеке (0 – операция выполнена успешно)

`FILE.BIN.EOF // index – t|f` – поместить на стек ИСТИНУ, если достигнут конец файла

`FILE.BIN.SIZE // index – size` – поместить на стек размер двоичного файла в байтах, заданного индексом `index`

`FILE.BIN.POSITION` – не реализовано (зарезервировано для будущей реализации)

`. // x --` - напечатать число со стека в консоли

`F. // f: x --` - напечатать число со стека с плавающей точкой в консоли

`PRINT // str` – напечатать в консоли строку, заданную начальным адресом на стеке

Последующие слова относятся к управлению визуальными компонентами. Точка в имени слов не имеет специального назначения и не обрабатывается интерпретатором каким-либо особым образом. Она является элементом оформления синтаксиса, разделяя компоненты слова.

Ряд команд являются общими для многих визуальных компонентов, хотя и реализуются индивидуально для каждого типа компонента. Например, `SHOW` относится к включению видимости компонента, `HIDE` к выключению видимости, а `RECT` определяет координаты и размеры.

Для ряда компонентов применяется понятие «родительский компонент». В настоящее время родительским компонентом может быть панель (TPanel). В случае установки родительского компонента видимость будет определяться видимостью родительского компонента, а координаты отсчитываться от левого верхнего угла родительского компонента.

Перечень команд является временным и планируется к расширению.

LABEL.SHOW // index -- сделать видимой метку с индексом index

LABEL.HIDE // index -- сделать невидимой метку с индексом index

LABEL.RECT // x, y, width, height, index -- задать координаты и размеры метке с индексом index

LABEL.TEXT // str, index – поместить текст, заданный строкой str, на метку с индексом index

LABEL.FONT.SIZE // size, index – установить для метки с индексом index размер шрифта size

LABEL.FONT.NAME // str, index -- установить для метки с индексом index имя шрифта, заданное строкой str

LABEL.FONT.COLOR // color, index -- установить для метки с индексом index цвет шрифта, заданный параметром color

LABEL.PARENT.PANEL // panel, index -- установить для метки с индексом index номер родительской панели panel

LABEL.INT // x, index – поместить на метку с индексом index значение числа со стека x в текстовом формате

BUTTON.SHOW // index -- сделать видимой кнопку с индексом index

BUTTON.HIDE // index -- сделать невидимой кнопку с индексом index

BUTTON.RECT // x, y, width, height, index -- задать координаты и размеры кнопке с индексом index

BUTTON.TEXT // str, index – поместить текст, заданный строкой str, на кнопку с индексом index

BUTTON.ACTION // str, index – задать строку str для выполнения при нажатии на кнопку с индексом index

`BUTTON.PARENT.PANEL // panel, index` -- установить для кнопки с индексом `index` номер родительской панели `panel`

`BUTTON.DOWN // str, index` – задать строку `str` для выполнения при опускании (`down`) кнопки с индексом `index`

`BUTTON.UP // str, index` – задать строку `str` для выполнения при отпускании (`up`) кнопки с индексом `index`

Объект типа `TBitBtn` представляет собой кнопку с возможностью вывода на поверхности изображения. Объекты `TBitBtn` не эквивалентны объектам `TButton` и реализованы в виде отдельного массива. Таким образом, одновременно могут существовать кнопки `TButton` и `TBitBtn` с одинаковыми индексами.

`BITBUTTON.SHOW // index` -- сделать видимой кнопку с индексом `index`

`BITBUTTON.HIDE // index` -- сделать невидимой кнопку с индексом `index`

`BITBUTTON.RECT // x, y, width, height, index` -- задать координаты и размеры кнопке с индексом `index`

`BITBUTTON.TEXT // str, index` – поместить текст, заданный строкой `str`, на кнопку с индексом `index`

`BITBUTTON.ACTION // str, index` – задать строку `str` для выполнения при нажатии на кнопку с индексом `index`

`BITBUTTON.PARENT.PANEL // panel, index` -- установить для кнопки с индексом `index` номер родительской панели `panel`

`BITBUTTON.DOWN // str, index` – задать строку `str` для выполнения при опускании (`down`) кнопки с индексом `index`

`BITBUTTON.UP // str, index` – задать строку `str` для выполнения при отпускании (`up`) кнопки с индексом `index`

`BITBUTTON.IMAGE // str, index` – задать изображение, заданное именем файла `str` для поверхности кнопки с индексом `index`

`CHART.SHOW // index` -- сделать видимым график с индексом `index`

`CHART.HIDE // index` -- сделать невидимым график с индексом `index`

CHART.RECT // x, y, width, height, index -- задать координаты и размеры графику с индексом index

CHART.ADDSERIES // series, index – добавить серию series к графику с индексом index

CHART.ALIGN.NONE // index – отключить выравнивание графика с индексом index

CHART.ALIGN.CLIENT // index – установить выравнивание графика с индексом index по границам окна-клиента (при изменении размеров окна график увеличивается по размерам окна)

CHART.PARENT.PANEL // panel, index -- установить для графика с индексом index номер родительской панели panel

CHART.ADDPIESERIES – зарезервировано для будущих версий (в текущей реализации компонента TChart не поддерживается динамическое изменение типа серии на PieChart)

Объект TLineSeries используется в сочетании с объектом TChart. Каждая линия может принадлежать какому-либо графику. Подключение линий к графикам выполняется командой CHART.ADDSERIES. Последующие изменения подключенных линий приводит к соответствующему изменению ее отображения.

SERIES.CLEAR // index – очистить серию графика с индексом index

SERIES.XY // x, y, index – добавить точку с целочисленными координатами x, y к серии с индексом index

SERIES.FXY // index, f: x, y, – добавить точку с координатами x, y, заданными в формате с плавающей точкой, к серии с индексом index

SERIES.COLOR // color, index – задать цвет линии для отображения серии графика с индексом index

SERIES.LINEWIDTH // width, index – задать толщину линии для отображения серии графика с индексом index

PIESERIES.CLEAR – зарезервировано для будущих версий (в текущей реализации компонента TChart не поддерживается динамическое изменение типа серии на PieChart)

PIESERIES.X – зарезервировано для будущих версий (в текущей реализации компонента TChart не поддерживается динамическое изменение типа серии на PieChart)

PIESERIES.FX – зарезервировано для будущих версий (в текущей реализации компонента TChart не поддерживается динамическое изменение типа серии на PieChart)

EDIT.SHOW // index -- сделать видимым поле ввода с индексом index

EDIT.HIDE // index -- сделать невидимым поле ввода с индексом index

EDIT.RECT // x, y, width, height, index -- задать координаты и размеры полю ввода с индексом index

EDIT.SETTEXT // str, index -- установить текст из строки str для поля ввода с индексом index

EDIT.GETTEXT // str, index -- записать текст из поля ввода с индексом index в строку, заданную адресом str

EDIT.GETINT // index -- x – преобразовать содержимое поля ввода с индексом index в целое число и положить его на стек данных

EDIT.GETFLOAT // index – f: x – преобразовать содержимое поля ввода с индексом index в целое число и положить его на стек чисел с плавающей точкой

EDIT.PARENT.PANEL // panel, index -- установить для поля ввода с индексом index номер родительской панели panel

IMAGE.SHOW // index -- сделать видимым изображение (TImage) с индексом index

IMAGE.HIDE // index -- сделать невидимым изображение (TImage) с индексом index

IMAGE.RECT // x, y, width, height, index -- задать координаты и размеры изображению с индексом index

IMAGE.ALIGN.NONE // index – отключить выравнивание изображения с индексом index

IMAGE.ALIGN.CLIENT // index – установить выравнивание изображения с индексом index по границам окна-клиента (при изменении размеров окна график увеличивается по размерам окна)

IMAGE.PIXEL // x, y, color, index – установить для изображения с индексом index цвет пиксела с координатами x, y в значение color

IMAGE.LOAD // str, index -- загрузить в изображение с индексом index содержимое файла, заданного именем в строке str

IMAGE.PEN.WIDTH // width, index – установить ширину пера (pen) для изображения с индексом index

IMAGE.PEN.COLOR // color, index – установить цвет пера в значение color для изображения с индексом index

IMAGE.BRUSH.COLOR // color, index – установить цвет кисти в значение color для изображения с индексом index

IMAGE.LINE // x1, y1, x2, y2, index – провести линию из точки x1, y1 в точку x2, y2 для изображения с индексом index, используя параметры пера (pen)

IMAGE.BOX /// x1, y1, width, height, index – нарисовать закрашенный прямоугольник (используя параметры кисти) с координатами x1, y1, шириной width и высотой height для изображения с индексом index

IMAGE.CIRCLE // x, y, r, index -- нарисовать закрашенный круг (используя параметры кисти) с координатами центра x, y и радиусом r для изображения с индексом index

IMAGE.ELLIPSE // x, y, gx, gy, index -- нарисовать закрашенный эллипс (используя параметры кисти) с координатами центра x, y и размерами половины осей gx, gy для изображения с индексом index

IMAGE.TEXTXY // x, y, str, index – вывести текст, заданный строкой str, начиная с координат x, y для изображения с индексом index

IMAGE.FONT.SIZE // size, index – установить для изображения с индексом index размер шрифта size

IMAGE.FONT.NAME // str, index -- установить для изображения с индексом index имя шрифта, заданное строкой str

IMAGE.FONT.COLOR // color, index – установить для изображения с индексом index цвет шрифта color

IMAGE.PARENT.PANEL // panel, index -- установить для изображения с индексом index номер родительской панели panel

LISTBOX.SHOW // index -- сделать видимым список с индексом index

LISTBOX.HIDE // index -- сделать невидимым список с индексом index

LISTBOX.RECT // x, y, width, height, index -- задать координаты и размеры списку с индексом index

LISTBOX.CLEAR // index -- очистить список с индексом index

LISTBOX.ADD // str, index – добавить элемент, заданный строкой str, к списку с индексом index

LISTBOX.INDEX // index – id – вернуть на стеке данных номер выбранного элемента в списке с индексом index

LISTBOX.ADD.INT // x, index – добавить элемент, заданный целым числом x, к списку с индексом index

LISTBOX.PARENT.PANEL // panel, index -- установить для списка с индексом index номер родительской панели panel

STRINGGRID.SHOW // index -- сделать видимой таблицу с индексом index

STRINGGRID.HIDE // index -- сделать невидимой таблицу с индексом index

STRINGGRID.RECT // x, y, width, height, index -- задать координаты и размеры таблице с индексом index

STRINGGRID.COLS // cols, index – установить число колонок cols таблице с индексом index

STRINGGRID.ROWS // rows, index – установить число строк rows таблице с индексом index

STRINGGRID.TEXT // str, col, row, index – установить текст, заданный строкой str, для ячейки таблицы с индексом index, заданной координатами col, row

STRINGGRID.INT // x, col, row, index – установить текст, преобразованный из целого числа x, для ячейки таблицы с индексом index, заданной координатами col, row

STRINGGRID.FLOAT // col, row, index, f:x – установить текст, преобразованный из числа с плавающей точкой x, для ячейки таблицы с индексом index, заданной координатами col, row

STRINGGRID.COLS.FIXED // cols, index – установить число фиксированных колонок cols таблице с индексом index

STRINGGRID.ROWS.FIXED // rows, index – установить число фиксированных строк rows таблице с индексом index

STRINGGRID.EDIT // flag, index – установить для таблицы с индексом index флаг «разрешить редактирование» в значение flag

STRINGGRID.GETTEXT // str, col, row, index – записать текст из ячейки таблицы с индексом index, заданной координатами col, row, в строку с адресом str

STRINGGRID.GETINT - зарезервировано

STRINGGRID.GETFLOAT - зарезервировано

STRINGGRID.LOAD // str, index – попытка загрузить в таблицу с индексом str содержимого текстового файла, заданного именем str, не затрагивая фиксированные колонки и строки, каждая строка загружается в одну ячейку

STRINGGRID.LOAD.CSV - зарезервировано

STRINGGRID.SAVE - зарезервировано

STRINGGRID.SAVE.CSV - зарезервировано

STRINGGRID.GETCOLS - зарезервировано

STRINGGRID.GETROWS - зарезервировано

STRINGGRID.PARENT.PANEL // panel, index -- установить для таблицы с индексом index номер родительской панели panel

TRACKBAR.SHOW // index -- сделать видимым ползунок с индексом index

TRACKBAR.HIDE // index -- сделать невидимым ползунок с индексом index

TRACKBAR.RECT // x, y, width, height, index -- задать координаты и размеры ползунку с индексом index

TRACKBAR.ACTION // str, index – установить строку str для выполнения при изменении положении ползунка с индексом index

TRACKBAR.MIN // min, index – установить минимальное значение для ползунка с индексом index

TRACKBAR.MAX // max, index – установить максимальное значение для ползунка с индексом index

TRACKBAR.GETPOSITION // index – position – поместить на стек текущую позицию ползунка с индексом index

TRACKBAR.STEP // step, index – установить шаг между рисками для ползунка с индексом index

TRACKBAR.VERTICAL // index – установить для ползунка с индексом index вертикальную ориентацию

TRACKBAR.HORIZONTAL // index – установить для ползунка с индексом index горизонтальную ориентацию

TRACKBAR.PARENT.PANEL // panel, index -- установить для ползунка с индексом index номер родительской панели panel

PROGRESSBAR.SHOW // index -- сделать видимым индикатор с индексом index

PROGRESSBAR.HIDE // index -- сделать невидимым индикатор с индексом index

PROGRESSBAR.RECT // x, y, width, height, index -- задать координаты и размеры индикатору с индексом index

PROGRESSBAR.MIN // min, index – установить минимальное значение для индикатора с индексом index

PROGRESSBAR.MAX // max, index – установить максимальное значение для индикатора с индексом index

PROGRESSBAR.SETPOSITION // position, index – установить позицию для индикатора с индексом index

PROGRESSBAR.PARENT.PANEL // panel, index -- установить для индикатора с индексом index номер родительской панели panel

CHECKBOX.SHOW // index -- сделать видимым переключатель с индексом index

CHECKBOX.HIDE // index -- сделать невидимым переключатель с индексом index

CHECKBOX.RECT // x, y, width, height, index -- задать координаты и размеры переключателя с индексом index

CHECKBOX.GETSTATE // index – state – поместить на стек логическое значение, отражающее текущее состояние переключателя с индексом index

CHECKBOX.PARENT.PANEL // panel, index -- установить для переключателя с индексом index номер родительской панели panel

TOOLBUTTON.SHOW – не реализовано, зарезервировано

TOOLBUTTON.HIDE – не реализовано, зарезервировано

TOOLBUTTON.ADDGLYPH – не реализовано, зарезервировано

TOOLBUTTON.TEXT – не реализовано, зарезервировано

UART.SETPORT // port, index – установить для компонента COM-порта с индексом index номер port

UART.OPEN // index – t|f – открыть COM-порт, параметры которого определяются компонентом с индексом index, вернуть ИСТИНУ, если порт открыт успешно или 0, если порт не открыт

UART.CLOSE // index -- закрыть COM-порт, параметры которого определяются компонентом с индексом index

UART.BAUDRATE // baudrate, index – установить для компонента COM-порта с индексом index скорость передачи baudrate

UART.WRITE // byte, index – записать байт byte в COM-порт, параметры которого определяются компонентом с индексом index

UART.READ // index – byte - прочитать байт byte из COM-порта, параметры которого определяются компонентом с индексом index

UART.SEND // str, index -- записать строку str в COM-порт, параметры которого определяются компонентом с индексом index

UART.READLN // str, index – прочитать строку из COM-порта, параметры которого определяются компонентом с индексом index, разместив ее начиная с адреса str

COMBOBOX.SHOW // index -- сделать видимым выпадающий список с индексом index

COMBOBOX.HIDE // index -- сделать невидимым выпадающий список с индексом index

COMBOBOX.RECT // x, y, width, height, index -- задать координаты и размеры выпадающего списка с индексом index

COMBOBOX.CLEAR // index -- очистить выпадающий список с индексом index

COMBOBOX.ADD // str, index -- добавить строку str к выпадающему списку с индексом index

COMBOBOX.INDEX // index -- id – вернуть номер id выбранного элемента выпадающего списка с индексом index

COMBOBOX.ADD.INT // x, index -- добавить элемент x, взятый со стека данных как целое число, к выпадающему списку с индексом index

COMBOBOX.PARENT.PANEL // panel, index -- установить для переключателя с индексом index номер родительской панели panel

PANEL.SHOW // index -- сделать видимой панель с индексом index

PANEL.HIDE // index -- сделать невидимой панель с индексом index

PANEL.RECT // x, y, width, height, index -- задать координаты и размеры панели с индексом index

PANEL.ALIGN.LEFT // index – установить выравнивание панели с индексом index по левому краю окна-клиента

PANEL.ALIGN.RIGHT // index – установить выравнивание панели с индексом index по правому краю окна-клиента

PANEL.ALIGN.TOP // index – установить выравнивание панели с индексом index по верхнему краю окна-клиента

PANEL.ALIGN.BOTTOM // index – установить выравнивание панели с индексом index по нижнему краю окна-клиента

PANEL.ALIGN.CLIENT // index – установить выравнивание панели с индексом index по границам окна-клиента (при изменении размеров окна панель увеличивается по размерам окна)

PANEL.ALIGN.NONE // index – отключить выравнивание панели с индексом index

PANEL.COLOR // color, index – установить для панели с индексом index цвет color

SPLITTER.SHOW // index -- сделать видимым сплиттер с индексом index

SPLITTER.HIDE // index -- сделать невидимым сплиттер с индексом index

SPLITTER.ALIGN.LEFT // index – установить выравнивание сплиттера с индексом index по левому краю окна-клиента

SPLITTER.ALIGN.RIGHT // index – установить выравнивание сплиттера с индексом index по правому краю окна-клиента

SPLITTER.ALIGN.TOP // index – установить выравнивание сплиттера с индексом index по верхнему краю окна-клиента

SPLITTER.ALIGN.BOTTOM // index – установить выравнивание сплиттера с индексом index по нижнему краю окна-клиента

RADIOGROUP.SHOW // index -- сделать видимой группу радиокнопок с индексом index

RADIOGROUP.HIDE // index -- сделать невидимой группу радиокнопок с индексом index

RADIOGROUP.RECT // x, y, width, height, index -- задать координаты и размеры группы радиокнопок с индексом index

RADIOGROUP.PARENT.PANEL // panel, index -- установить для переключателя с индексом index номер родительской панели panel

RADIOGROUP.CLEAR // index -- очистить группу радиокнопок с индексом index

RADIOGROUP.ADD // str, index -- добавить кнопку со строкой str к группе радиокнопок с индексом index

RADIOGROUP.ADD.INT // x, index -- добавить кнопку со строкой, определяемой числом x со стека данных, к группе радиокнопок с индексом index

RADIOGROUP.INDEX // index -- id – вернуть номер id выбранной радиокнопки из группы радиокнопок с индексом index

MEMO.SHOW // index -- сделать видимым поле редактирования текста с индексом index

MEMO.HIDE // index -- сделать невидимым поле редактирования текста с индексом index

MEMO.RECT // x, y, width, height, index -- задать координаты и размеры поля редактирования текста с индексом index

MEMO.PARENT.PANEL // panel, index -- установить для переключателя с индексом index номер родительской панели panel

MEMO.CLEAR // index -- очистить поле редактирования текста с индексом index

Компонент Sprite является вариантом компонента «изображение» и используется для размещения на других изображениях.

SPRITE.LOAD // str, index -- загрузить спрайт с индексом index из файла, имя которого определяется строкой str

SPRITE.DRAW // index, x, y – вывести спрайт с индексом index в активное изображение

SPRITE.IMAGE // index – задать индекс активного изображения, в которое будут выводиться спрайты.

Компонент OpenGL представлен в программе в единственном экземпляре, поэтому для него не используется параметр index. Общим принципом для реализации слов являлось копирование синтаксиса и порядка следования аргументов из руководства по библиотеке OpenGL. Слова, типичные для Irbis Forth, реализованы с тем же синтаксисом, за исключением параметра index.

OPENGL.SHOW – сделать компонент OpenGL видимым

OPENGL.HIDE – сделать компонент OpenGL невидимым

OPENGL.RECT // x, y, w, h – задать координаты и размеры компоненты OpenGL

OPENGL.PARENT.PANEL // panel -- установить для компонента OpenGL номер родительской панели panel

Следующие слова являются аналогами соответствующих функций библиотеки OpenGL и имеют тот же порядок аргументов и тип результата. Перечень поддерживаемых функций будет расширяться, поэтому текущий список дается в качестве справки.

OPENGL.END

GLEND

OPENGL.POINTS

OPENGL.POINT

GLPOINTS

OPENGL.LINES

OPENGL.LINE

GLLINES

OPENGL.TRIANGLES
OPENGL.TRIANGLE
GLTRIANGLE
OPENGL.QUADS
OPENGL.QUAD
GLQUADS
OPENGL.VERTEX3D
OPENGL.COLOR3I
OPENGL.COLOR3F
OPENGL.UPDATE
OPENGL.LOADIDENTITY
OPENGL.ROTATEF
OPENGL.TRANSLATEF
OPENGL.PUSHMATRIX
OPENGL.POPMATRIX
OPENGL.LOOKAT
OPENGL.ENABLE
GLENABLE
OPENGL.DISABLE
GLDISABLE
OPENGL.LIGHTFV
OPENGL.COLORMATERIAL
OPENGL.TEXCOORD2F
OPENGL.BINDTEXTURE
OPENGL.NEWLIST
OPENGL.ENDLIST
OPENGL.CALLLIST
OPENGL.CLEAR
OPENGL.CULLFACE.FRONT
OPENGL.FRONT.CW

OPENGL.FRONT.CCW

OPENGL.MATRIX.MODELVIEW

OPENGL.ORTHO

OPENGL.CLEARCOLOR

OPENGL.ALIGN.LEFT – установить для компонента OpenGL
выравнивание по левому краю родительского компонента

OPENGL.ALIGN.RIGHT – установить для компонента OpenGL
выравнивание по правому краю родительского компонента

OPENGL.ALIGN.TOP – установить для компонента OpenGL
выравнивание по верхнему краю родительского компонента

OPENGL.ALIGN.BOTTOM – установить для компонента OpenGL
выравнивание по нижнему краю родительского компонента

OPENGL.ALIGN.CLIENT – установить для компонента OpenGL
выравнивание по размерам родительского компонента

OPENGL.ALIGN.NONE – отключить для компонента OpenGL
выравнивание по границам родительского компонента

MESSAGE.OK // str – показать окно сообщения со строкой str и кнопкой
ОК

MESSAGE.OKCANCEL // str – id - показать окно сообщения со строкой
str и кнопками «ОК», «Отмена», вернуть на стек данных номер нажатой
кнопки

MESSAGE.YESNO // str – id - показать окно сообщения со строкой str и
кнопками «Да», «Нет», вернуть на стек данных номер нажатой кнопки

MESSAGE.YESNOCANCEL // str – id - показать окно сообщения со
строкой str и кнопками «Да», «Нет», «Отмена», вернуть на стек данных номер
нажатой кнопки

DIALOG.OPEN // str -- t|f – показать диалоговое окно выбора файла, если
нажата кнопка ОК, имя выбранного файла записывается в строку с адресом str,
а на стек данных помещается ИСТИНА, если нажата кнопка «Отмена», на стек
данных помещается ЛОЖЬ (0)

DIALOG.SAVE // str -- t|f – показать диалоговое окно файла, если нажата кнопка ОК, имя выбранного файла записывается в строку с адресом str, а на стек данных помещается ИСТИНА, если нажата кнопка «Отмена», на стек данных помещается ЛОЖЬ (0)

SOUND.PLAY // str – начать проигрывание звукового файла, заданного именем str, вернуть управление программе сразу после начала проигрывания

MIDI.AVAILABLE // -- t|f – вернуть на стеке данных ИСТИНУ, если доступно звуковое устройство для проигрывания MIDI

MIDI.OPEN – открыть MIDI-устройство

MIDI.NOTE.ON // Note, Instrument, Volume, Channel, Panning – начать проигрывание ноты Note (0-127) инструмента Instrument (0-127) с громкостью Volume (0-127) по каналу Channel (0-15) с панорамированием Panning (0-127), см. руководства по MIDI-устройствам для подробной информации

MIDI.NOTE.OFF // Note, Channel – остановить проигрывание ноты Note по каналу Channel

MIDI.PERCUSSION.ON – не реализовано, зарезервировано

MIDI.PERCUSSION.OFF – не реализовано, зарезервировано

MIDI.ALL.OFF – отключить проигрывание звуков по всем каналам MIDI

MIDI.CLOSE – закрыть MIDI-устройство

FORM.COLOR // color – установить цвет фона для формы

FORM.EDITOR.ON – сделать компонент встроенного редактора видимым

FORM.EDITOR.OFF – сделать компонент встроенного редактора невидимым

FORM.CONSOLE.ON – сделать компонент консоли видимым

FORM.CONSOLE.OFF – сделать компонент консоли невидимым

FORM.TOOLBAR.ON – сделать компонент панели кнопок видимым

FORM.TOOLBAR.OFF – сделать компонент панели кнопок невидимым

FORM.ALL.ON – сделать все встроенные компоненты видимыми

FORM.ALL.OFF – сделать все встроенные компоненты невидимыми

SYNEDIT.SHOW // index -- сделать видимым редактор текста с индексом index

SYNEDIT.HIDE // index -- сделать невидимым редактор текста с индексом index

SYNEDIT.RECT // x, y, width, height, index -- задать координаты и размеры редактора текста с индексом index

SYNEDIT.PARENT.PANEL // panel, index -- установить для редактора с индексом index номер родительской панели panel

SYNEDIT.CLEAR // index -- очистить содержимое редактора текста с индексом index

SYNEDIT.ALIGN.LEFT // index -- -- установить для редактора текста с индексом index выравнивание по левому краю родительского компонента

SYNEDIT.ALIGN.RIGHT // index -- -- установить для редактора текста с индексом index выравнивание по правому краю родительского компонента

SYNEDIT.ALIGN.TOP // index -- -- установить для редактора текста с индексом index выравнивание по верхнему краю родительского компонента

SYNEDIT.ALIGN.BOTTOM // index -- -- установить для редактора текста с индексом index выравнивание по нижнему краю родительского компонента

SYNEDIT.ALIGN.CLIENT // index -- -- установить выравнивание редактора текста с индексом index по границам окна-клиента (при изменении размеров окна панель увеличивается по размерам окна)

SYNEDIT.ALIGN.NONE // index -- -- отключить выравнивание редактора текста с индексом index

SYNEDIT.LINES // index -- lines -- вернуть на стеке количество строк в редакторе текста с индексом index

SYNEDIT.GETLINE // str, linenumber, index -- прочитать строку с номером linenumber из редактора текста с индексом index и поместить ее в строку, заданную адресом str

SYNEDIT.SETLINE // str, linenumber, index -- записать строку, заданную адресом str, в строку с номером linenumber в редакторе текста с индексом index

`SYNEDIT.FONT.SIZE // size, index` – установить размер текста `size` для редактора текста с индексом `index`

`SYNEDIT.LOAD // str, index` – загрузить содержимое редактора текста из файла, заданного именем `str`

`SYNEDIT.SAVE // str, index` – сохранить содержимое редактора текста в файл, заданный именем `str`

`OPENGL3D` – виртуальное слово, выполняется каждый раз при необходимости перерисовки компонента OpenGL, не влияет на стек данных

`TIMER.INTERVAL` – интервал запуска таймера в миллисекундах

`MOUSE.MOVE` – виртуальное слово, выполняется при перемещении мыши

`MOUSE.LEFT` – виртуальное слово, выполняется при нажатии левой кнопки мыши

`MOUSE.LEFT.DOWN` – виртуальное слово, выполняется при опускании левой кнопки мыши

`MOUSE.LEFT.UP` – виртуальное слово, выполняется при отпускании левой кнопки мыши

`MOUSE.RIGHT` – виртуальное слово, выполняется при нажатии правой кнопки мыши

`MOUSE.RIGHT.DOWN` – виртуальное слово, выполняется при опускании правой кнопки мыши

`MOUSE.RIGHT.UP` – виртуальное слово, выполняется при отпускании правой кнопки мыши

`MOUSE.X // -- x` – поместить на стек координату `x` указателя мыши

`MOUSE.Y // -- y` – поместить на стек координату `y` указателя мыши

`KEY.DOWN` // виртуальное слово, выполняется при опускании клавиши

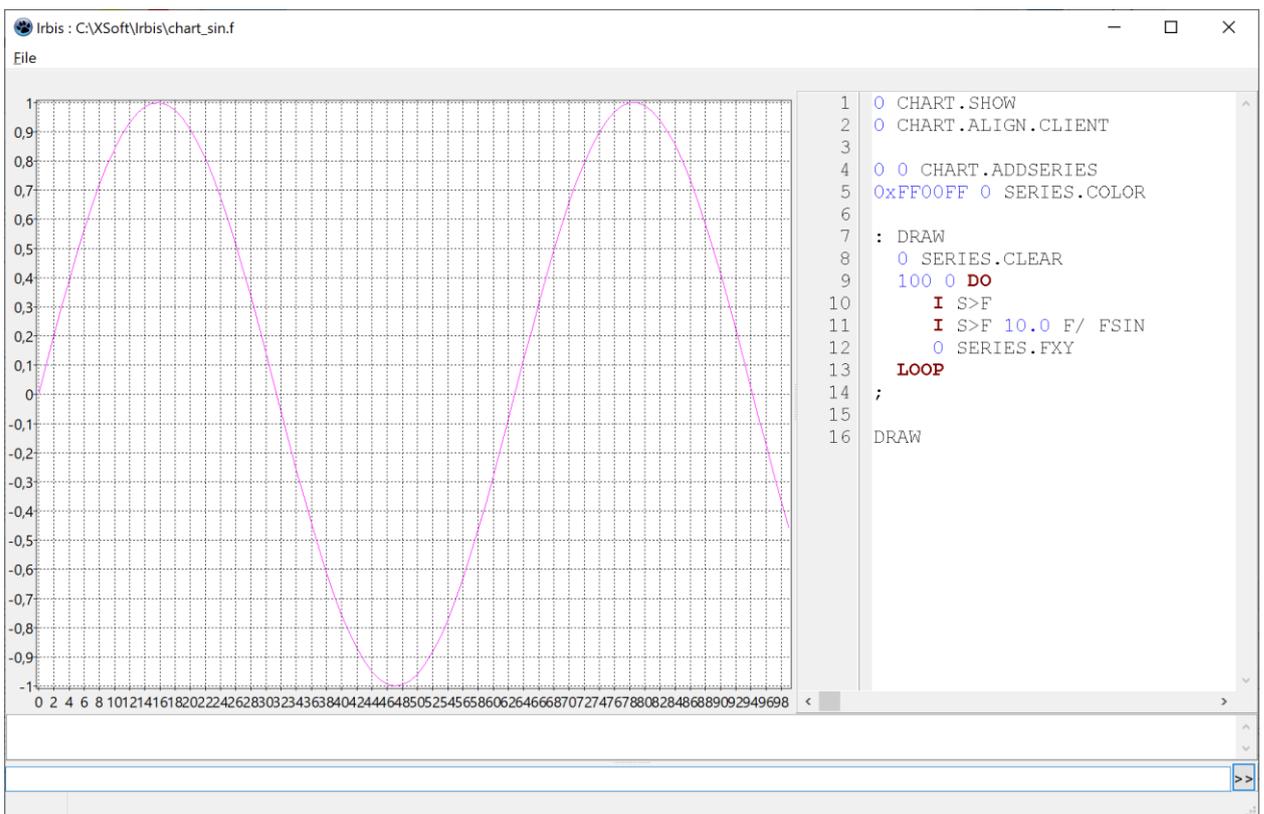
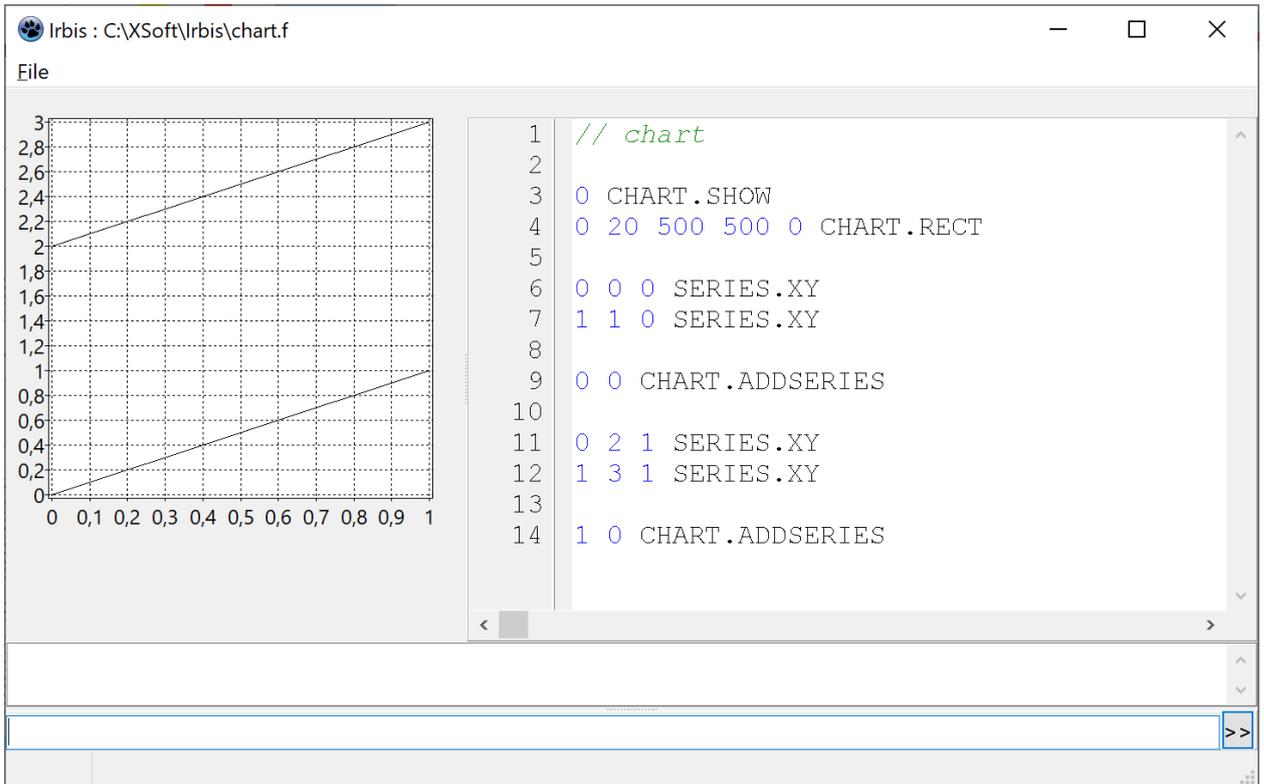
`KEY.CODE // -- code` – поместить на стек код нажатой клавиши

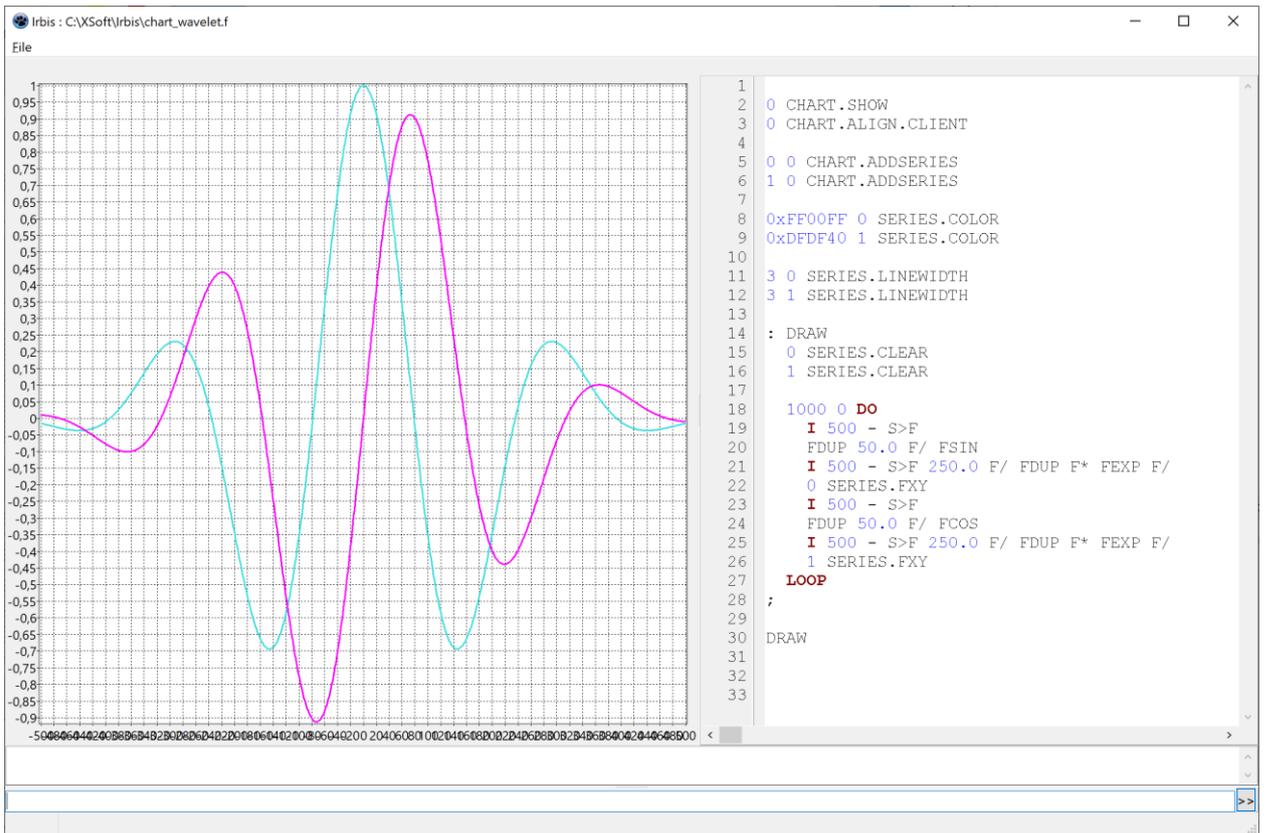
В текущей версии заданы следующие количества предопределенных компонентов:

MAXLABELS = 100;
MAXBUTTONS = 100;
MAXEDITS = 100;
MAXCHARTS = 10;
MAXLINESERIES = 100;
MAXPIESERIES = 50;
MAXCOLORMAPSERIES = 10;
MAXIMAGES = 20;
MAXTRACKBARS = 50;
MAXPROGRESSBARS = 20;
MAXCHECKBOXES = 50;
MAXTOOLBUTTONS = 20;
MAXLISTBOXES = 20;
MAXSTRINGGRIDS = 20;
MAXUARTS = 5;
MAXCOMBOBOXES = 20;
MAXPANELS = 20;
MAXSPLITTERS = 20;
MAXRADIOGROUPS = 20;
MAXMEMOS = 20;
MAXSYNEDITS = 20;
MAXSPRITES = 100;

MAXTEXTFILES = 16;
MAXBINFILES = 16;

4. Примеры





Irbitis : C:\XSoft\Irbitis\combobox.f

```

1
2 0 COMBOBOX.SHOW
3 50 50 100 120 0 COMBOBOX.RECT
4 \" First item\" 0 COMBOBOX.ADD
5 \" Second item\" 0 COMBOBOX.ADD
6 3 0 COMBOBOX.ADD.INT

```

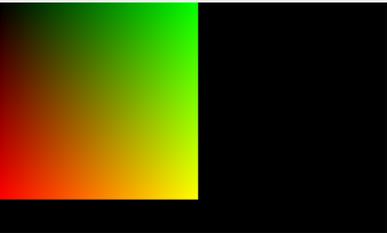
Irbis : C:\Soft\Irbis\dll_demo2.f

```
File
1 int hlib
2 int hfunc
3
4 " user32.dll" loadlibrary to hlib
5 hlib " MessageBoxA" getprocaddress to hfunc
6
7 0 " 111" data[]+ " 222" data[]+ 0 hfunc invoke4
```

1: 1

Irbis : C:\Soft\Irbis\image.f

```
File
1 0 IMAGE.SHOW
2
3 : TEST
4 0 IMAGE.HIDE
5 256 0 DO
6 256 0 DO
7 I J
8 I 256 * J + 0 IMAGE.PIXEL
9 LOOP
10 LOOP
11 0 IMAGE.SHOW
12 ;
13 TEST
14
```



Irbitis : C:\XSoft\Irbitis\mouse.f

File

143 275

```
1
2 0 LABEL.SHOW
3 10 40 50 30 0 LABEL.RECT
4 1 LABEL.SHOW
5 110 40 50 30 1 LABEL.RECT
6
7 : MOUSING
8     MOUSE.X 0 LABEL.INT
9     MOUSE.Y 1 LABEL.INT
10 ;
11 USE MOUSING TO MOUSE.MOVE
12
13
```

>>

Irbitis : C:\XSoft\Irbitis\progressbar.f

File



```
1
2 0 PROGRESSBAR.SHOW
3 10 30 200 20 0 PROGRESSBAR.RECT
4
5 : TEST
6     100 0 DO
7         I 0 PROGRESSBAR.SETPOSITION
8         1000000 0 DO LOOP
9     LOOP
10 ;
11
12 TEST
```

>>

Irbis : C:\XSoft\Irbis\rosetta_aplusb.f

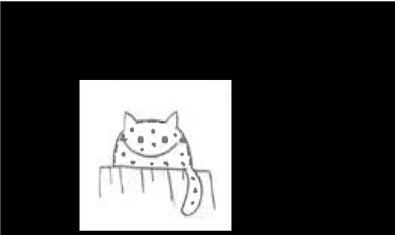
File

```
1 0 EDIT.SHOW
2 50 100 75 25 0 EDIT.RECT
3
4 1 EDIT.SHOW
5 150 100 75 25 1 EDIT.RECT
6
7 0 BUTTON.SHOW
8 100 150 100 30 0 BUTTON.RECT
9 "+" 0 BUTTON.TEXT
10
11 " 0 EDIT.GETFLOAT 1 EDIT.GETFLOAT F+ F." 0 BUTTON.ACTION
12
13
```

5

Irbis : C:\XSoft\Irbis\sprite.f

File



```
1 0 SPRITE.IMAGE
2 0 IMAGE.SHOW
3 " irbis_avatar.png" 0 SPRITE.LOAD
4
5 0 100 100 SPRITE.DRAW
6
7 variable x
8 variable y
9
10 100 x !
11 100 y !
12
13
14 : keys
15 KEY.CODE 37 = if
16   x @ 1 - x !
17   0 x @ y @ SPRITE.DRAW
18 then
19 KEY.CODE 39 = if
20   x @ 1 + x !
21   0 x @ y @ SPRITE.DRAW
22 then
23
24
25 ;
26 use keys to key.down
```

Irbis : C:\XSoft\Irbis\timer1.f

File

8

```
1
2 0 label.show
3 variable x
4 variable pressed
5
6 : mleftdown
7   1 pressed !
8 ;
9 use mleftdown to MOUSE.LEFT.DOWN
10
11 : mleftup
12   0 pressed !
13 ;
14 use mleftup to MOUSE.LEFT.UP
15
16 : tim
17   pressed @ if
18     x @ 1 + x !
19     x @ 0 label.int
20   then
21 ;
22 100 TIMER.INTERVAL
23 use tim to timer
```

>>



Заключение

Данное описание соответствует прототипу программы, которой присвоен номер версии 0.1. Предполагается дальнейшее расширение функциональных возможностей, добавление новых компонентов и новых методов для реализованных компонентов, развитие поддержки компонента OpenGL и добавление средств диагностики ошибок для повышения устойчивости программы к вводу неверных данных.

Версии документа

1. 20.04 – Начальная реализация документа, соответствует v0.1 build 5.