Инструкция по программированию на языке ForthLogic™

2011

Содержание

Программирование на языке ForthLogic™

Вступление

Язык программирования Форт (англ. forth вперед и одновременное сокращение от fourth четвертый), который лежит в основе языка ForthLogic™, появился в начале 1970-х гг. в США. Его изобретатель Чарльз Мур сначала применил его для разработки программного обеспечения микро-ЭВМ, которая управляла работой радиотелескопа. Преимущества работы с языком Форт были настолько большими, что вскоре его начали использовать и на других специализированных ЭВМ.
Эффективность применения языка Форт подтверждается тем, что он используется наиболее известными грандами индустрии: корпорация Boeing использует встроенный интерпретатор языка Форт в системе авионики (бортового оборудования) самолета Boeing 777; корпорация Tektronix использует язык Форт для серии анализаторов сетевых протоколов K1297 и K1205; корпорация Lockheed Martin использует язык Форт в бортовом оборудовании наземной телеметрической системы SMART для баллистической ракеты Trident 2 D5; корпорация FedEx использует ручной считыватель штрих-кодов SuperTracker со встроенной системой Форт в своем программно-аппаратном комплексе электронного контроля посылок COSMOS II; корпорация General Electric использует язык Форт для серии SONET -коммутаторов JungleMUX; корпорация Europay использует язык Форт при создании программного обеспечения для универсальных кассовых терминалов/считывателей смарт-карт с архитектурой Open Terminal Architecture (OTA); корпорация Sun Microsystems с 1989 г. использует загрузчик OpenBoot (программа типа BIOS) со встроенным интерпретатором языка Форт в своих компьютерах SparcStation и серверах SPARCServer, а корпорация Apple Inc. использует аналогичный загрузчик Open Firmware в своих компьютерах Power Macintosh. Кроме того, язык Форт является стандартным языком управления оборудованием телескопов как на земле так и в космосе.

Основы

Синтаксис языка ForthLogic™ максимально простой. Запись каждой инструкции (команды) состоит из одного слова, в качестве которого может выступать последовательность любых символов, которая не содержит пробелов. В языке ForthLogic™ такие инструкции (команды) так и называются - слова. Простота синтаксиса является следствием того, что в качестве вычислительной модели используется стековая машина. В большинстве случаев слова-команды этой машины снимают необходимые операнды со стека и оставляют свои результаты (если они есть) также на стеке. Таким образом, программа, написанная на языке ForthLogic™, выглядит как последовательность слов, каждое из которых имеет ввиду выполнение тех или других действий. Слова разделяются любым числом пробелов; ограничение накладывается только на длину слова - оно должно содержать не больше 14 символов. Стандартно определенно сравнительно

небольшой набор "встроенных" слов. Среди них есть слова, которые позволяют определять новые через уже существующие и тем самым расширять начальный набор слов-команд в нужном для данного задания направлении.
Вычислительная модель, которая лежит в основе языка ForthLogic™, состоит из стека данных, стека чисел с плавающей запятой, глобальных переменных, глобальных переменных в формате чисел с плавающей запятой, глобальных битовых переменных, глобальных строчных переменных, словаря и выходного буфера для вывода результатов на терминал или обмена строчными данными между словами. Язык ForthLogic™ позволяет описывать параллельно выполняемые процессы и функционирует в многозадачной среде.
Стек данных (далее по тексту - либо *стек* либо *стек данных*) расположен в оперативной памяти и используется для передачи числовых параметров и результатов между словами, выполняемыми в пределах одной задачи. Его элементами являются четырехбайтные значения, которые интерпретируются как целые числа со знаком в диапазоне от - 2147483648 до +2147483647. В процессе выполнения слов значения помещаются на стек и снимаются с него. Переполнение и исчерпание стека проверяется и сообщается как ошибка; его максимальный объем равняется 16 элементам.
Стек чисел с плавающей запятой (далее по тексту - *математический стек*) также расположен в оперативной памяти и используется для математических вычислений над числами с плавающей запятой. Его элементами являются четырехбайтные представления чисел с плавающей запятой одинарной точности согласно стандарта IEEE-754, которые могут принимать значение в диапазоне $\pm(1{,}4x10^{-45}...3{,}4x10^{38})$. В процессе выполнения математических операций над числами с плавающей запятой, значения помещаются на математический стек и снимаются с него. Переполнение и исчерпание математического стека проверяется и сообщается как ошибка; его максимальный объем равняется 16 элементам.
Глобальные переменные являются обычными статичными переменными, которые расположены в оперативной памяти. Их элементами являются четырехбайтные значения, которые рассматриваются как целые числа со знаком. Глобальные переменные обычно используются для передачи числовых параметров и результатов целочисленных вычислений между словами выполняемыми в разных задачах или для долговременного хранения целочисленных параметров и результатов в рамках одной задачи. Количество глобальных переменных зависит от аппаратной платформы.
Глобальные переменные в формате чисел с плавающей запятой (далее по тексту - *математические переменные*) являются статичными переменными, которые расположены в оперативной памяти. Их элементами являются четырехбайтные представления чисел с плавающей запятой одинарной точности согласно стандарта IEEE-754. Математические переменные обычно используются для хранения промежуточных математических значений и результатов вычислений. Количество глобальных математических переменных зависит от аппаратной платформы.
Глобальные битовые переменные являются статичными переменными, которые расположены в оперативной памяти. Их элементами являются однобитовые числа которые принимают значение 0 или 1. Битовые переменные обычно используются в качестве разнообразных флажков и в задачах логического управления - они естественным образом согласуются с унитарными сигналами контроллера. Количество глобальных битовых переменных зависит от аппаратной платформы.

Глобальные строчные переменные являются статичными массивами длиной 15 элементов каждый, которые расположены в оперативной памяти. Их элементами являются однобайтные значения, которые представляют коды символов. Строчные переменные предназначены для оперативного хранения строк текста длиной до 15 символов, в которых могут присутствовать пробелы. Количество глобальных строчных переменных зависит от аппаратной платформы.
Начальную часть энергонезависимой памяти занимает *словарь* - хранилище слов, констант и строчных данных. По мере расширения начального набора слов, словарь растет в сторону увеличения адресов. Существуют специальные слова, которые позволяют работать со словарем (например, определять свободное место в словаре, удалять слова, и тому подобное). Объем энергонезависимой памяти и размер начального словаря зависит от аппаратной платформы.
Выходной буфер является обычным статичным буфером, который расположен в оперативной памяти. Его элементами являются однобайтные значения, которые представляют коды символов. В буфере располагаются строчные данные для обмена с терминалом или между отдельными словами. Размер буфера зависит от аппаратной платформы.

Работа в диалоговом режиме

Программирование на языке ForthLogic™ является диалоговым процессом. Работая за терминалом, пользователь вводит слова-команды, а форт-система, то есть программно-аппаратная реализация языка ForthLogic™, немедленно производит действия, которые обозначаются этими словами - это терминальный режим работы форт-системы. О своей готовности к обработке дежурной строки текста форт-система информирует пользователя приглашениям ">", которое печатается на терминале. Получив такое приглашение, пользователь набирает на терминале дежурную порцию текста, заканчивая ее клавишей "Enter". Получив сигнал о завершении ввода, форт-система начинает обработку введенного текста (он размещается в специальном, программно-недоступном входном буфере для ввода из терминала), выделяя в нем слова-команды и выполняя их. Успешно обработав весь введенный текст, форт-система опять приглашает пользователя к вводу, и описанный цикл диалога повторяется. После успешного завершения обработки введенного текста, форт-система выводит на терминал подтверждающее сообщение (OK) (от английского o'kay - "все в порядке"). Если во время обработки введенного текста встречается ошибка (например, встретилось неизвестное форт-системе слово), то на терминал выводится объясняющее сообщение, обработка введенного текста прекращается и форт-система приглашает пользователя к вводу нового текста. Известные форт-системе ошибки и их коды описаны в приложении.
Максимальная длина строки текста, которая принимается для обработки, составляет 80 символов. При работе с терминалом, встроенный редактор форт-системы автоматически ограничивает длину строки до 77 символов и извещает об этом звуковым сигналом. При наборе дежурной порции текста, кроме клавиш букв и цифр, допускается применять лишь клавиши "Backspace", "Space"(пробел) и "Enter".
Под словом терминал, мы понимаем любую из доступных программ, которые осуществляют эмуляцию терминала. В средах разных операционных систем таких программ есть очень много, причем как платных так и бесплатных - процедура настройки программы-терминала в среде Microsoft® Windows®XP описана в

приложении.
Непосредственно вводить с терминала большие тексты неудобно, поэтому их сохраняют в текстовые файлы и с помощью карты памяти SD/MMC переносят в форт-систему (процедура создания файлов и переноса их на карту памяти SD/MMC описана в приложении). Также текстовый файл можно перенести в форт-систему с помощью протокола Xmodem непосредственно с терминала - данная процедура также описана в приложении.
Контроллеры без встроенного дисплея имеют специальный режим отображения статусной информации о системе в окне терминала. Для входа и выхода из этого режима существуют соответственно команды SHOW STATUS и HIDE STATUS, которые следует вводить с новой строки терминала без предварительных пробелов.

Стек данных и вычисления

Как уже упоминалось, в основе вычислительной модели языка ForthLogic™ лежит стековая машина. На данный момент, форт-система содержит два программно доступных стека: стек данных и математический стек. Механизмы работы этих стеков похожи - разница заключается в представлении данных и в наборе слов для работы с данными в этих стеках.
Стек представляет собой программный буфер данных, который функционирует по принципу "последний зашел - первый вышел". Стек является единственным местом проведения математических и логических вычислений и является чрезвычайно простым и надежным механизмом автоматической передачи числовых параметров и данных между отдельными словами. В качестве аналогии со стеком можно привести колоду карт: положить новое значение на стек можно сравнить с тем, как мы кладем новую карту на верх колоды; снять число со стека аналогично действию, когда мы снимаем карту с вершины колоды. Слова на языке ForthLogic™ обычно используют в качестве операндов верхние элементы стеков, убирая их со стека и возвращая результаты (если они есть) на место операндов. Как правило, слова используют одно-два верхних значения на стеке. Для их описания будем применять следующую диаграмму стековой нотации и цветное выделение (это касается обоих стеков):

```
имя      вершина стека до  --->  вершина стека после
слова    исполнение слова        исполнение слова
```

При этом считаем, что само верхнее значение в стеке (то, которое было добавлено последним) находится справа. Дополнительно, перед значениями, которые находятся на математическом стеке, будем указывать тип стека с помощью буквы F и двоеточия.
Для работы с вершиной стека данных существуют следующие слова:

```
DUP       A          ---> A, A
DROP      A          ---> -
OVER      A,B        ---> A,B,A
ROT       A,B,C      ---> B,C,A
SWAP      A,B        ---> B,A
```

Слово DUP (от DUPLICATE - дублировать) дублирует вершину стека данных, добавляя в стек еще одно значение, равное тому, которое было до этого верхним. Слово DROP (сбросить) убирает верхнее значение. Слово OVER (через) дублирует значение, которое расположено на стеке данных непосредственно под верхним. Слово ROT (от ROTATE вращать) циклически переставляет по часовой стрелке (если смотреть на диаграмму) три верхних значения в стеке данных. Слово SWAP (обменять) меняет местами два верхних значения.
Также можно работать с любым элементом стека данных с помощью слов:

```
PICK An,An-1,...,Ao,n --->    An,An-1,...,Ao,An
ROLL An,An-1,...,Ao,n --->    An-1,...,Ao,An
```

Слово PICK (взять) дублирует n-й элемент стека данных (считая от нуля), так что 0 PICK тождественно DUP, а 1 PICK тождественно OVER. Слово ROLL (повернуть) циклический переставляет n верхних элементов стека (тоже считая от нуля) по часовой стрелке (если смотреть на диаграмму), так что 2 ROLL тождественно ROT, 1 ROLL тождественно SWAP, а 0 ROLL является пустой операцией.
Чтобы "увидеть" верхнее значение на стеке данных, используется слово . (точка), которое снимает значение с вершины стека и печатает его в выходном буфере и на терминале как целое число в свободном формате (то есть без ведущих нулей и со знаком минус, если число негативно). Если пользователь хочет, чтобы напечатанное значение осталось на стеке, он должен выполнить следующий текст:

```
> DUP .
```

Слово DUP создаст копию верхнего значения, а точка распечатает его и уберет со стека.
Для работы с вершиной математического стека существуют следующие слова:

```
FDUP      F:A        ---> F:A, A
FDROP     F:A        ---> F: -
FOVER     F:A,B      ---> F:A,B,A
FROT      F:A,B,C    ---> F:B,C,A
FSWAP     F:A,B      ---> F:B,A
```

Слово FDUP (от FLOAT DUPLICATE - дублировать) дублирует вершину математического стека, добавляя в стек еще одно значение, равное тому, которое было до этого верхним. Слово FDROP (сбросить) убирает верхнее значение. Слово FOVER (через) дублирует значение, которое расположено на математическом стеке непосредственно под верхним. Слово FROT (от FLOAT ROTATE - вращать) циклический переставляет по часовой стрелке (если смотреть на диаграмму) три верхних значения в математическом стеке. Наконец, слово FSWAP (обменять) меняет местами два верхних значения.
Также можно работать с любым элементом математического стека с помощью слов:

```
FPICK     F:An,An-1,...,Ao,n --->  F:An,An-1,...,Ao,An
FROLL     F:An,An-1,...,Ao,n --->  F:An-1,...,Ao,An
```

Слово FPICK (взять) дублирует n-й элемент математического стека (считая от нуля), так что 0 FPICK тождественно FDUP, а 1 FPICK тождественно FOVER. Слово FROLL

(повернуть) циклически переставляет n верхних элементов математического стека (также считая от нуля) по часовой стрелке (если смотреть на диаграмму), так что 2 FROLL тождественно FROT, 1 FROLL тождественно FSWAP, а 0 FROLL является пустой операцией.
Чтобы "увидеть" верхнее значение на математическом стеке, используются слова F. (FLOAT - точка) или FE. (FLOAT ENGINEER — точка) которые снимают значение с вершины математического стека и печатают его в выходном буфере и на терминале. В первом случае число печатается с фиксированной запятой в свободном формате с шестью знаками после запятой (то есть без ведущих нулей и со знаком минус, если число негативно). Во втором случае число печатается в инженерном/научном представлении с мантиссой, основой 10 и экспонентой (например -1,234E-02; 1,98E+12). Точность представления чисел при распечатке (то есть количество цифр после десятичной запятой) можно установить с помощью системной переменной FPREC, которая, по умолчанию равна 6. Для записи нового значения в системную переменную FPREC существует слово FPREC!, которое работает следующим образом: с вершины стека данных снимается число (новое значение переменной FPREC в диапазоне от 0 до 6) и присваивается этой переменной. Точность равная 0 означает отсутствие десятичной запятой вообще, что тождественно преобразованию к целому. Значение системной переменной FPREC не сохраняется при выключении питания, поэтому при использовании иной, чем по умолчанию, точности представления чисел при распечатке, эту переменную следует инициализировать.
Как уже упоминалось, стек представляет собой структуру данных, которая функционирует по принципу "последний зашел - первый вышел". Такая структура может пригодится для построения некоторых алгоритмов обработки данных. Для этого необходимо знать количество элементов данных на стеке, чтобы иметь возможность выполнить цикл обработки этих данных определенное количество раз. Для определения количества элементов на стеке данных и математическом стеке, существуют слова DEPTH (глубина) и FDEPTH (FLOAT глубина), которые кладут на вершину стека данных количество элементов на стеке данных и на математическом стеке соответственно:

```
DEPTH     An,An-1,...,A1      ---> An,An-1,...,A1,n

FDEPTH    -                   ---> n
          F:An,An-1,...,A1    ---> F:An,An-1,...,A1
```

Перечисленные выше слова работают со значениями, которые уже находятся в стеке. А как занести значение в стек? Язык ForthLogic™ имеет следующее *правило по умолчанию*: если введенное слово форт-системе не известно, то прежде чем сообщать пользователю об ошибке, форт-система пытается понять это слово как запись числа. Если слово-число состоит из одних цифр с возможным начальным знаком минус, то ошибки нет: слово считается известным и его действие заключается в том, что данное число кладется на вершину стека данных. Если слово-число состоит из одних цифр разделенных десятичной запятой (в действительности это буква . (точка)) с возможным начальным знаком минус и возможным научным представлением экспоненты числа (с помощью букв e или E и возможным знаком экспоненты минус), то слово также считается известным и его действие заключается в том, что данное число кладется на вершину

математического стека.
Для непосредственного переноса чисел из одного стека в другой существуют слова D>F и F>D. Слово D>F снимает верхнее значение с вершины стека данных и переносит на вершину математического стека с соответствующим превращением представления числа. Слово F>D снимает верхнее значение с вершины математического стека и переносит на вершину стека данных с округлением к ближайшему целому числу и соответствующим превращением представления числа. В стековой нотации для данных слов необходимо отображать состояние обеих стеков:

```
D>F       A          ---> -
          F: -       ---> F:преобразованное значение A

F>D       F:A        ---> F: -
          -          ---> округленное значение A
```

Теперь у нас достаточно средств, чтобы привести примеры диалога. Рассмотрим следующий протокол работы (далее по тексту будем выделять протокол работы с терминалом серым цветом):

```
> 5 6 7
(OK)
> SWAP . . .
6 7 5 (OK)
> 123.456 -12.987E-2 FE. F.
1.234560E+02 -0.129870 (OK)
>
```

В ответ на приглашение к вводу (символ ">", напечатанный системой) пользователь вводит три числа: 5, 6 и 7. Обрабатывая введенный текст, форт-система кладет эти числа в указанном порядке на стек данных и после окончания обработки выводит подтверждающее сообщение OK и снова приглашает пользователя к вводу. Далее пользователь вводит текст с четырех слов: SWAP и три точки. Выполняя эти слова-команды, форт-система меняет местами два верхних элемента стека данных (5, 6, 7 -> 5, 7, 6) и потом по очереди три раза снимает верхнее значение со стека данных и печатает его. В результате на терминале появляется текст 6 7 5 и сообщение OK, указывающее на окончание обработки, после чего система снова выдает пользовательское приглашение на ввод. В ответ на приглашение к вводу пользователь вводит два числа с фиксированной и плавающей запятой и слова для отображения двух значений с вершины математического стека в разном формате. Обрабатывая введенный текст, форт-система кладет эти числа в указанном порядке на математический стек а потом снимает их со стека печатая в указанном формате.
В предыдущем примере, при выводе значений со стека на терминал (и в выходной буфер), автоматически печатается один пробел. Такое поведение форт-системы устанавливается по умолчанию, однако, можно произвольным образом активировать или дезактивировать автоматическую печать одного пробела после любого вывода на терминал (и в выходной буфер). Для этого существуют слова AUTOSPACE - активация автоматической печати одного пробела, и NOAUTOSPACE - дезактивация автоматической печати одного пробела.
Чтобы увеличить количество пробелов (для форматирования вывода) можно

применить слово SPACE (пробел). Данное слово печатает в выходном буфере и на терминале один пробел:

```
> NOAUTOSPACE 5 6 7
(OK)
> SWAP . SPACE . SPACE . SPACE
6 7 5 (OK)
> AUTOSPACE 5 6 7
(OK)
> . . .
7 6 5 (OK)
>
```

Чтобы перенести текст на новую строку (а также для форматирования вывода) можно применить слово NEWLINE (новая строка). Данное слово переносит текст в выходном буфере и на терминале на новую строку - однако, практическое применение этого слова имеет смысл при выводе текста на встроенный дисплей и для форматирования текстов SMS (см. следующие разделы).
Слова-команды, которые выполняют целочисленные арифметические операции над числами со стека данных, являются общепринятыми математическими обозначениями:

```
+         A,B  ---> сумма A+B
-         A,B  ---> разница A-B
*         A,B  ---> произведение A*B
/         A,B  ---> частное от деления A/B
MOD       A,B  ---> остаток от деления A/B
ABS       A    ---> абсолютная величина A
NEGATE    A    ---> значение с обратным знаком -A
```

Слова-команды, которые выполняют математические операции над числами в формате с плавающей запятой с математического стека, отличаются от общепринятых добавлением буквы F:

```
F+        F:A,B     ---> F:сумма A+B
F-        F:A,B     ---> F:разница A-B
F*        F:A,B     ---> F:произведение A*B
F/        F:A,B     ---> F:деление A/B
FABS      F:A       ---> F:абсолютная величина A
FNEGATE   F:A       ---> F:значение с обратным знаком -A
```

Слова-команды, которые выполняют вычисление математических функций над числами в формате с плавающей запятой с математического стека:

```
    FSIN      F:A       ---> F:синус угла A в радианах sin(A)
    FCOS      F:A       ---> F:косинус угла A в радианах cos(A)
    FTAN      F:A       ---> F:тангенс угла A в радианах tg(A)
    FSINH     F:A       ---> F:синус гиперболичный A sh(A)
    FCOSH     F:A       ---> F:косинус гиперболичный A ch(A)
    FTANH     F:A       ---> F:тангенс гиперболичный A th(A)
    FASIN     F:A       ---> F:арксинус A arcsin(A), -1.0<=A<=1.0
    FACOS     F:A       ---> F:арккосинус A arccos(A), -1.0<=A<=1.0
    FATAN     F:A       ---> F:арктангенс A arctg(A)
    FLOG      F:A       ---> F:логарифм десятичный A log10(A)
    FLN       F:A       ---> F:логарифм натуральный A ln(A)
    FEXP      F:A       ---> F:экспонента A eA
    F**       F:A,B     ---> F:возведение A в степень B AB
    FSQRT     F:A       ---> F:квадратный корень из A √A
```

Использование стека для хранения промежуточных значений естественным образом приводит к так называемой "обратной польской форме" - одному из способов бесскобочной записи арифметических выражений, состоящему в написании знака операции после операндов. Например, выражение (D*E-F*(G-H)) записывается следующим образом: A B / C + D E * F G H - * - *. Таким образом, форт-систему можно использовать в качестве калькулятора целочисленных значений. Чтобы вычислить, например, значение (25+18+32)*5, достаточно ввести такой текст:

```
> 25 18 + 32 + 5 * .
375 (OK)
>
```

В ответ система напечатает (выполняя точку) желаемый результат. Также, форт-систему можно использовать в качестве калькулятора для чисел с плавающей запятой. Чтобы вычислить значение 1.2e-2*sin(25.23+0.18*3.1415), можно ввести текст:

```
> 1.2e-2 25.23 0.18 3.1415 F* F+ FSIN F* F.
0.007383 (OK)
>
```

Введение новых слов

Базовое свойство языка ForthLogic™ - это возможность вводить новые слова, расширяя тем самым набор команд в нужном для пользователя направлении. Для ввода новых слов чаще всего используется *определение через двоеточие* - определение нового слова через уже известные. Такое определение начинается словом : (двоеточие) и заканчивается словом ; (точка с запятой). Сразу после двоеточия идет слово, которое определяется, а за ним последовательность слов, через которые оно определяется. Например, текст:

```
> : S2 DUP * SWAP DUP * + ;
(OK)
>
```

определяет слово S2, которое вычисляет сумму квадратов двух чисел, которые снимаются с вершины стека. Стековая нотация такого слова имеет вид:

```
    S2        A,B       --->      (A**2+B**2)
```

После ввода в словарь данного описания, слово S2 можно выполнять и включать в описания других слов. При создании таких определений рекомендуется тщательным образом отслеживать все изменения стека.
Перепишем приведенное выше определение слова S2 с комментариями, показывая в скобках состояние вершины стека после выполнения каждой строки. Слово ((левая скобка) предназначено для комментариев - форт-система игнорирует все слова после левой скобки или до конца строки или до первой закрывающей правой скобки, перед которой должен стоять по крайней мере один пробел:

```
: S2          ( A,B ---> A**2+B**2 сумма квадратов )
  DUP         ( A,B,B )
  * SWAP      ( B**2,A )
  DUP *       ( B**2,A**2 )
  + ;         ( A**2+B**2 )
(OK)
>
```

Рассмотрим подробнее работу форт-системы во время определения новых слов. Мы уже знаем, что получив от пользователя дежурную порцию входного текста, форт-система выделяет в ней отдельные слова и ищет их в своем словаре. Эту работу производит встроенный текстовый интерпретатор форт-системы. Если слово в словаре не найдено, то текстовый интерпретатор пытается понять его как число, используя описанное выше правило по умолчанию. Если слово найдено или оказалось записью числа, то дальнейшие действия интерпретатора зависят от его текущего состояния. В каждый момент времени текстовый интерпретатор находится в одном из двух состояний: в состоянии *выполнения* или в состоянии *компиляции*. В состоянии выполнения найденное слово выполняется (то есть производится действие, которое представляет его семантику), а число кладется на стек. Если же интерпретатор находится в состоянии компиляции, то найденное слово не выполняется, а компилируется, то есть включается в создаваемую последовательность действий для слова, которое определяется в данный момент. Найденное и скомпилированное таким образом слово будет выполнено вместе с другими такими словами во время выполнения определенного через них слова. Если нужно скомпилировать число, то текстовый интерпретатор компилирует особый код, который во время выполнения положит значение данного числа на стек (это действие абсолютно прозрачно для пользователя). Скомпилированные слова выполняются чрезвычайно быстро: разница между временем интерпретации произвольной последовательности действий и временем выполнения той же скомпилированной последовательности действий достигает трех порядков - это огромное преимущество языка ForthLogic™ над другими интерпретируемыми языками.
Проследим за работой текстового интерпретатора по обработке уже рассмотренного определения слова S2. Допустим, что перед началом обработки введенной строки интерпретатор находится в состоянии выполнения. Первым словом является :

(двоеточие), которое выполняется. Его семантика заключается в том, что из входной строки выбирается следующее слово и запоминается как слово, которое определяется, а интерпретатор переключается в состояние компиляции. Следующие слова, которые интерпретатор будет выбирать из входной строки, не учитывая комментарии (DUP *, SWAP, и тому подобное), будут компилироваться, а не выполняться, поскольку интерпретатор находится в состоянии компиляции. В результате, со словом S2 связывается последовательность действий, которая отвечает этим словам. Процесс выделения и компиляции слов будет продолжаться до тех пор, пока не встретится ; (точка с запятой). Это слово особенное, оно имеет так называемый признак немедленного выполнения. Слова с таким признаком выполняются независимо от текущего состояния текстового интерпретатора, поэтому точка с запятой будет вторым выполненным словом после двоеточия. Семантика точки с запятой заключается в том, что построение определения, начатого двоеточием, завершается и интерпретатор снова переключается в состояние выполнения. Поэтому после ввода определения слова S2, мы сразу можем проверить, как оно работает на конкретных значениях:

```
> 5 4 S2 .
41 (OK)
```

Слова с признаком немедленного выполнения выполняются по-разному в зависимости от состояния текстового интерпретатора. Некоторые из них даже используются только в одном из этих состояний. Например слово ; допустимо применять только в состоянии компиляции. Оно завершает построение нового определения и переключает текстовый интерпретатор в состояние выполнения.
Введенные слова можно исключить из словаря с помощью слова FORGET (забыть), которое выбирает из входной строки следующее слово (введено после слова FORGET) и исключает его из словаря вместе со всеми словами, определенными в последствии. Такой способ исключения слов выглядит естественным при описании задачи методом "снизу-вверх", который используется в языке ForthLogic™.
Разберем следующий протокол диалога:

```
> 2 2 * .
4 (OK)
> : 2 3 ;
(OK)
> 2 2 * .
9 (OK)
> FORGET 2
(OK)
> 2 2 * .
4 (OK)
>
```

Сначала пользователь вычисляет результат от умножения 2 на 2 и получает ответ 4. Введя после этого определение слова 2 как число 3, он в дальнейшем получает уже другой ответ. Исключив это определение слова 2 через FORGET, он возвращается к бывшей семантике слова 2 - то есть просто цифры 2.
Для работы с новыми словами и словарем, существуют также слова WORDS

(слова) и UNUSED (не использовано). Слово WORDS печатает на терминале список всех доступных слов в словаре в порядке их определения (если список не помещается на экране, то делается остановка пока пользователь не нажмет клавишу "Space" (пробел)). Слово UNUSED кладет на стек свободное место в словаре в байтах. Дальше, например, с помощью слова . (точка), это значение можно отобразить на терминале. Также существует возможность возвращения словаря до первичного состояния, когда в нем присутствуют лишь встроенные слова. Для этого, в терминальном режиме работы текстового интерпретатора форт-системы необходимо с новой строки без предварительных пробелов указать команду BUILD DICTIONARY.
Конкретное содержимое словаря зависит от версии программной прошивки. Узнать о версии можно с помощью слова VERSION (версия) которое печатает в выходной буфер и на терминал строку версии данной программной прошивки.
При определении новых слов через двоеточие, иногда случается так, что не хватает места в пределах одной строки, чтобы закончить построение нового слова. В таких случаях, когда строка превышает установленную длину, допускается переходить на другую строку с помощью клавиши "Enter" и продолжать определение в новой строке. Этот процесс можно повторять вплоть до окончания определения - то есть до слова ; (точка с запятой).
В определении новых слов через двоеточие можно применять как встроенные слова форт-системы так и другие предварительно определенные через двоеточие слова. Глубина вложений определений через двоеточие может достигать 64 уровней и тесно связанная с программно-недоступным *стеком возвратов*. Стек возвратов используется для временного хранения адресов и некоторой служебной информации. Во время выполнения определения любого слова, на вершине стека возвратов будет находиться адрес того слова, с которого данное определение было вызвано (*адрес возвратов*). Адреса возвратов кладутся на стек возвратов и снимаются из него абсолютно прозрачно для пользователя, однако, переполнение и исчерпание стека возвратов проверяется и сообщается как ошибка.
Общая идеология использования собственных слов при написании программы на языке ForthLogic™ заключается в том, что сначала создаются самые простые "базовые" слова которые оперируют наименьшими порциями данных или производят самые простые действия. Их роботу обычно можно немедленно проверить и откорректировать в терминальном режиме путем имитации "рабочего" окружения (подставив на стек или в выходной буфер необходимые данные, подав на вход необходимый сигнал, и т.п.). Убедившись в корректности этих слов, на их основе можно создавать и проверять более сложные слова, на их основе еще более сложные и так далее. Этот процесс в конце сводится к созданию собственного словаря задачи и одного главного слова, которое определяет и запускает на выполнение всю программу. Понятно, что таких программ в словаре может быть несколько и каждая из них обязательно имеет собственное главное слово. Выполнение того или другого главного слова может полностью изменить поведение системы. Данный факт является уникальным в среде программируемых контроллеров. Также важен упомянутый выше способ написания программ на языке ForthLogic™ снизу-вверх, который, к тому же, является единственно возможным. Такие программы содержат наименьшее возможное количество ошибок.

Константы, строки и переменные

Пользователю часто бывает удобно работать не с "анонимными" значениями, а с поименованными. По аналогии со средствами других языков эти средства языка ForthLogic™ называются константами и переменными. Также существуют строки - объекты, которые используются для вывода текста в выходной буфер и на терминал.

Константы, как следует из их названия, являются поименованными неизменными величинами. Физически константы располагаются в словаре среди других слов, а поскольку словарь находится в энергонезависимой памяти то и константы также имеют свойство хранить свое значение при выключении питания. Однако существует возможность менять значение констант, что позволяет реализовать долговременное энергонезависимое хранение любых числовых параметров конкретной задачи. Надо лишь помнить, что количество изменений одной константы ограничено на уровне приблизительно 100000 раз и сам процесс изменения является достаточно долговременным (4-10 мсек). Константы могут быть *обычные* - это те которые были определены пользователем и могут быть переопределены, и *системные* - это константы из базового набора слов, которые упрощают выполнение определенных заданий и не могут быть переопределены.

Рассмотрим слова для работы с константами. Слово CONSTANT (константа) работает следующим образом. Со стека данных снимается верхнее значение, а из входного текста выбирается следующее слово (введено после слова CONSTANT) и запоминается в словаре как новая команда. Ее действие заключается в следующем: положить на стек данных значение, снятое со стека в момент ее определения. Слово TO (в) работает следующим образом. Со стека данных снимается верхнее значение, а из входного текста выбирается следующее слово (введено после слова TO) - название константы целого типа. Дальше осуществляется модификация данной константы в соответствии с введенным значением.

Слово FCONSTANT (FLOAT-константа) работает аналогичным способом - лишь относится к математическому стеку. С математического стека снимается верхнее значение, а из входного текста выбирается следующее слово (введено после слова FCONSTANT) и запоминается в словаре как новая команда. Ее действие заключается в следующем: положить на математический стек значение, снятое со стека в момент ее определения. Слово TOF (в) работает следующим образом. С математического стека данных снимается верхнее значение, а из входного текста выбирается следующее слово (введено после слова TOF) - название константы плавающего типа. Дальше осуществляется модификация данной константы в соответствии с введенным значением. Рассмотрим пример:

```
> 4 CONSTANT ХОРОШО 3.14156 FCONSTANT PI
(OK)
> ХОРОШО . PI F.
4 3.141560 (OK)
> 5 TO ХОРОШО ХОРОШО .
5 (OK)
```

В дальнейшем, при выполнении слова ХОРОШО, на стек данных будет положено число 5, а при выполнении слова PI - на математический стек будет положено число 3,14156.

Переменные это поименованные ячейки оперативной памяти. Физическая природа этой памяти позволяет осуществлять модификацию переменных бесконечное количество раз, процесс модификации чрезвычайно быстрый (100 нсек), однако, оперативная память не есть энергонезависимой.
Пользователю доступно D_MAX глобальных переменных для хранения целых чисел, имена которых - это номера в диапазоне от 1 до D_MAX. Количество глобальных переменных D_MAX зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении. Для работы с глобальными переменными существуют слова VAR? и VAR!. Слово VAR? работает следующим образом. Со стека данных снимается верхнее значение - номер переменной, а на стек данных кладется копия содержимого соответствующей переменной. Слово VAR! работает иным образом. Со стека данных снимается два верхних значения - номер переменной и новое значение, которое запоминается в соответствующей переменной. Приведем пример работы с этими словами:

```
> 4 1 VAR! -237889 16 VAR!
(OK)
> 1 VAR? .
4 (OK)
> 16 VAR? .
-237889 (OK)
```

Пользователю также доступно F_MAX глобальных математических переменных для хранения чисел в формате с плавающей запятой, имена которых - это номера в диапазоне от 1 до F_MAX. Количество глобальных математических переменных F_MAX зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении. Для работы с глобальными математическими переменными существуют слова FVAR? и FVAR!. Слово FVAR? работает следующим образом. Со стека данных снимается верхнее значение - номер математической переменной, а на математический стек кладется копия содержимого соответствующей переменной. Слово FVAR! работает иным образом. Со стека данных снимается верхнее значение - номер переменной, с математического стека снимается верхнее значение и запоминается в соответствующей математической переменной. Приведем пример работы с данными словами:

```
> 4.123 1 FVAR! -2.37889 16 FVAR!
(OK)
> 1 FVAR? F.
4.123000 (OK)
> 16 FVAR? FE.
-0.237889E+01 (OK)
```

Целые и математические переменные имеют достаточно широкий диапазон представления чисел и, соответственно, занимают в памяти достаточно много места. Однако, для задач управления часто необходимо оперировать отдельными битами и флажками. Конечно, для этого можно использовать целые числа и целые переменные, но это неэффективно с точки зрения затрат памяти и времени выполнения. Для работы с отдельными битами и флажками пользователю доступно B_MAX глобальных однобитовых переменных, имена которых - это номера в

диапазоне от 1 до B_MAX. Количество глобальных однобитовых переменных B_MAX зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении. Для работы с однобитовыми переменными существуют слова FLAG? и FLAG!. Слово FLAG? работает следующим образом. Со стека данных снимается верхнее значение - номер переменной, а на стек данных кладется копия содержимого соответствующей переменной, которая принимает значение ИСТИНА или НЕ ИСТИНА (с данными определениями можно ознакомиться в параграфе "Логические операции"). Слово FLAG! работает иным образом. Со стека данных снимается два верхних значения - номер битовой переменной и новое значение, которое интерпретируется как ИСТИНА (для чисел не ровных нулю) или НЕ ИСТИНА (для чисел ровных нулю) и запоминается в соответствующей битовой переменной в виде 1 или 0. Приведем пример работы с данными словами:

```
> 1 1 FLAG! 0 16 FLAG! 124 100 FLAG!
(OK)
> 1 FLAG? . 16 FLAG? . 100 FLAG? .
- 1 0 -1 (OK)
```

Строки текста помогают пользователю при программировании специфических команд, таких как осуществление и прием голосовых звонков, воспроизведения звуковых файлов, вывода на дисплей, и т.п. Эти команды используют форматированное содержимое выходного буфера. При работе в терминальном режиме форт-системы вывод в выходной буфер одновременно дублируется выводом на терминал. Размер выходного буфера составляет OUTBUF_MAX - он зависит от аппаратной платформы, на которой функционирует форт-система и приведен в приложении. При переполнении выходного буфера происходит автоматическое отбрасывание данных которые в него не вмещаются и это не вызывает никаких ошибок.
Для определения строки текста, которая будет печататься в выходном буфере, используется слово ." (точка-кавычки), после которого вводится любой необходимый текст с возможными пробелами. В конце текста необходимо ввести пробел и слово " (кавычки), или можно просто закончить текст клавишей "Enter". Определение строки текста с помощью слов ." (точка-кавычки) и " (кавычки) не позволяет применять в тексте сами кавычки если перед ними должен быть пробел. Если возникает необходимость печатать в выходном буфере текст с кавычками, то для этого можно использовать слово QUOTE (кавычки). Приведем пример:

```
> NOAUTOSPACE ." Hello " SPACE QUOTE ." WORLD " QUOTE
> Hello "WORLD"(OK)
>
```

Такое определение строки текста можно использовать как внутри определения через двоеточие, которое имеет большое практическое значение, так и в диалоговом режиме. Строка текста, определенная внутри определения через двоеточие, имеет признак константной строки, не подлежит дальнейшей модификации и есть энергонезависимой. Например:

```
> ." +375296127630 "
+375296127630 (OK)
> : Hi! ." Hello world! " ;
(OK)
> Hi!
> Hello world! (OK)
> ". Вас приветствует система X
Вас приветствует система X (OK)
```

Для определения того, что строка текста в выходном буфере не является пустой, существует слово LENGTH (длина), которое кладет на стек длину строки текста (включая с пробелы), которая находится в выходном буфере. В отличие от других слов, которые используют содержимое выходного буфера и очищают его при этом, слово LENGTH оставляет выходной буфер без изменений. Для непосредственной очистки выходного буфера применяется слово FLUSH (смывать).
Для оперативного хранения строк символов длиной до 15 букв в оперативной памяти существуют статические строчные переменные. Пользователю доступно S_MAX строчных переменных, имена которых - это номера в диапазоне от 1 до S_MAX. Количество строчных переменных S_MAX зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении. Для работы со строчными переменными существуют слова STRING? и STRING!. Слово STRING? работает следующим образом. Со стека данных снимается верхнее значение - номер переменной и в выходной буфер печатается содержимое данной переменной. Слово STRING! работает обратным образом. Со стека данных снимается верхнее значение - номер переменной, а из выходного буфера считывается строка символов и запоминается в соответствующей переменной. Приведем пример работы с этими словами:

```
> ". +375296127630 "   1 STRING!
+375296127630 (OK)
> 1 STRING?
+375296127630 (OK)
```

Имена переменных в виде целых чисел естественным образом позволяют реализовать индексный доступ к ячейкам памяти разного типа. В частности, организация массивов является естественным развитием такого доступа. Рассмотрим следующий пример: необходимо организовать массив из 10 целых чисел с соответствующим доступом к нему. Массив расположим начиная с номера переменной записанной в константе base :

```
> 15 CONSTANT base
(OK)
> : readarray base + VAR? ; ( n ---> A(n))
(OK)
> : writearray base + VAR! ; ( A(n), n ---> - )
(OK)
> 123 4 writearray
(OK)
> 1 readarray 4 readarray
0 123 (OK)
```

В этом примере мы также использовали косвенную адресацию переменных, когда номер переменной внутри массива определялся путем математических вычислений. Номер переменной любого типа также может храниться в целочисленной переменной и это дает еще одно преимущество индексного доступа к оперативным данным.

Логические операции

В языке ForthLogic™ целое число 0, в двоичном эквиваленте которого все разряды нули, представляет логическое значение НЕ ИСТИНА, а любое другое 32-разрядное число воспринимается как ИСТИНА. Вместе с тем стандартные слова, которые должны возвращать в качестве результата логическое значение, из всех возможных представлений значения ИСТИНА используют только одно: целое число -1, в двоичном эквиваленте которого все разряды единицы. Это связано с тем, что логические операции конъюнкции, дизъюнкции и отрицания выполняются в языке ForthLogic™ *поразрядно* над всеми разрядами операндов и в этом случае соответственно трактуются как поразрядные логические операции. Однако их можно трактовать и как традиционные логические операции, но только над величинами ИСТИНА и НЕ ИСТИНА в упомянутом выше представлении. Для удобства при осуществлении традиционных логических операций существуют специальные системные константы TRUE (истина) и FALSE (не истина), которые кладут на стек логические значения ИСТИНА и НЕ ИСТИНА соответственно:

```
    TRUE      -         --->      -1
    FALSE     -         --->      0
```

Слова-команды, которые выполняют логические вычисления над значениями стека данных, имеют традиционный синтаксис и работают или как обычные логические операции или как поразрядные логические операции:

```
AND       A, B      ---> поразрядное логическое И A•B
OR        A, B      ---> поразрядное логическое ИЛИ A+B
XOR       A, B      ---> поразрядное логическое ИСКЛЮЧИТЕЛЬНОЕ ИЛИ A⊕B
NOT       A         ---> поразрядное логическое ОТРИЦАНИЕ ^A
```

Логические значения также возникают в операциях целочисленного сравнения, которые входят в базовый набор слов и имеют общепринятые обозначения:

```
    <     A, B      ---> A < B    меньше
    =     A, B      ---> A = B    равно
    >     A, B      ---> A > B    больше
    <=    A, B      ---> A <= B   меньше равно
    <>    A, B      ---> A != B   не равно
    >=    A, B      ---> A >= B   больше равно
```

Все эти операции сравнения снимают со стека данных два верхних значения, сравнивают их как числа со знаком и возвращают на стек данных результат сравнения в виде значений ИСТИНА или НЕ ИСТИНА в описанном выше представлении.
Логические значения также возникают в операциях сравнения чисел с

математического стека. Однако, эти операции снимают два верхних значения с математического стека, сравнивают их как числа со знаком и возвращают результат сравнения на стек данных в виде значений ИСТИНА или НЕ ИСТИНА в описанном выше представлении:

```
F<   F: A, B   ---> F : -
     -         ---> A < B   меньше

F>   F: A, B   ---> F : -
     -         ---> A > B   больше
```

Приведем пример применения логических вычислений:

```
> 32 124 < .
-1 (OK)
> 2.4e-2 124.904 F< .
-1 (OK)
> 1.23 56.5678 F> 34 35 < AND .
0 (OK)
> TRUE DUP NOT . NOT NOT .
0 -1 (OK)
```

К логическим операциям над стеком данных также относятся слова логического сдвига влево LSHIFT и вправо RSHIFT. Слово LSHIFT возвращает на стек значение которое получается из A сдвигом его на B разрядов влево и заполнением нулями освобожденных справа разрядов. Аналогично, слово RSHIFT возвращает на стек значение которое получается из A сдвигом его на B разрядов вправо и заполнением нулями освобожденных слева разрядов:

```
LSHIFT    A, B      --->      (A << B)  логический сдвиг влево
RSHIFT    A, B      --->      (A >> B)  логический сдвиг вправо
```

Приведем пример применения операций сдвига:

```
> 1 1 LSHIFT .
2 (OK)
> 2 1 RSHIFT .
1 (OK)
> 1 20 LSHIFT .
1048576 (OK)
> - 1048576 20 RSHIFT .
4095 (OK)
```

Операции сдвига можно использовать для вычисления битовых масок, которые необходимы при установке отдельных битов в поразрядных логических операциях. Приведем пример применения операций сдвига для вычисления битовых масок:

```
> : i>m 1 SWAP LSHIFT ; ( index -- mask )
(OK)
> : SetBit i>m OR ; ( flags_before index -- flags_after )
(OK)
> : ClearBit i>m NOT AND ; ( flags_before index -- flags_after )
(OK)
> : InvertBit i>m XOR ; ( flags_before index -- flags_after )
(OK)
> 10 i>m 20 i>m . .
1048576 1024 (OK)
> 0 10 SetBit .
1024 (OK)
> 1024 10 ClearBit .
0 (OK)
> 0 10 InvertBit InvertBit .
0 (OK)
```

Слово i>m вычисляет битовую маску для соответствующего номера бита. Слова SetBit, ClearBit и InvertBit осуществляют соответствующую манипуляцию над отдельными битами. Например, 1 VAR? 10 SetBit устанавливает 10 бит в первой целочисленной переменной.

Структуры управления

В языке ForthLogic™ есть лишь один тип структур управления процессом выполнения алгоритма - условный оператор. Все другие типы структур управления, такие как многовариантный выбор, условные циклы, циклы со счетчиком и т.п. можно построить с помощью условного оператора и концепции многозадачности на основе таймеров (см. параграф "Таймеры и многозадачность"). Такой подход гарантирует (особенно для разнообразных бесконечных циклов) разделение процессорного времени между отдельными задачами и невозможность его монополизации одной задачей в условиях кооперативной многозадачности, в которой форт-система функционирует как одна из задач.

Условный оператор строится с помощью слов IF, ELSE и THEN. Эти слова используются лишь внутри определений через двоеточие и разделяют тело определения на отрезки. Текст IF <часть-это> ELSE <часть-другое> THEN задает следующую последовательность действий. Слово IF (если) снимает значение с вершины стека и рассматривает его как логическое. Если это ИСТИНА (любое ненулевое значение), то выполняется <часть-это> - то есть слова, которые находятся между IF и ELSE, а если НЕ ИСТИНА (равно нулю), то выполняется <часть-другое> - слова между ELSE и THEN. Сами слова ELSE (иначе) и THEN (тогда) играют роль ограничителей для слова IF и самостоятельного значения не имеют.

Другая форма условного оператора заключается в том, что <часть-другое> вместе со словом ELSE может отсутствовать, и тогда условный оператор имеет сокращенную форму IF <часть-это> THEN. Если логическое значение, которое снимается со стека словом IF, есть ИСТИНОЙ, то выполняются слова, которые образовывают <часть-это>, а если НЕ ИСТИНОЙ, то данный оператор не производит никаких действий. Обратите внимание, что в обоих случаях условие на вершине стека для слова IF вычисляется предыдущими словами.

Приведем, например, определение полезного слова ?DUP, которое дублирует верхнее значение стека, если это не ноль, и оставляет стек в предыдущем состоянии, если на вершине ноль:

```
  > : ?DUP DUP IF DUP THEN ;
  (OK)
```

Спецификация данного слова может иметь такой вид (косая черта разделяет два варианта результата, который это слово может оставить на стеке) :

```
     ?DUP       A          --->       A,A / A
```

Приведем, например, определение другого полезного слова LOGIC, которое превращает верхнее значение стека данных в логическое значение пригодное для дальнейших логических операций, и примеры его применения:

```
> : LOGIC IF TRUE ELSE FALSE THEN ;
(OK)
> 12 NOT . 12 LOGIC NOT . 0 LOGIC NOT .
-13 0 -1 (OK)
> 12 45 < 123 LOGIC AND .
-1 (OK)
```

Спецификация данного слова может иметь такой вид:

```
     LOGIC      A          --->       -1 / 0
```

Рассмотрим пример построения многовариантного выбора типа SWITCH - CASE на основе условного оператора:

```
> : ТЕМПЕРАТУРА ( температура -- )
DUP 18 < IF ". Умеренно " ELSE
DUP 21 < IF ". Нормально " ELSE
DUP 24 < IF ". Уютно " ELSE
DUP 27 < IF ". Жарко " ELSE
DUP 40 < IF ". Горячо " ELSE
            ". Пожар! "
THEN THEN THEN THEN THEN DROP 1 STRING! ;
(OK)
```

В этом примере в зависимости от значения на вершине стека в строчную переменную 1 записывается соответствующее слово. Обратите внимание на одинаковое количество слов IF, ELSE и THEN.
Преимущество применения условного оператора для построения многовариантного выбора заключается в том, что сравнение и выбор не ограничено лишь целыми числами - можно сравнивать математические и логические величины, можно даже сравнивать разные типы данных в пределах одной такой составной конструкции.

Таймеры и многозадачность

Как уже вспоминалось раньше, язык ForthLogic™ позволяет описывать параллельно выполняемые процессы, а сама форт-система функционирует в многозадачной

среде. Такое ее свойство тесно связано с понятием таймер. Таймер - это программный объект, который позволяет реализовать часовой интервал, после окончания которого может быть выполнено любое известное форт-системе слово.
Пользователю доступно T_MAX независимых таймеров, имена которых - это номера в диапазоне от 1 до T_MAX. Количество таймеров T_MAX зависит от аппаратной платформы на которой функционирует форт-система и приведено в приложении. Для конфигурации таймеров предназначено слово TIMER!. Слово TIMER! работает следующим образом: с математического стека снимается верхнее значение - временная задержка в секундах, со стека данных снимается верхнее значение - номер таймера, из входного текста выбирается следующее слово (введенное после слова TIMER!) и запускается таймер с соответствующим номером. Смысл выполнения слова TIMER! заключается в следующем: после окончания интервала времени, который равняется значению временной задержки, выполнить слово, которое было указано во время запуска таймера (введено после слова TIMER!).
Дискретность установки времени задержки срабатывания таймеров составляет 0,01 сек. Внутренняя разрядная сетка представления времени позволяет реализовать временные интервалы от 0 сек. До 21474836,47 сек. с шагом 0,01 сек. То-есть, совершенно реально запустить таймер на 248 дней! Особенный случай времени задержки срабатывания - это 0 сек (записывается как 0.0). Задержка 0 сек. означает выполнение назначенного данному таймеру слова в следующем такте работы всей системы - фактическое немедленное выполнение. Дело в том, что форт-система функционирует в условиях кооперативной многозадачности в составе программно-аппаратной системы контроллера, в которой существуют другие программные модули и подсистемы. Все они (равно как и форт-система) выполняются по очереди в отведенных для них тактах. Таким образом, когда выполняется любое слово на языке ForthLogic™, то оно выполняется в пределах такта работы форт-системы и упомянутый выше случай с таймером, который имеет задержку 0 сек. означает, что в следующем такте всей системы будет опять запущена форт-система которая выполнит назначенное таймеру слово. Любая другая временная задержка для таймера означает постановку форт-системы в очередь отложенных заданий и выполнения лишь через определенный промежуток времени, который отвечает данной временной задержке.
Приведем пример использования таймера:

```
> : устан_вых 1 1 RO! ;
(OK)
> 4.0 1 TIMER! устан_вых
(OK)
```

В этом примере, мы определили новое слово устан_вых, которое записывает логическую единицу в релейный выход S1 (переключает контакты реле - об этом немного позже). Дальше мы запустили таймер 1 на время 4 секунды, после окончания которого будет выполненное слово устан_вых (обратите внимание на десятичную точку в представлении времени задержки). Следовательно, контакты реле переключатся через 4 секунды, начиная с момента времени успешной обработки последней строки текстовым интерпретатором форт-системы (после выведения сообщения (OK)).
Слово TIMER! можно применять и внутри определений через двоеточие, тогда появляется возможность создавать последовательность действий, которые

выполняются через определенные промежутки времени. Одна из форм такой последовательности - это создание действий, которые циклически выполняются бесконечное количество раз - то есть, по существу, это механизм создания параллельно выполняемых процессов. Приведем пример:

```
> : сирена 1 RO? NOT 1 RO! 2.0 1 TIMER! сирена ;
(OK)
> сирена
(OK)
```

В этом примере, мы определили новое слово "сирена" со следующими смыслом: на стек кладется логическое состояние релейного выхода S1, осуществляется логическая инверсия вершины стека, после чего значение со стека записывается в релейный выход S1, дальше запускается таймер 1 на время 2 секунды, после окончания которого будет выполнено слово сирена - то есть весь процесс выполнения слова сирена опять повторится. После определения слова, мы выполнили слово "сирена", таким образом, запустив бесконечное количество раз только что описанный процесс и получив поочередное переключение контактов реле S1.

Как уже упоминалось, реализованные в системе таймеры являются независимыми, следовательно, есть возможность запустить T_MAX параллельных заданий, каждое из которых описывается своим словом. В таких многозадачных системах необходимо каким-то образом администрировать отдельные задачи. Поскольку, механизм многозадачности реализован с помощью таймеров, то управляя таймерами можно управлять задачами. Для этого предназначено слово TIMER? и слово NAME (имя).

Слово TIMER? работает следующим образом: со стека снимается верхнее значение - номер таймера, а на стек кладется или адрес слова, которое будет выполнено соответствующим таймером после заданного промежутка времени, или 0, если слово не задано.

Адрес слова - это адрес слова в словаре и для его интерпретации предназначено слово NAME, которое работает следующим образом: со стека снимается верхнее значение и интерпретируется как адрес, дальше осуществляется поиск слова, в определении которого содержится данный адрес (каждое определенное слово занимает в словаре последовательность адресов) и, если слово найдено, то оно печатается в выходном буфере и на терминале, иначе не печатается ничего. С учетом предыдущего примера, приведем протокол следующего диалога :

```
> 1 TIMER? DUP . NAME
1534 сирена (OK)
```

Поскольку в конце выполнения слова "сирена", таймер 1 запускается опять, то в любой момент времени, во время выполнения слова TIMER?, на стек будет положен адрес слова "сирена" (показанное значение адреса приведено для примера). Обратите внимание, с помощью слов DUP . NAME мы одновременно вывели на терминал адрес и название слова.

В процессе работы таймера (когда длится обратной отсчет времени) его можно перезапустить на выполнение другого слова через другой промежуток времени. При этом предыдущее слово и предыдущий промежуток времени аннулируется. В

частности, если перезапустить таймер со специальным словом STOP (остановить), которое не выполняет никаких действий, можно фактически остановить задачу, которая циклический запускается данным таймером. С учетом предыдущих примеров, приведем протокол следующего диалога:

```
  > 0.0 1 TIMER! STOP
  (OK)
```

Переопределив слово таймера 1 со слова "сирена" на слово STOP, мы остановили бесконечное действие слова "сирена". Задержка 0 секунд, которую мы указали в примере, в данном случае означает немедленное выполнение слова STOP (в описанном выше понимании), однако, может быть произвольной, ведь предыдущее слово и предыдущий промежуток времени таймера 1 уже аннулированные.
Как уже упоминалось, для построения некоторых типов структур управления можно использовать условный оператор и таймеры. В частности, это касается разнообразных условных циклов и циклов со счетчиком. Рассмотрим пример построения на основе условного оператора условного цикла типа BEGIN – UNTIL, который существует в языке Форт, однако отсутствует в языке ForthLogic™. Формально, синтаксис построения такого цикла может выглядеть так:

BEGIN <тело цикла> UNTIL

Такой цикл называется условным циклом с проверкой в конце. После выполнения слов, которые образуют его тело, на стеке остается логическое значение - условие завершения цикла. Слово UNTIL (пока не) снимает это значение и анализирует его. Если это ИСТИНА, то выполнение цикла прекращается. Слово UNTIL можно заменить условным оператором и таймером, который с приемлемой для данной задачи периодичностью будет запускать на выполнения слова, которые образуют тело цикла:

```
> : <тело_цикла> 1 RO? NOT 1 RO! 1 DI? ; ( ---> 1DI )
(OK)
> : BEGIN <тело_цикла> NOT IF 0.1 1 TIMER! BEGIN THEN ;
(OK)
> BEGIN
(OK)
```

В этом примере условного цикла слово <тело_цикла> оставляет на стеке логическое значение, которое анализируется в слове BEGIN, и если это НЕ ИСТИНА, то слово BEGIN запускается на выполнение через 0,1сек, иначе слово BEGIN не запускается и действие условного цикла "прекращается". Таким образом, слово UNTIL было заменено рядом слов: "NOT IF 0.1 1 TIMER! BEGIN THEN".
Рассмотрим пример построения на основе условного оператора другого условного цикла - цикла со счетчиком типа DO – LOOP, который также существует в языке Форт, однако отсутствует в языке ForthLogic™. Формально, синтаксис построения такого цикла может выглядеть так:

DO <тело цикла> LOOP

Цикл начинается со слова DO (делать), которое снимает со стека два значения:

начальное значение (на вершине стека) и конечное значение и запоминает их. Текущее значение счетчика устанавливается равным начальному значению и выполняется тело цикла. Слово LOOP увеличивает значение счетчика на 1 и проверяет условие завершения цикла. Условием завершения цикла является превышение конечного значения на 1. Слово LOOP можно заменить операторами увеличения и проверки счетчика и таймером, который с приемлемой для данной задачи периодичностью будет запускать на выполнения слова, которые образуют цикл:

```
> : <тело_цикла> ; ( ---> )
(OK)
> : DO <тело_цикла> 1 VAR? 1 + DUP 1 VAR! 2 VAR? <=
IF 0.1 1 TIMER! DO THEN ;
(OK)
> 1 1 VAR! 10 2 VAR! DO
(OK)
```

Слово LOOP в этом примере было заменено рядом слов: "1 VAR? 1 + DUP 1 VAR! 2 VAR? <= IF 0.1 1 TIMER! DO THEN". Переменная 1 используется в качестве счетчика - она доступна также в теле цикла, а переменная 2 хранит максимальное значение счетчика. Во время работы счетчик увеличивается на 1 и проверяется условие завершения цикла. Перед вызовом слова DO в переменные 1 и 2 необходимо записать минимальное и максимальное значение счетчика.

Векторное выполнение

При определении новых слов, в языке ForthLogic™ разрешается применять лишь известные форт-системе слова. Однако, существует механизм *обратной ссылки*, который фактически позволяет реализовать определение слов после того, как они были предварительно применены в определениях других слов. Этот механизм тесно связан с понятием *векторного выполнения* и основывается на словах EXECUTE и FIND. Слово EXECUTE (выполнить) работает следующим образом: со стека снимается верхнее значение - *исполнительный адрес* любого слова в словаре и данное слово немедленно выполняется так, если бы его выполнили непосредственно введя в форт-систему.

Внимание: следует быть чрезвычайно осторожными с исполнительными адресами - если число на вершине стека во время выполнения слова EXECUTE не является таким адресом, то последствия выполнения слова EXECUTE будут непредсказуемыми!

Благодаря слову EXECUTE появляется возможность передавать через стек исполнительные адреса однотипных слов и таким образом реализовывать векторное (то есть, зависимое от контекста) выполнение программы.

Узнать исполнительный адрес любого слова в словаре можно с помощью слова FIND (найти). Слово FIND работает следующим образом: из выходного буфера считывается строка до первого пробела – слово, для которого необходимо найти исполнительный адрес и, если слово известно форт-системе, на стек кладется исполнительный адрес этого слова. В обратном случае на стек кладется значение НЕ ИСТИНА (или 0).

Приведем пример применения упомянутых слов для реализации обратной ссылки:

```
> 0 CONSTANT 1beep_adr
(OK)
> : (1beep) 1beep_adr EXECUTE ;
(OK)
> : 2beep 1.0 1000 BEEP 1.0 1 TIMER! (1beep) ;
(OK)
> : 1beep 1.0 2000 BEEP 1.0 1 TIMER! 2beep ;
(OK)
> ". 1beep " FIND TO 1beep_adr
(OK)
> 1beep
(OK)
```

Сначала мы создали константу 1beep_adr для хранения исполнительного адреса (в равной степени мы могли бы применить для этого целочисленную переменную). Дальше мы создали слово-транслятор (1beep) которое просто выполняет адрес 1beep_adr. Это слово уже можно использовать там, где планировалось использовать еще не определенное слово 1beep - как, например, в слове 2beep. И наконец, мы определили само слово 1beep. Потом мы запомнили исполнительный адрес слова 1beep в константе 1beep_adr, тем самым "завершая" построение бесконечной последовательности выполнения слов 1beep-2beep-1beep -..

Программирование аппаратных средств

Для работы с аппаратными средствами платформы, на которой реализована форт-система, в языке ForthLogic™ существует ряд базовых слов, которые можно разделить по соответствующим категориям: входы, выходы, система, голосовые звонки, SMS, и тому подобное. Список слов для работы с аппаратными средствами будет постоянно расширяться, отображая возможности конкретной аппаратуры и потребности конечных пользователей.

Входы

Входные сигналы аналоговых и цифровых входов перед тем как попасть в систему фильтруются. Аналоговые входы фильтруются цифровым фильтром низких частот с временем наращивания 690 мс, что отвечает сигнальной частоте среза 0,24 Гц. Цифровые входы фильтруются методом накопления активного/пассивного состояния сигнала с дальнейшим сравнением с пороговыми значениями. Параметры данного алгоритма фильтрации позволяют эффективно поглощать переходные процессы длительностью до 40 мс.
Обращение к входам осуществляется по их номерам. Максимальное значение соответствующего номера зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении.
Для работы с цифровыми входами существует слово DI?, которое снимает со стека верхнее значение - номер входа в диапазоне от 1 до DI_MAX и кладет на стек логическое состояние соответствующего цифрового входа (DI/AIx, DIx, Dx). Логическое состояние ИСТИНА означает замыкание соответствующего входа на любой из контактов GND (общий). В случае комбинированных входов DI/AIx, если соответствующий вход с помощью внутренней перемычки сконфигурирован как аналоговый, то логическое состояние этого входа всегда будет НЕ ИСТИНА.

Для работы с комбинированными входами DI/AIx, которые перемычками сконфигурированы как аналоговые, существует слово AI?, которое снимает со стека верхнее значение - номер входа в диапазоне от 1 до AI_MAX и кладет на математический стек состояние соответствующего комбинированного входа. При этом, если в системе вход определен как цифровой, то значение отвечает абсолютному состоянию 10-разрядного АЦП, если вход определен как вход по току, то значение отвечает току в миллиамперах, и наконец, если вход определен как вход по напряжению, то значение отвечает напряжению в вольтах. Приведем пример применения упомянутых слов:

```
> 1 DI? .
0 (OK)
> 5 DI? .
-1 (OK)
> 1 AI? F. 2 AI? F.
456.000000 3.234136 (OK)
```

Для удобства работы с комбинированными входами DI/AIx в полной версии контроллера, в которой реализовано конфигурационное меню и фиксированный алгоритм работы, существует слово AIS?, которое снимает со стека верхнее значение - номер входа в диапазоне от 1 до AI_MAX и кладет на математический стек состояние соответствующего комбинированного входа. При этом, если в конфигурационном меню вход определен как цифровой, то значение отвечает абсолютному состоянию 10-разрядного АЦП, если вход определен как вход по току или напряжению, то значение отвечает масштабируемой физической величине. Масштабные коэффициенты принимаются из значений указанных в соответствующих пунктах конфигурационного меню. Таким образом, слово AIS? можно использовать для получения реальных показаний физических величин (например, температуры) от датчиков с унифицированным аналоговым выходным сигналом.

Выходы

Обращение к выходам также осуществляется по их номерам. Максимальное значение соответствующего номера зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении.
Для работы с цифровыми выходами типа "открытый коллектор" существует слово DO?, которое снимает со стека верхнее значение - номер выхода в диапазоне от 1 до DO_MAX и кладет на стек логическое состояние соответствующего цифрового выхода (DOx, Dx). Логическое состояние ИСТИНА означает, что транзистор соответствующего выхода находится в проводящем состоянии.
Чтобы установить цифровой выход в нужное состояние, существует слово DO!, которое снимает со стека два верхних значения - номер выхода в диапазоне от 1 до DO_MAX и состояние выхода. Дальше, рассматривая это состояние как логическое значение, устанавливается соответствующим образом логическое состояние соответствующего цифрового выхода (DOx, Dx). Логическое состояние ИСТИНА означает, что транзистор соответствующего выхода будет установлен в проводящее состояние.
Для работы с релейными выходами существует слово RO?, которое снимает со

стека верхнее значение - номер выхода в диапазоне от 1 до RO_MAX и кладет на стек логическое состояние соответствующего релейного выхода Sx. Логическое состояние ИСТИНА означает, что центральный контакт соответствующего релейного выхода замкнут на нормально разомкнутый контакт данного выхода.
Чтобы установить релейный выход в нужное состояние, существует слово RO!, которое снимает со стека два верхних значения - номер выхода в диапазоне от 1 до RO_MAX и состояние выхода. Дальше, рассматривая это состояние как логическое значение, устанавливается соответствующим образом логическое состояние соответствующего релейного выхода Sx. Логическое состояние ИСТИНА означает, что центральный контакт соответствующего релейного выхода будет замкнут на нормально разомкнутый контакт данного выхода.

```
> 1 DO? . 1 1 DO! 1 DO? .
0 -1 (OK)
> 3 RO? .
-1 (OK)
> 3 RO? NOT 3 RO!
(OK)
> 3 RO? .
0 (OK)
```

Последовательный порт RS485

Обмен по последовательному порту RS485 происходит согласно коммуникационному протоколу MODBUS RTU. Устройства подсоединяются к порту RS485 параллельно и образуют сеть. Обмен в сети MODBUS RTU осуществляется между главным устройством (MASTER) и подчиненными устройствами (SLAVE). При этом процесс обмена может инициировать лишь главное устройство типа MASTER, а подчиненные устройства типа SLAVE могут лишь отвечать на запросы. Контроллеры серии MAX Logic могут работать лишь в режиме главного устройства.
По умолчанию, данные между главным и подчиненными устройствами пересылаются в виде байтов, упакованных в 11-битовые посылки на скорости 9600 битов/сек. Посылка начинается со стартового бита (со значением 0), дальше посылается байт с данными (8 битов), а на конце посылается два стоп-бита (со значением 1). Отдельные посылки складываются в пакеты сообщений с определенной структурой.
Устройство типа MASTER начинает все пакеты данных с адреса устройства типа SLAVE, к которому адресуется пакет. Каждое устройство типа SLAVE должно иметь уникальный адрес из диапазона от 1 до 247. В случае, когда устройство типа SLAVE высылает ответ на запрос, то в поле адреса он размещает свой собственный сетевой адрес. Это дает возможность устройству типа MASTER определить, откуда поступил ответ.
Следующий байт пакета данных, высланного устройством типа MASTER, образует код поручения (функции), которое должно быть выполнено устройством типа SLAVE. В ответ SLAVE высылает пакет с таким же кодом поручения. Контроллеры серии MAX Logic могут формировать и обслуживать коды следующих поручений:

- 01 (0x01) Read Coils (Чтение статуса дискретных выходов)
- 02 (0x02) Read Discrete Inputs (Чтение состояния дискретных входов)
- 03 (0x03) Read Holding Registers (Чтение статуса регистров)

- 04 (0x04) Read Input Registers (Чтение состояния входных регистров)
- 05 (0x05) Write Single Coil (Запись отдельного дискретного выхода)
- 06 (0x06) Write Single Register (Запись отдельного регистра)
- 15 (0x0F) Write Multiple Coils (Запись нескольких дискретных выходов)
- 16 (0x10) Write Multiple registers (Запись нескольких регистров)

В дальнейших байтах пакета высылаются или принимаются данные. Количество данных, которые пересылаются, зависит от кода поручения.

После отсылки/получения всех данных, к пакету присоединяется два байта с контрольной суммой CRC. Назначением контрольной суммы CRC является исключение разнообразных ошибок, которые могут возникнуть во время пересылки данных, например, как следствие влияния сильных электромагнитных помех. Контрольную сумму всегда подсчитывает устройство, которое высылает пакет данных. Дальше устройство, которое принимает пакет, еще раз подсчитывает CRC из полученных данных и сравнивает ее с принятым значением. Если обе суммы совпадают, то устройство приступает к обработке поручения, которое содержится в пакете, если нет - пакет игнорируется.

Пакеты сообщений с перечисленными выше кодами поручений формируются и обрабатываются контроллерами серии MAX Logic автоматически в соответствии с кодом поручения. При этом, данные которые высылаются или принимаются, располагаются в глобальных целочисленных или глобальных однобитовых переменных контроллера, что так же зависит от кода поручения. Пакеты сообщений могут формироваться и высылаться в двух режимах: циклическом с установленным периодом и одиночном. В циклическом режиме параллельно могут формироваться до 4 разных пакетов сообщений пронумерованных от 1 до 4 (не путать этот номер с номерами функций, адресами подчиненных устройств и тому подобное - это внутренний для контроллера логический номер структуры данных, которая описывает все аспекты обмена), причем это могут быть сообщения как для разных устройств, так и для одного, но с разными кодами поручений. Такое количество пакетов сообщений, которые могут формироваться параллельно, никоим образом не ограничивает количество устройств, которые могут быть адресованы в сети. Применив одиночный режим формирования пакетов сообщений можно каждый раз произвольно выбирать адрес подчиненного устройства при формировании пакета с тем же номером. Для такого режима формирования пакетов очень удобно применять автоматические вызовы слов, которые происходят в конце одиночного обмена, - так называемые callbacks (обратный вызов).

Для формирования и активации пакетов сообщений существует слово MODBUSSTART (начать обмен согласно протокола MODBUS). Слово MODBUSSTART употребляется вместе с числовыми параметрами и системными константами CYCLIC_ACCESS (циклический режим обмена), SINGLE_ACCESS (одиночный режим обмена), READ_COILS (чтение статуса дискретных выходов - функция 01), READ_INPUTS (чтение состояния дискретных входов - функция 02), READ_HOLDREGS (чтение статуса регистров - функция 03), READ_INPUTREGS (чтение состояния входных регистров - функция 04), WRITE_COIL (запись отдельного дискретного выхода - функция 05), WRITE_REG (запись отдельного регистра - функция 06), WRITE_COILS (запись нескольких дискретных выходов — функция 15) и WRITE_REGS (запись нескольких регистров — функция 16) и предназначено для формирования и активации пакетов сообщений согласно коммуникационного протокола MODBUS RTU. Поскольку системные константы

просто кладут на стек определенные числа, то формально синтаксис применения слова MODBUSSTART можно представить в виде диаграммы стековой нотации однако, учитывая большое количество параметров, более наглядно это можно сделать показав последовательность числовых параметров и системных констант во время вызова данного слова (порядок слов имеет значение и не следует забывать о пробелах!):

<period> CYCLIC_ACCESS <slave> <addr> <amount> <index> <func> <packet> MODBUSSTART

- формируется и активируется пакет сообщений в цикличном режиме;

SINGLE_ACCESS <slave> <addr> <amount> <index> <func> <packet> MODBUSSTART

- формируется и активируется пакет сообщений в одиночном режиме;

Числовые параметры в угловых скобках являются обязательными:

<period> *- период формирования пакета в секундах, может принимать значение от 0,1 сек. до 600,0 сек. с шагом 0,01 сек;*

<slave> *- сетевой адрес подчиненного устройства, должен быть уникальным в пределах целой сети и может принимать значение от 1 до 247;*

<addr> *- адрес данных подчиненного устройства, данный адрес в зависимости от кода поручения может быть как адресом дискретного типа данных так и адресом регистрового типа данных и может принимать значение от 0 до 65535;*

<amount> *- количество данных подчиненного устройства, количество в зависимости от кода поручения может быть как количеством данных дискретного типа так и количеством данных регистрового типа и может принимать значение от 1 до 64;*

<index> *- номер глобальной целочисленной или однобитовой переменной, с которой располагаются данные для записи или результаты считывания, количество задействованных переменных равняется параметру <amount>;*

<func> *- код поручения, может принимать значение READ_COILS, READ_INPUTS, READ_HOLDREGS, READ_INPUTREGS, WRITE_COIL, WRITE_REG, WRITE_COILS, WRITE_REGS (см. коды поручений);*

<packet> *- номер пакета сообщения, может принимать значение от 1 до 4.*

Конечно, вместо конкретных значений числовых параметров могут быть использованные значения переменных или результаты математических операций, надо лишь правильно расположить их на стеке. В стековой нотации синтаксис слова MODBUSSTART имеет две формы в зависимости от параметра режима доступа, который представлен системными константами CYCLIC_ACCESS и SINGLE_ACCESS (цифры 1 и 2 соответственно):

```
MODBUSSTART 1, slave, addr, amount, index, func, packet  ---> -
                                    F: period      ---> F :-

MODBUSSTART 2, slave, addr, amount, index, func, packet  ---> -
                                    F :-           ---> F :-
```

Протоколом MODBUS определены два типа данных:

- дискретный тип - однобитовая величина, которая может принимать значение 0 или 1 и в поручениях используется для описания дискретных входов и выходов (см. коды поручений);
- регистровый тип - 16-битовая величина, которая может принимать значение от 0 до 65535 и в поручениях используется для описания входных и внутренних (holding) регистров.

В тех поручениях, в которых указываются регистровые данные, их содержимое отвечает содержимому целочисленных глобальных переменных контроллера - это касается поручений как записи, так и чтения. То есть, для поручения записи регистров, значения для записи копируются из содержимого соответствующих целочисленных глобальных переменных, а для поручения чтения регистров считанные значения записываются в соответствующие целочисленные глобальные переменные. Соответствие между регистрами подчиненного устройства и целочисленными глобальными переменными контроллера устанавливается с помощью параметров <addr>, <amount> и <index>. То же касается и дискретного типа данных, только ему в соответствие относятся однобитовые глобальные переменные контроллера.

Для настройки автоматического вызова слов в одиночном режиме обмена существует слово MODBUSCALLBACK (обратной вызов протокола MODBUS). Слово MODBUSCALLBACK работает следующим образом: со стека данных снимается верхнее значение - номер пакета сообщения, из входного текста выбирается следующее слово (введено после слова MODBUSCALLBACK) и происходит настройка автоматического обратного вызова этого слова в конце одиночного обмена. Выполнение данного слова является "одноразовым", то есть в конце следующего одиночного обмена оно не будет выполнено, если не будет новой настройки с помощью слова MODBUSCALLBACK. Покажем способ вызова данного слова:

<packet> MODBUSCALLBACK <word>

- настройка автоматического одноразового обратного вызова в одиночном режиме;

Параметры в угловых скобках являются обязательными:

<packet> *- номер пакета сообщения, может принимать значение от 1 до 4.*

<word> *- известно форт-системе слово, которое будет выполнено в конце одиночного обмена.*

Для прекращения формирования пакетов сообщений в циклическом режиме существует слово MODBUSSTOP (остановить обмен по протоколу MODBUS). Слово MODBUSSTOP употребляется вместе с числовым параметром номер пакета и предназначено для остановки непрерывного процесса формирования этого пакета сообщения в циклическом режиме. Покажем способ вызова данного слова:

<packet> MODBUSSTOP

- прекращается формирование пакета сообщения в циклическом режиме;

Числовой параметр в угловых скобках является обязательным:

<packet> *- номер пакета сообщения, может принимать значение от 1 до 4.*

В стековой нотации синтаксис слова MODBUSSTOP имеет очень простую форму:

```
MODBUSSTOP              packet ---> -
```

Для определения состояния процесса обмена существует слово MODBUSSTATUS? (состояние обмена по протоколу MODBUS). Слово MODBUSSTATUS? снимает со стека номер пакета, который может принимать значение от 1 до 4, и возвращает на стек код состояния обмена для данного пакета сообщений. В стековой нотации синтаксис слова MODBUSSTATUS? имеет форму:

```
MODBUSSTATUS?           packet ---> code
```

Коды состояния обмена разделяются на коды ошибок главного устройства, которые возникают в главном устройстве (то есть контроллере) и коды ошибок подчиненного устройства, которые приходят в ответе на пакет сообщения. Данные коды и их описание представлены в двух следующих таблицах:

Код	Описание кодов ошибок главного устройства
0	Нет никаких ошибок обмена с подчиненным устройством
16	Подчиненное устройство не ответило на поручение в течение заданного интервала времени
17	В ответе подчиненного устройства содержится ошибка контрольной суммы CRC

Код	Название ошибки протокола MODBUS	Описание кодов ошибок подчиненного устройства
1	ILLEGAL FUNCTION	Принятый код поручения не может быть обработан подчиненным устройством.
2	ILLEGAL DATA ADDRESS	Адрес данных указный в пакете сообщения является не допустимым для данного подчиненного устройства.
3	ILLEGAL DATA VALUE	Величина которая содержится в поле данных пакета сообщения является не допустимой величиной для данного подчиненного устройства.
4	SLAVE DEVICE FAILURE	Необратимая ошибка имела место когда подчиненное устройство пыталось выполнить поручение.
5	ACKNOWLEDGE	Подчиненное устройство приняло поручение и обрабатывает его, но это требует много времени.
6	SLAVE DEVICE BUSY	Подчиненное устройство занято обработкой поручения. Главное устройство должно повторить сообщение позже, когда подчиненное устройство освободится.

Если обмен для данного пакета происходит в циклическом режиме, то код состояния свидетельствует о возможных ошибках обмена главного устройства или, в случае кодов ошибок подчиненного устройства, о том что обмен прекратился вообще. Если обмен для данного пакета происходит в одиночном режиме, то код состояния свидетельствует о возможных ошибках после того, как обмен завершился (при условии, что запрос состояния осуществлен уже после обмена).
Код ошибки 16 возникает, когда подчиненное устройство не ответило на поручение в течение заданного интервала времени TIMEOUT. Данный интервал времени по умолчанию равняется 1,0 сек. Однако с помощью слова MODBUSTIMEOUT!

(установить время TIMEOUT протокола MODBUS) это время можно изменить. Слово MODBUSTIMEOUT! работает следующим образом: с вершины математического стека снимается число - значение времени в диапазоне от 1,0 сек. До 600,0 сек. с шагом 0,01 сек., и время TIMEOUT устанавливается равным данному значению. В стековой нотации синтаксис слова MODBUSTIMEOUT! имеет следующую форму:

```
MODBUSTIMEOUT!    F: timeout ---> F : -
```

В некоторых случаях существует необходимость изменить параметры обмена по последовательному каналу принятые по умолчанию на другие (иногда даже нестандартные). Для этого существует слово MODBUSPARAM (установить параметры последовательного порта) которое употребляется вместе с числовым параметром и системными константами NONE (отсутствует), EVEN (парный), ODD (непарный) и предназначено для изменения параметров обмена. Покажем способ вызова данного слова:

<parity> <stopbits> <baudrate> MODBUSPARAM

- устанавливает параметры обмена;

Числовые параметры в угловых скобках являются обязательными:

<baudrate>	*- устанавливает скорость обмена, может принимать значение из ряда 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200*
<parity>	*- значение бита четности, может принимать значение NONE (отсутствует), EVEN (парный) или ODD (непарный).*
<stopbits>	*- количество стоповых битов, может принимать значение 1 или 2. При этом 1 стоп бит имеет смысл только при четности NONE. При других типах четности данный параметр значения не имеет.*

В стековой нотации синтаксис слова MODBUSPARAM имеет следующую форму:

```
MODBUSPARAM       parity, stopbits, baudrate ---> -
```

При работе с поручениями, в которых указывается дискретный тип данных, никакой проблемы с интерпретацией данных не возникает: 0 дискретного типа отвечает значению НЕ ИСТИНА однобитовой глобальной переменной, а 1 дискретного типа отвечает значению ИСТИНА однобитовой глобальной переменной. При интерпретации значений регистрового типа существует определенная особенность, которая связана с тем, что регистровый тип является беззнаковым. Это значит, что при работе с поручениями, в которых указывается чтение регистрового типа данных, который подчиненное устройство трактует как знаковое, необходимо превращать полученные беззнаковые числа в знаковые для разрядной сетки целых чисел контроллера. Однако при работе с поручениями, в которых указывается запись регистрового типа данных, необходимость в превращении знаковых чисел в беззнаковые отпадает из-за того, что разрядная сетка целых чисел контроллера перекрывает как знаковые, так и беззнаковые числа регистрового типа данных.
Превращение знаковых чисел регистрового типа данных (представленных как беззнаковые) в знаковые числа целого типа (принятого в контроллере) можно

осуществить с помощью слова US>S (беззнаковый формат в знаковый). Синтаксис данного слова лучше всего объясняет стековая нотация:

```
US>S    беззнаковое_число ---> число_с_знаком
```

Приведем пример:

```
> 65535 US>S .
-1 (OK)
```

Число 65535 попадает в разрядную сетку регистрового типа данных, но в зависимости от интерпретации может значить или 65535 для беззнаковых чисел, или -1 для знаковых.
Особенность интерпретации также возникает при обмене иными, чем регистровый, типами данных – например, при обмене числами в формате с плавающей запятой. Превращение чисел регистрового типа данных в числа математического типа (принятого в контроллере) можно осуществить с помощью слова US>F (беззнаковый формат в математический). Обратное превращение можно осуществить с помощью слова F>US (математический формат в беззнаковый)
Синтаксис данных слов лучше всего объясняет стековая нотация:

```
US>F    беззнаковое_число1,беззнаковое_число2   --->      -
        F:                                  -    --->      F: число

F>US              -    --->      беззнаковое_число1,беззнаковое_число2
        F:    число    --->      F : -
```

Приведем пример:

```
> 1.2345 F>US . .
1048 16286 (OK)
> 16286 1048 US>F F.
1.234500 (OK)
```

Число 1,2345 в двоичном представлении, согласно стандарта IEEE-754, имеет вид 0x3F9E0418, в десятичном эквиваленте регистрового типа данных это отвечает числам 16286 (0x3F9E) и 1048 (0x0418).
Теперь у нас достаточно средств, чтобы запрограммировать работу с подчиненным устройством по протоколу MODBUS (в данном случае с термопреобразователем):

```
> : pT 1 VAR? US>S D>F 10.0 F/
". Temp = " F. 1 1 PRINT 1.0 1 TIMER! pT ;
(OK)
> : AR594 1 FPREC! 3.0 1 TIMER! pT
1.0 CYCLIC_ACCESS 1 0 1 1 READ_INPUTREGS 1 MODBUSSTART ;
(OK)
> AR594
(OK)
```

Сначала мы определили слово pT, которое превращает значение температуры, которое находится в 1 переменной, из представления принятого в термопреобразователе в математическое число и печатаем его на дисплее (в

термопреобразователе температура представлена в виде целого числа со знаком в диапазоне от -1900 до +1900, что отвечает -190,0ºC.. +190,0ºC). Дальше мы определили слово AR594, которое устанавливает точность отображения чисел, запускает печать температуры и настраивает 1 пакет сообщения на циклическое считывание каждые 1,0 сек. данных регистрового типа из одного входного регистра, расположенного по адресу 0 в подчиненном устройстве с логическим адресом 1 и запись этих данных в целочисленную переменную 1.
Приведем также пример обмена с тремя одинаковыми устройствами с использованием одиночного режима обмена и автоматических обратных вызовов:

```
> 0 CONSTANT scan1_a
(OK)
> 0 CONSTANT scan2_a
(OK)
> 0 CONSTANT scan3_a
(OK)
> : (scan1) scan1_a EXECUTE ;
(OK)
> : (scan2) scan2_a EXECUTE ;
(OK)
> : (scan3) scan3_a EXECUTE ;
(OK)
> : scan1 1 MODBUSSTATUS? . SPACE 6 0 PRINT
SINGLE_ACCESS 20 1000 1 1 WRITE_REGS 1 MODBUSSTART
1 MODBUSCALLBACK (scan3);
(OK)
> : scan2 1 MODBUSSTATUS? . SPACE 12 0 PRINT
SINGLE_ACCESS 21 1000 1 2 WRITE_REGS 1 MODBUSSTART
1 MODBUSCALLBACK (scan1);
(OK)
> : scan3 1 MODBUSSTATUS? . SPACE 1 0 PRINT
SINGLE_ACCESS 22 1000 1 3 WRITE_REGS 1 MODBUSSTART
1 MODBUSCALLBACK (scan2);
(OK)
> ". scan1 " FIND TO scan1_a
(OK)
> ". scan2 " FIND TO scan2_a
(OK)
> ". scan3 " FIND TO scan3_a
(OK)
> : main NONE 1 MODBUSPARAM scan1 ;
(OK)
```

Константы scan1_a, scan2_a, scan3_a и слова (scan1), (scan2) (scan3) служат для реализации обратной ссылки на слова scan1, scan2, scan3. В самих же словах scan1, scan2, scan3 реализована печать состояния предыдущего обмена на дисплее, настройки следующего одноразового обмена и настройки автоматического обратного вызова, который, в свою очередь, должен продолжить аналогичный процесс для следующего обмена. Дальше в примере идет связывание обратных ссылок. В слове main происходит настройка параметров обмена и запуск бесконечной цепочки обменов с тремя устройствами. Остановить обмен можно словом MODBUSSTOP:

```
> 1 MODBUSSTOP
(OK)
```

Система

Под термином система мы имеем в виду обще-системные аппаратные ресурсы. Для установки параметров работы системы обычно используется конфигурационное меню (там, где оно присутствует), однако, практически все параметры можно установить программно. На данный момент реализованы следующие слова, сгруппированные по определенному признаку, которые предназначены для получения статуса или установки параметров системы. Все настройки хранятся во внутренней энергонезависимой памяти. Для установления всех параметров в начальное состояние (фабричные настройки) необходимо в терминальном режиме работы указать с новой строки без предварительных пробелов команду RESTORE DEFAULTS.

Управление системой

Для установки параметров коррекции и опций системных часов существует слово SETWATCH, которое употребляется вместе с системными константами SUMMER_ON (автоматический переход включить) и SUMMER_OFF (автоматический переход выключить). Данные опции, в зависимости от комбинации слова SETWATCH с системными константами, могут принимать следующие значения (порядок слов имеет значение!):

<correction> <tzone> SUMMER_ON SETWATCH

- включается автоматический переход на зимнее/летнее время;

Числовые параметры в угловых скобках являются обязательными:

<correction>	*- целое число - ежемесячная автоматическая коррекция системных часов в секундах, может принимать значение от -360 до +360;*
<tzone>	*- целое число - часовой пояс в часах, может принимать значение от -12 до +12;*

<correction> SUMMER_OFF SETWATCH

- выключается автоматический переход на зимнее/летнее время;

<correction>	*- целое число - ежемесячная автоматическая коррекция системных часов в секундах, может принимать значение от -360 до +360;*

Для получения актуальных параметров коррекции и опций системных часов существует слово WATCHPARAM (параметры часов), которое возвращает на стек данные параметры и опции. В стековой нотации синтаксис данного слова имеет следующий вид:

```
WATCHPARAM     -     ---> <correction>,<tzone>,summer_flag
```

где summer_flag равняется значению ИСТИНА если включен автоматический

переход на зимнее/летнее время.
Для установки системных часов существует слово WATCH!, которое снимает со стека шесть верхних значений календарного времени и устанавливает текущее время системных часов:

<sec> <min> <hour> <year> <mon> <day> WATCH!

- устанавливается текущее время системных часов;

<sec> *- целое число - секунды в диапазоне 0-59.*
<min> *- целое число - минуты в диапазоне 0-59.*
<hour> *- целое число - часы в диапазоне 0-23.*
<year> *- целое число - год в диапазоне 1969-2038.*
<mon> *- целое число - месяц в диапазоне 1-12.*
<day> *- целое число - день месяца в диапазоне 1-31.*

Стековая нотация слова WATCH! имеет следующий вид (полезно для автоматизированной установки времени из программы):

```
WATCH!    СЕК,МИН,ЧАСЫ,ГОДЫ,МЕС,ДНИ    --->    -
```

Для установки типа комбинированных входов предназначено слово DIAI, которое употребляется вместе с системными константами SET_TO_V (установить по напряжению), SET_TO_I (установить по току) и SET_TO_D (установить цифровым). Дальше представлен порядок слов и констант при употреблении слова DIAI (порядок слов имеет значение!):

<type> <input> DIAI

- тип комбинированного входа <input> устанавливается равным <type>;

Числовые параметры в угловых скобках являются обязательными:

<type> *- тип входа - может принимать значение SET_TO_V, SET_TO_I или SET_TO_D;*
<input> *- целое число из диапазона от 1 до 4 - номер входа.*

Для получения актуального типа комбинированных входов существует слово DIAIPARAM (параметр входов), которое возвращает на стек код типа входа. В стековой нотации синтаксис данного слова имеет следующий вид:

```
DIAIPARAM <input>   --->  <type>
```

Для осуществления процедур калибровки реализованы следующие слова: CALPOW и CALBAT - калибровка напряжений питания ПЛК; CALV - калибровка комбинированных аналоговых входов в режиме работы по напряжению; CALI - калибровка комбинированных аналоговых входов в режиме работы по току; CALZ - калибрование нулей аналоговых входов. Порядок проведения процедур калибровки описан в отдельном эксплуатационном документе и предоставляется по требованию.
Дистанционное управление системой с помощью SMS или голосового меню

предусматривает осуществление настроек системных опций, которые позволяют определенным образом ограничить права пользователей. Для этого реализован ряд слов и системных констант.
Слово CONTROL (управлять) употребляется вместе с системными константами LOCAL (локальный), REMOTE (отдаленный) и FOR_LOYAL (для избранных), FOR_ALL (для всех) и предназначено для установления опций дистанционного управления. Данная опция, в зависимости от комбинации слова CONTROL с системными константами, может принимать следующие значения (порядок слов имеет значение!):

FOR_ALL REMOTE CONTROL

- дистанционное управление можно осуществлять с любого номера мобильного телефона;

FOR_LOYAL REMOTE CONTROL

- управление можно осуществлять лишь с первых шести номеров избранных пользователей, указанных в конфигурационном меню (или с помощью слова USERPHONE - см. дальше);

FOR_ALL LOCAL CONTROL

- дистанционное управление запрещено вообще.

Для получения актуальных опций управления системой существует слово CONTROLPARAM (параметры управления), которое возвращает на стек данные опции. В стековой нотации синтаксис данного слова имеет следующий вид:

```
CONTROLPARAM   -    --->  control_all_flag,remote_on_flag
```

где control_all_flag равняется значению ИСТИНА если дистанционное управление можно осуществлять с любого номера; remote_on_flag равняется значению ИСТИНА если дистанционное управление вообще разрешено.
Слово PASSWORD (пароль) употребляется вместе с системными константами PROTECT_BY (защитить с помощью), DISABLE (отменить) а также со строкой-паролем, определенной с помощью слова ." (точка-кавычки) и предназначено для установки опций защиты паролем. Данная опция, в зависимости от комбинации слова PASSWORD с системными константами и строкой-паролем (формально пароль считывается из выходного буфера), может принимать следующие значения (порядок слов имеет значение!):

PROTECT_BY <pwd> PASSWORD

- включается защита доступа для дистанционного управления, пароль устанавливается равным строке <pwd>

Строчный параметр в угловых скобках является обязательным:

<pwd> *строка пароля в выходном буфере (определенная с помощью слова .")*

PROTECT_BY PASSWORD

- включается защита доступа для дистанционного управления, пароль не изменяется

DISABLE PASSWORD

- выключается защита доступа для дистанционного управления.

Для получения актуальных опций защиты системы существует слово PROTECTPARAM (параметр защиты), которое возвращает на стек данную опцию. В стековой нотации синтаксис данного слова имеет следующий вид:

```
    PROTECTPARAM   -    --->  protect_on_flag
```

где protect_on_flag равняется значению ИСТИНА если включена защита доступа для дистанционного управления. Другое слово PASSWORD. (пароль-точка) печатает в выходном буфере и на терминале строку актуального пароля.
Приведем пример применения слова PASSWORD:

```
> PROTECT_BY ." 123456 " PASSWORD
123456 (OK)
```

Для автоматического ввода пин-кода задействованной SIM-карты (при активированном пин-коде), существует слово PIN, которое работает следующим образом: из выходного буфера считывается строка - пин-код и запоминается во внутренней энергонезависимой памяти. Данный пин-код автоматически вводится каждый раз при старте системы. Для отображения пин-кода в выходном буфере и на терминале существует слово PIN. (пин-точка). Приведем пример применения слова PIN и PIN.:

```
> ". 1234 " PIN
1234 (OK)
> PIN.
1234 (OK)
>
```

Для установки главного слова (или слов) управления системой на языке ForthLogic™ используется слово BOOT (загрузить), которое считывает с выходного буфера строку текста длиной не более чем 15 символов и запоминает её в энергонезависимой памяти. В дальнейшем, каждый раз при старте системы (при включении питания или при перезапуске системы) форт-система будет интерпретировать и выполнять эту строку (для полной версии контролера это происходит при условии установки в конфигурационном меню "Система" функции системы - скрипт). Приведем пример:

```
> : syrena 1 RO? NOT 1 RO! 2.0 1 TIMER! syrena ;
(OK)
> ". 1 2 RO! syrena " BOOT
(OK)
```

В этом примере мы определили слово syrena, которое непрерывно переключает контакты 1 релейного выхода, а потом сформировали скрипт управления системой, который состоит из следующих действий: переключить 2 релейный выход, и запустить на выполнение слово syrena.
Полезным также может быть слово BOOT. (загрузить-точка) которое печатает в

выходном буфере и на терминале скрипт управления системой. С учетом предыдущего примера:

```
> BOOT.
1 2 RO! syrena (OK)
```

Регистратор

Встроенный в контроллеры серии MAX Logic регистратор работает независимо от алгоритма работы контроллера и полностью настраивается с помощью конфигурационного меню. Однако, для гибкости процесса регистрации, реализован ряд слов и системных констант, которые позволяют осуществить упомянутые настройки непосредственно из программы. В процессе регистрации применяется двойная буферизация на уровне данных и двойная буферизация на уровне файлов данных (о ней отдельно упомянуто далее по тексте).
Система автоматически и абсолютно прозрачно для пользователя поддерживает механизм двух внутренних буферов данных, расположенных в оперативной памяти. Независимо от режима работы, данные сначала попадают в один из этих буферов и при его заполнении автоматически переписываются или в энергонезависимую память или передаются через существующее GPRS-соединение, что определяется режимом регистрации. Другой буфер, при этом, позволяет автоматически продолжать регистрацию данных. Такая буферизация является практически необходимой при высокой частоте регистрации а также для учета ресурса энергонезависимой памяти. Размер буферов данных составляет LOGBUF_MAX байт и позволяет значительно сократить частоту обращения к энергонезависимой памяти, которая имеет ограниченный ресурс (~100000 циклов записи). Параметр LOGBUF_MAX зависит от аппаратной платформы.
Слово LOGON (включить регистрацию) употребляется вместе с системными константами EVENTS_MODE (режим событий), USER_MODE (режим пользователя), INTERVAL_MODE (режим интервалов), TO_SD (на SD-карту), TO_FLASH (во флеш память), TO_TCP (передача по протоколу TCP), ALL_DATA (все данные), INPUTS (входы) и OUTPUTS (выходы) и предназначено для запуска процесса регистрации с одновременной установкой опций регистрации. Все настройки при этом сохраняются во внутренней энергонезависимой памяти. Данные опции, в зависимости от комбинации слова LOGON с системными константами, могут принимать следующие значения (порядок слов имеет значение и не следует забывать о пробелах!):

<data> <dest> EVENTS_MODE LOGON

- запускается процесс регистрации в режиме событий;

<data> <dest> USER_MODE LOGON

- запускается процесс регистрации в режиме пользователя;

<data> <dest> <interv> INTERVAL_MODE LOGON

- запускается процесс регистрации в режиме интервалов;

Числовые параметры в угловых скобках являются обязательными:

<data> *- набор данных, может принимать значение ALL_DATA, INPUTS или OUTPUTS;*

<dest> *- место хранения данных, может принимать значение TO_SD, TO_TCP или TO_FLASH;*

<interv> *- целое число равно интервалу в секундах.*

Слово LOGON кладет на стек логическое значение - результат своего выполнения. При этом, логическое значение ИСТИНА означает, что регистрация успешно началась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, процесс регистрации уже длится или не удалось открыть файл).
Приведем пример применения слова LOGON :

```
> ALL_DATA TO_FLASH EVENTS_MODE LOGON .
-1 (OK)
> OUTPUTS TO_SD 25 INTERVAL_MODE LOGON .
-1 (OK)
```

В первом случае мы запустили "регистрацию всех данных во флеш память в режиме событий". Во втором случае мы запустили "регистрацию состояния выходов на карту SD/MMC каждые 25 сек. в режиме интервалов".
Процесс регистрации останавливается словом LOGOFF (прекратить регистрацию), а слово LOGRUN (запустить регистрацию) предназначено для запуска регистрации с существующими опциями - оно полезно, когда возникает необходимость для частого запуска и остановки уже настроенного процесса регистрации. Слово LOGRUN кладет на стек логическое значение - результат своего выполнения. При этом, логическое значение ИСТИНА означает, что регистрация успешно началась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, процесс регистрации уже длится или не удалось открыть файл).
Для переноса данных регистрации из внутренней энергонезависимой памяти на карту SD/MMC существует слово LOG>SD. Слово LOG>SD работает следующим образом: из выходного буфера считывается строка - название файла и осуществляется попытка или открыть данный файл на карте SD/MMC, если он существует, или создать его, дальше происходит перенос данных из внутренней памяти в данный файл, а на стек кладется логическое значение - результат выполнения. При этом, логическое значение ИСТИНА означает, что данные успешно начали переноситься, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации или не удалось открыть файл). Допускается не указывать названия файла, тогда будет использован стандартный файл "datalog.txt".
Для передачи данных регистрации из внутренней энергонезависимой памяти по протоколу TCP существует слово LOG>TCP. Слово LOG>TCP осуществляет попытку перенести данные из внутренней памяти через существующее GPRS-соединение согласно протокола TCP. На стек при этом кладется логическое значение - результат выполнения данного слова: логическое значение ИСТИНА означает, что даны успешно начали переноситься а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации, не удалось открыть файл или отсутствует GPRS-соединение.)
Передача данных через существующее GPRS-соединение согласно протоколу TCP происходит без дополнительных средств проверки целостности переданных данных

- принимается допущение, что протокол TCP/IP является достаточно надежным и автоматически обеспечивает целостность данных. Однако, ситуация внезапного разрыва GPRS-соединения может привести к потере данных которые передавались в этот момент. Поэтому нужны дополнительные меры гарантирования успешной даставки данных регистрации. Среди возможных механизмов реализации этого задания была выбрана передача данных регистрации на сервер баз данных MYSQL с применением протокола HTTP. Для этого существует слово LOG>DB. Слово LOG>DB осуществляет попытку перенести данные из внутренней памяти через существующее GPRS-соединение в режиме CLIENTDB согласно протоколу HTTP. На стек при этом кладется логическое значение - результат выполнения данного слова: логическое значение ИСТИНА означает, что данные успешно начали переноситься а НЕ ИСТИНА - что возникли аппаратные проблемы (например, длится процесс регистрации, не удалось открыть файл или отсутствует GPRS-соединение в режиме CLIENTDB).
Процесс переноса данных регистрации из внутренней энергонезависимой памяти может быть достаточно длительным (в зависимости от размера зарегистрированных данных). Для выяснения состояния процесса переноса данных существует слово LOGSEND?, которое кладет на стек логическое значение - состояние процесса переноса данных. При этом, логическое значение ИСТИНА означает, что осуществляется переписывание, а НЕ ИСТИНА - что переписывание завершено.
Состояние процесса регистрации можно выяснить с помощью слова LOG?, которое кладет на стек логическое значение - состояние процесса регистрации. При этом, логическое значение ИСТИНА означает, что длится регистрация, а НЕ ИСТИНА - что регистрация остановлена.
Для выяснения свободного места во внутренней памяти существует слово FREELOG? (свободное место для регистрации), которое возвращает на стек данных количество свободного места во внутренней энергонезависимой памяти в байтах. Для полной очистки внутренней памяти регистрации существует слово LOGFORMAT (форматировать память регистрации), которое выполняет форматирование внутренней файловой системы с удалением всех зарегистрированных данных.
При регистрировании данных во внутреннюю память применяется принцип двойной буферизации на уровне файлов данных. Система автоматически и абсолютно прозрачно для пользователя поддерживает систему из двух файлов данных во внутренней памяти данных. Когда пользователь начинает регистрацию с помощью слов LOGON или LOGRUN то данные попадают в один из этих файлов (в который именно - определяет система). Последующие остановки и возобновление регистрации происходят с использованием того же файла. Так происходит до тех пор, пока пользователь не применит одно из слов LOG>SD, LOG>DB или LOG>TCP. С этого момента система автоматически выбирает для регистрирования другой файл, а переписывание данных на карту SD, в базу данных или по протоколу TCP происходит из предыдущего файла. Такой механизм позволяет осуществлять параллельную работу регистратора с переписыванием уже зарегистрированных данных на карту SD, в базу данных или по протоколу TCP. Если процесс переписывания уже зарегистрированных данных заканчивается успешно, то файл с этими данными удаляется из внутренней памяти, если же нет - то он остается и система в следующий раз использует его для дописывания новых данных. Таким образом в системе одновременно могут присутствовать до двух файлов даных регистрации. Это не является проблемой, потому что с помощью двух

последовательных процессов переписывания их можно извлечь из внутренней памяти. Для выяснения наличия зарегистрированных данных во внутренней памяти существует слово LOGFILE? (файл регистрации), которое кладет на стек числовое значение - количество файлов зарегистрированных данных. При этом, значение 1 или 2 означает, что файлы присутствуют, а 0 - что файлов зарегистрированных данных нет.
Во время регистрации данные автоматически записываются по порядку в текстовом виде в файл "datalog.txt", который размещается или во встроенной энергонезависимой памяти, или на карте памяти SD/MMC. Для разных опций набора данных регистрации одна запись имеет разный формат и всегда заканчивается символами конца строки и перевода строки - \n\r (в шестнадцатеричном представлении 0x0A и 0x0D):

- ALL_DATA - регистрируются время, дата, напряжения питания, входы и выходы в следующей форме:

"13 :04:39|19/03|18.4 13.8|22.23 20.29 0.12 134.34|10000000|0000|000\n\r", где:

13:04:39	19/03	18.4 13.8	22.23 20.29 0.12 134.34	10000000	0000	000
часы, минуты, секунды	день, месяц	напряжение питания напряжение аккумулятора, В	значение аналоговых входов AI1..AI4	логическое состояние входов DI1..DI8	логическое состояние выходов DO1..DO4	логическое состояние выходов S1..S3

- INPUTS - регистрируются время, дата и входы в следующей форме:

"13 :04:39|19/03|22.23 20.29 0.12 134.34 |10000000\n\r", где:

13:04:39	19/03	22.23 20.29 0.12 134.34	10000000
часы, минуты, секунды	день, месяц	значение аналоговых входов AI1..AI4	логическое состояние входов DI1..DI8

- OUTPUTS - регистрируются время, дата и выходы в следующей форме:

"13 :04:39|19/03|0000|000\n\r", где:

13:04:39	19/03	0000	000
часы, минуты, секунды	день, месяц	логическое состояние выходов DO1..DO4	логическое состояние выходов S1..S3

Во время регистрации данных в режиме интервалов происходит периодическая запись строк в указанном выше формате. Во время регистрации данных в режиме событий запись строк происходит лишь при выявлении любых изменений на унитарных (цифровых) входах или выходах. Дополнительно фиксируются все

входящие/исходные звонки и SMS в следующем формате строки:

"13 :04:39|19/03|SMS>|+375296127630|Hello world!\n\r", где:

13:04:39	19/03	SMS>	+375296127630	Hello world!
часы, минуты, секунды	день, месяц	тип события	номер телефона	для SMS - текст сообщения

Тип события может быть следующим:

- ">SMS" - получено входящее SMS;
- "SMS>" - отправлено исходящее SMS;
- ">VOICE" - начался входной голосовой вызов;
- "VOICE>" - начался выходной голосовой вызов;
- "HOLD" - прекратился голосовой вызов.

Аналоговые значения представляют собой целые числа, которые отвечают отсчетам 10-разрядного АЦП для соответствующего входа. Однако для полной версии контроллера, в которой реализовано конфигурационное меню и фиксированный алгоритм работы, значения отвечают масштабируемой физической величине представленной в виде чисел с фиксированной запятой и двумя цифрами после запятой. Масштабные коэффициенты принимаются из значений указанных в соответствующих пунктах конфигурационного меню контроллера. Таким образом, для полной версии контроллера регистрируются реальные показания физических величин (например, температура) от датчиков с унифицированным аналоговым выходным сигналом.

Кроме данных регистрации, которые записываются системой автоматически, существует возможность записать в системный журнал данные непосредственно из программы. Слово LOG (регистрируй) записывает содержимое выходного буфера в системный журнал. Данные записываются только во время работы регистратора в режиме пользователя и, по умолчанию, им предшествуют автоматические поля с временем, датой и ключевым словом "ForthLogic>":

"12:32:57|19/03|ForthLogic> [строка текста]\n\r"

Максимальная длина строки текста, которая считывается из выходного буфера, равняется 120 символам. Упоминавшиеся автоматические поля можно с помощью слов LOGTITLEON (включить оглавление регистрации) и LOGTITLEOFF (выключить оглавление регистрации) включать и выключать по желанию в любой момент времени. Приведем пример применения слов для регистрации данных пользователя непосредственно из программы:

```
 > : Print IF ." ON " ELSE ." OFF " THEN ;
 (OK)
 > : log1 ." Bat = " BAT? F. LOG 10.0 3 TIMER! log1 ;
 (OK)
 > : log2 ." Pump is  " 1 RO? Print LOG 15.0 2 TIMER! log2 ;
 (OK)
 > : StartLog ALL_DATA TO_FLASH USER_MODE LOGON
 IF ." Log started " LOG 1 FPREC! 10.0 3 TIMER! log1 15.0 2 TIMER! log2
 THEN ;
 (OK)
 > : StopLog ." Log stoped " LOG
 0.0 3 TIMER! STOP 0.0 2 TIMER! STOP ;
(OK)
 >
```

Слово Print является вспомогательным: Print печатает в выходном буфере словами "ON" и "OFF" логическое состояние на вершине стека. Слова log1 и log2 осуществляют, собственно, саму регистрацию данных пользователя через равные промежутки времени. Весь механизм запускается и останавливается словами StartLog и StopLog.

Подсистема питания

Слово POW? кладет на математический стек состояние напряжения основного питания в вольтах. Слово BAT? кладет на математический стек состояние напряжения питания аккумулятора в вольтах. Приведем пример применения данных слов:

```
 > POW? BAT? F. F.
 18.120234 13.780560 (OK)
```

Системное время

Слово TIME? кладет на стек три значения в такой последовательности: состояние секунд, состояние минут, состояние часов системных часов. Соответственно, при снятии со стека, сначала будут полученные часы, потом минуты и, наконец, секунды.

```
    TIME?     -     ---> СЕК,МИН,ЧАС
```

Слово DATE? кладет на стек три значения в такой последовательности: состояние годов, состояние месяцев, состояние дней системных часов. Соответственно, при снятии со стека, сначала будут полученные дни, потом месяцы и, наконец, годы.

```
    DATE?     -     ---> ГОДЫ,МЕС,ДНИ
```

Слово WDAY? кладет на стек состояние значения дней недели, при этом, воскресенье отвечает значению 0, понедельник - 1, вторник - 2 и так далее.

```
    WDAY?     -     ---> ДНИ_НЕДЕЛИ
```

Слово DST? кладет на стек флажок действия летнего времени. Значение ИСТИНА означает действие летнего времени.

```
DST?        -     ---> dst_flag
```

Слово UTC? кладет на стек состояние системных часов в виде секунд, так как это принято в операционной системе UNIX. Моментом начала отсчета секунд считается ночь с 31 декабря 1969 года на 1 января 1970, время с этого момента называют "эрой UNIX" (англ. Unix Epoch). Время UNIX согласуется со временем UTC. Способ хранения времени в виде количества секунд очень удобно использовать при сравнении дат (с точностью до секунды), а также для хранения дат: при необходимости их можно превратить в любой легкий для чтения формат.
Для превращения общепринятого представления времени во время UTC существует слово >UTC.

```
>UTC       СЕК,МИН,ЧАСЫ,ГОДЫ,МЕС,ДНИ      --->       UTC
```

Для превращения времени UTC в общепринятое представление времени существует ряд слов: >TIME, >DATE и >WDAY. Эти слова снимают со стека значения времени в виде секунд UTC и возвращают на стек время, дату и день недели. Лучше всего описать работу данных слов с помощью стековой нотации:

```
>TIME                          UTC        --->       СЕК,МИН,ЧАСЫ
>DATE                          UTC        --->       ГОДЫ,МЕС,ДНИ
>WDAY                          UTC        --->       ДНИ_НЕДЕЛИ
```

Для печати в выходном буфере и на терминале форматированного общепринятого календарного времени существует слово TIMESTAMP (метка времени). Календарное время печатается в виде (без кавычек): "час:мин:сек день/мес/год". Например"15 :02:45 10/01/11".
При этом, дни недели интерпретируются следующим образом: воскресенье отвечает значению 0, понедельник - 1, вторник - 2 и так далее.
Для сравнения между собой общепринятых дат и времен существует слово ISNOW, которое попарно сравнивает шесть верхних значений на стеке данных (убирая их при этом со стека) и возвращает на стек логическое значение ИСТИНА, только в том случае, если все сравнения истинны, и НЕ ИСТИНА в противном случае. Сравнение происходит в особом порядке:

```
ISNOW      A1,B1,C1,C2,B2,A2    --->       (A1=A2)•(B1=B2)•(C1=C2)
```

если A1=A2, B1=B2, C1=C2, то на стеке будет - 1 (ИСТИНА), иначе 0 (НЕ ИСТИНА).
Приведем пример применения упомянутых слов:

```
> DATE? . . . TIME? . . .
21 6 2008 14 7 2 (OK)
> UTC? .
1214057197 (OK)
> TIME? DATE? >UTC .
1214057199 (OK)
> 23 0 0 TIME? ISNOW . 21 6 2008 DATE? ISNOW .
0 -1 (OK)
> 0 0 15 DATE? WATCH!
(OK)
```

В последней строке приведен пример установки системного времени, равным 15:00:00 без изменения даты.

Меню пользователя

Для контроллеров с графическим дисплеем можно настраивать особое меню пользователя, которое функционирует в составе обще системного конфигурационного меню. Максимальное количество пунктов такого меню зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении. Для настройки пунктов меню пользователя, существуют слова MENU и HIDE. Слово MENU работает следующим образом: со стека снимается верхнее значение - номер меню в диапазоне от 0 до MENU_MAX, из выходного буфера считывается строка - текст, который будет отображаться в соответствующем пункте меню, из входного текста выбирается следующее слово (введено после слова MENU), которое будет выполняться при выборе соответствующего пункта меню. Дальше осуществляется настройка соответствующего пункта меню согласно заданных параметров и данный пункт меню становится видимым. Допустимо не указывать текстовую строку для отображения, тогда название пункта меню останется без изменений. Номера меню от 1 до MENU_MAX отвечают пунктам меню пользователя, а номер меню 0 отвечает за настройку функции, которая будет выполняться каждый раз при вхождении в меню "Функции Пользователя". Такая функция является полезной для начальной инициализации структуры меню пользователя. Надо также помнить о максимальной длине строки, которая будет отображаться в названии пункта меню пользователя - 15 символах, причем для пункта меню с номером 0 текст вообще не важен. Приведем простой пример применения данного слова:

```
> : inversija 3 RO? NOT 3 RO! ;
(OK)
> ". Инверсия " 1 MENU inversija
Инверсия (OK)
```

В этом примере мы определили новое слово inversija, которое производит следующие действия: на стек кладется логическое состояние релейного выхода S3, осуществляется логическая инверсия вершины стека, после чего значение со стека записывается в релейный выход S3. Дальше мы назначили это слово первому пункту меню пользователя, установив при этом название пункта "Инверсия".
Слова, которые назначены пунктам меню, а также текст, который отображается в пунктах меню пользователя не запоминаются при выключении питания контроллера. Это следует учитывать при программировании задач пользователя.
Слово HIDE работает следующим образом: со стека снимается верхнее значение - номер меню в диапазоне от 1 до MENU_MAX, дальше осуществляется выключение соответствующего пункта меню - оно становится невидимым и недоступным для пользователя. Повторное включение такого пункта меню происходит словом MENU. Таким образом, можно реализовать динамическое меню пользователя, пункты которого зависят от контекста задачи.
Определить, какой пункт был выполнен последним можно с помощью слова LASTMENU?, которое кладет на стек номер пункта меню пользователя, который был

выполнен последним. Данное слово позволяет параметризовать работу с отдельными пунктами меню.
При каждом выполнении слов MENU и HIDE, фокус меню - то есть выделенная видимая строка остается без изменений. Для большей гибкости при создании многоуровневых меню существует слово FOCUS, которое работает следующим образом: со стека снимается верхнее значение - номер меню в диапазоне от 1 до MENU_MAX и осуществляется установка фокуса на данный пункт меню - то есть он выделяется розовым цветом и отображается первой строкой в существующей структуре меню.
Дополнительно, для большего удобства использования меню, существует возможность вывода вспомогательной текстовой информации в информационном поле меню пользователя (левая верхняя часть дисплея). Для этого существует слово INFO (информация), которое с выходного буфера считывает строку текста длиной до 15 символов и показывает ее в левой части информационного поля. Данное слово работает лишь в пределах меню пользователя.
Все вышеперечисленные слова для работы с меню пользователя позволяют создавать многоуровневые динамические меню пользователя, которые не уступают по возможностям встроенному конфигурационному меню.

Клавиатура

Для настройки функций клавиш F1, F2, △, ▽, ◁, ▷ и OK существует слово BUTTON, которое применяется вместе с системными константами, имена которых отвечают названию клавиш: F1, F2, UP (△), DOWN (▽), LEFT (◁), RIGHT (▷), OK. Эти константы кладут на стек номера соответствующих клавиш. Слово BUTTON работает следующим образом: со стека снимается верхнее значение - номер клавиши в диапазоне от 1 до 7, из входного текста выбирается следующее слово (введено после слова BUTTON) которое будет выполняться при нажатии соответствующей клавиши, и осуществляется настройка данной клавиши согласно заданных параметров. Приведем пример применения слова BUTTON для настройки функции клавиши F2:

```
> : inversija 3 RO? NOT 3 RO! ;
(OK)
> F2 BUTTON inversija
(OK)
```

В процессе выполнения задачи пользователя, функции клавиш можно многократно изменять в зависимости от контекста задачи. Определенные таким образом функции клавиш F1 и F2 будут активными во всех режимах интерфейса кроме случая, когда в режиме меню происходит ввод значения, а функции клавиш △, ▽, ◁, ▷ и OK будут активными лишь в рабочем режиме интерфейса.
Слова, которые назначены клавишам, не запоминаются при выключении питания контроллера. Это следует учитывать при программировании задач пользователя.

Дисплей

Вывод информации

Для вывода текстовой информации на встроенный графический цветной дисплей существует слово PRINT (печатать). Вывод возможен только в рабочем режиме интерфейса - в других режимах форт-система блокирует вывод. Рабочее поле для вывода текстовой информации представляет собой прямоугольный участок посередине дисплея. Максимальное количество строк и символов в строке зависит от аппаратной платформы на которой функционирует форт-система и приведено в приложении. Нумерация строк - сверху вниз от 0 до ROW_MAX-1, нумерация символов - слева направо от 0 до COL_MAX-1. Текст отображается с помощью большого шрифта в одном из 9 цветов (белый, красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый, черный) на белом или на черном фоне. По умолчанию (после подачи питания на контроллер), вывод происходит черным шрифтом на белом фоне.

Слово PRINT работает следующим образом: со стека снимается два верхних значения - координаты позиции выведения текста, которые состоят из номера строки и номера символа, из выходного буфера считывается строка - текст, который будет выводиться в рабочее поле, начиная с данной позиции и происходит вывод на активный фон дисплея в активном цвете. При достижении конца строки, символы текста автоматически переносятся в следующую строку. В последней строке, при достижении конца строки, дальнейшее выведение блокируется.

Активный цвет шрифта можно установить с помощью слов WHITE (белый), RED (красный), ORANGE (оранжевый), YELLOW (желтый), GREEN (зеленый), BLUE (голубой), DEEPBLUE (синий), VIOLET (фиолетовый) и BLACK (черный), которые работают одинаковым образом: устанавливают соответствующий активный цвет текста. Все дальнейшие выводы происходят в активном цвете вплоть до его последующей смены.

Активный фон вывода (а также активный на данный момент цвет) с помощью слова INVERT (инвертировать) можно поменять на противоположный, осуществляя таким образом инверсию цветов, в результате которой белый фон становится черным (или наоборот) а цвет текста меняется на дополняющий ("негативный"). Все дальнейшие выводы происходят в данном цвете на данный фон вплоть до их последующей смены.

Для очистки рабочего поля вывода текстовой информации существует слово CLEAR (очистить), которое стирает все символы в соответствии с активным фоном - рабочее поле становится или белым или черным.

Приведем пример применения слов для работы с дисплеем:

```
> ". "Красный" " RED 0 0 PRINT
(OK)
> ". "Зеленый" " GREEN 0 1 PRINT
(OK)
> ". "Инверсия зеленого" " INVERT 0 2 PRINT
(OK)
> ". "Синий на черном" " DEEPBLUE 0 4 PRINT
(OK)
> CLEAR
(OK)
```

Слово "Красный" будет выведено, начиная с символа 0 строки 0 в красном цвете. Слово "Зеленый" будет выведено, начиная с символа 0 строки 1 в зеленом цвете. Фраза "Инверсия зеленого" будет выведена, начиная с символа 0 строки 2 в фиолетовом цвете на черном фоне. Фраза "Синей на черном" будет выведена, начиная с символа 0 строки 4 в синем цвете на черном фоне. Обратите внимание, что после выполнения слова CLEAR, рабочее поле (фон) станет черным.

Ввод информации

Для ввода числовых параметров с помощью клавиш и дисплея существует слово GET, которое работает следующим образом: с математического стека снимается верхнее значение - число, которое будет отображено в поле окна для ввода значений, из выходного буфера считывается строка - текст, который будет отображен в шапке окна для ввода значений длиной не больше 12 символов, из входного текста выбирается два следующих слова (введенных после слова GET) и на дисплее отображается стандартное окно для ввода значений.
При вводе значений в окне для ввода, перемещение курсора осуществляется клавишами ◁▷, при этом курсор подсвечивается розовым цветом. Клавишей F1 осуществляется выбор набора символов из списка: [S] - знаки пунктуации; [D] - цифры; [L] - латинские большие буквы; [l] - латинские малые буквы; [К] - кириллические большие буквы; [к] - кириллические малые буквы; при этом обозначение активного набора символов осуществляется в левом верхнем углу окна для ввода (на синем фоне). Клавишами △▽ осуществляется перебор символов из выбранного набора. Удаление символа в позиции курсора осуществляется клавишей F2, подтверждение выбранного значения - клавишей OK, а выход без внесения изменений - клавишей Esc.
Когда пользователь закончит ввод и нажмет клавишу OK, форт-система выполнит первое из указанных слов, причем, на момент выполнения данного слова, на математическом стеке будет находиться введенное значение. В случае, когда пользователь нажмет клавишу Esc, форт-система выполнит второе с указанных после слова GET слов.
Ввод числовых параметров возможен только в рабочем режиме интерфейса и в пределах меню "Функции Пользователя" - в иных случаях форт-система блокирует работу слова GET. Приведем пример применения слова GET:

```
> : print F. 0 0 PRINT ;
(OK)
> ". ТЕМП > " 65.0 GET print STOP
ТЕМП > (OK)
```

Для ввода строчных параметров с помощью клавиш и дисплея существует слово GETS, которое работает следующим образом: со стека снимается верхнее значение - номер строчной переменной, текст из которой будет отображен в поле окна для ввода значений (а в последующем она будет использована для хранения введенной строки), из выходного буфера считывается строка - текст, который будет отображен в шапке окна для ввода значений длиной не больше 12 символов. Дальше, из входного текста выбирается два следующих слова (введенных после слова GETS) и на дисплее отображается стандартное окно для ввода значений. Правила ввода значений аналогичны описанным в предыдущем параграфе для слова GET. Когда пользователь закончит ввод и нажмет клавишу OK, форт-система выполнит первое из указанных слов, причем, на момент выполнения данного слова, в указанной строчной переменной будет находиться введенная строка. В случае, когда пользователь нажмет клавишу Esc, форт-система выполнит второе с указанных после слова GETS слов.

Ввод строчных параметров также возможен лишь в рабочем режиме интерфейса и в пределах меню "Функции Пользователя" - в иных случаях форт-система блокирует роботу слова GETS. Приведем пример применения слова GETS:

```
> : print 1 STRING? 0 0 PRINT ;
(OK)
> ". Timout " 1 STRING!
(OK)
> ". СЛОВО 1 > " 1 GETS print STOP
СЛОВО 1 > (OK)
```

Звук

Для формирования звукового сигнала с помощью встроенного генератора звуковых сигналов, существует слово BEEP (сигнал), которое работает следующим образом: с математического стека снимается верхнее значение - длительность сигнала в диапазоне (0,0...64,0) секунды с шагом 0,001 сек, со стека данных снимается верхнее значение - частота звукового сигнала в диапазоне (100...10000) Гц и происходит генерация одиночного звукового сигнала с заданной частотой и длительностью.

Слово BEEP позволяет дополнительно оформить интерфейс с пользователем звуковыми событиями, что, в целом, увеличивает дружелюбие данного интерфейса. Приведем пример применения слова BEEP для генерации сигнала частотой 1000 Гц и длительностью 2 секунды:

```
> 2.0 1000 BEEP
(OK)
```

Голосовые звонки

Для работы с GSM-модулем, реализована группа слов, которые позволяют осуществить и принять голосовой звонок, подключить внешнюю гарнитуру и настроить реакцию форт-системы на DTMF сигналы.

Воспроизведение сообщений

Для воспроизведения звукового сообщения как во время осуществления или приема голосового звонка, так и в любой другой момент времени, существует слово PLAY (воспроизвести) которое работает следующим образом: из выходного буфера считывается строка - название файла в формате *.wav, который расположен на карте памяти SD/MMC (процедура создания звуковых файлов описана в приложении), из входного текста выбирается следующее слово (введено после слова PLAY) и запускается процесс воспроизведения звукового сообщения, а на стек кладется логическое значение - результат выполнения слова PLAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, файл или карта памяти отсутствуют или уже осуществляется воспроизведение, и т.п.). Когда звуковое сообщение будет полностью воспроизведено, форт-система выполнит указанное после слова PLAY слово.

Для воспроизведения словами значения числа с вершины математического стека существует слово SAY (сказать) которое работает следующим образом: из выходного буфера считывается строка - путь к набору файлов чисел в формате *.wav, который расположен на карте памяти SD/MMC, с математического стека снимается верхнее значение, которое должно находиться в пределах от -999,9 до +999,9 включительно, из входного текста выбирается следующее слово (введено после слова SAY) и запускается процесс воспроизведения словами значения числа, а на стек данных кладется логическое значение - результат выполнения слова SAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы или число находится вне установленных пределов. Когда значение числа будет полностью воспроизведено, форт-система выполнит указанное после слова SAY слово. По умолчанию, значение числа с вершины математического стека воспроизводится с указанием знака числа и с точностью 1 цифра после десятичной запятой. Воспроизведение знака числа можно отключить с помощью слова NOAUTOSAYPLUS и повторно включить с помощью слова AUTOSAYPLUS.

Путь к набору файлов для контролеров с дисплеем должен быть пустым. Набор файлов чисел в формате *.wav описан в приложении.

При создании программы, для более качественного воспроизведения звуковых сообщений, необходимо придерживаться определенных принципов, а именно: разбивать задачу на более мелкие фрагменты, которые должны выполняться с небольшими временными промежутками между ними. С этой самой целью, форт-система автоматически блокирует выведение информации на дисплей и реакцию на клавиатуру во время воспроизведения любого голосового фрагмента.

Для прекращения воспроизведения звукового сообщения или числа с вершины математического стека существует слово MUTE, которое немедленно прекращает воспроизведения звука и блокирует выполнение заданного после слов PLAY или SAY слова.

Управление вызовом

Для определения статуса голосового вызова существует слово HOOK? (крюк телефонной трубки), которое кладет на стек логическое значение, которое отвечает состоянию голосового вызова. При этом, логическое значение ИСТИНА означает,

что "трубка лежит на аппарате" - то есть, на данный момент не осуществляется ни один голосовой вызов, а НЕ ИСТИНА - что "трубка снята" - то есть, происходит голосовой вызов (входной или исходящий).
Для прекращения голосового вызова, существует слово HOLD (прекратить) которое прекращает любой голосовой вызов (входной или исходящий).

Осуществление вызова

Для осуществления голосового вызова существует слово DIAL (набрать номер) которое работает следующим образом: из выходного буфера считывается строка - номер телефона в международном или национальном формате, из входного текста выбирается два следующих слова (введенных после слова DIAL) и запускается процесс набора номера телефона, а на стек кладется логическое значение - результат выполнения слова DIAL. При этом, логическое значение ИСТИНА означает, что процесс дозвона успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM-модуль выключен или уже осуществляется вызов). Когда на втором конце телефонной линии будет снятая трубка, форт-система выполнит первое из указанных слов, а в случае, когда никто не ответит в течение 60 секунд или вызов будет прерван абонентом, форт-система выполнит второе из указанных слов.
Теперь мы можем привести пример осуществления голосового вызова:

```
> : сооб_темп 1 AI? 2.345 F* SAY HOLD DROP ;
(OK)
> : поздороваться ". hello.wav " PLAY сооб_темп DROP ;
(OK)
> : позвонить ". +375290123456 " DIAL поздороваться STOP DROP ;
(OK)
> позвонить
+375290123456 (OK)
```

Сначала мы определили новое слово "сооб_темп", которое производит следующие действия: на вершину математического стека кладется значение измеренное на первом аналоговом входе, которое множится на масштабный коэффициент - таким образом, на вершине математического стека будет находиться значение, например, температуры, которое нужно для выполнения слова SAY, дальше выполняется слово SAY - начинается воспроизведение словами значения числа, в конце которого будет выполнено слово HOLD, - положить трубку, а сразу после слова SAY будет выполнено слово DROP - чтобы отбросить со стека данных результат выполнения слова SAY, поскольку мы его не анализируем.
Дальше мы определили новое слово "поздороваться", которое производит следующие действия: в выходной буфер печатается название файла, которое нужно для выполнения слова PLAY. Дальше выполняется слово PLAY - начинается воспроизведение, в конце которого будет выполнено слово "сооб_темп", а сразу после слова PLAY будет выполнено слово DROP - чтобы отбросить со стека результат выполнения слова PLAY, поскольку мы его не анализируем.
Дальше мы определили новое слово "позвонить", которое производит следующие действия: в выходной буфер печатается номер телефона, который нужен для выполнения слова DIAL, дальше выполняется слово DIAL - начинается процесс дозвона, в конце которого будет выполнено или слово "поздороваться", - в случае,

если трубку снимут в пределах 60 секунд, или слово STOP - в случае, если трубку не снимут в пределах 60 секунд, а сразу после слова DIAL будет выполнено слово DROP - чтобы отбросить со стека результат выполнения слова DIAL, поскольку мы его не анализируем. Дальше мы просто выполняем слово "позвонить".

Прием вызова

Для приема голосового вызова, существует слово ANSWER (ответить), которое работает следующим образом: из выходного буфера считывается одна или несколько строк разделенных пробелом - список номеров телефонов в международном или национальном формате, из входного текста выбирается следующее слово (введено после слова ANSWER) и осуществляется попытка ответить на указанные номера телефонов, а на стек кладется логическое значение - результат выполнения слова ANSWER. При этом, логическое значение ИСТИНА означает, что попытка успешно завершилась и установлено голосовое соединение, а НЕ ИСТИНА - что на момент выполнения слова ANSWER не было входного вызова или голосовой вызов уже длится или входной вызов осуществлен с телефона, которого не было в списке. Когда попытка ответа успешно завершилась, форт-система выполнит указанное после слова ANSWER слово. Допустимо также не указывать никаких номеров телефонов, тогда ответ будет осуществлен на любой входной вызов, который существует на момент выполнения слова ANSWER. Приведем пример:

```
> : поздороваться ". hello.wav " PLAY HOLD DROP ;
(OK)
> : ожидать ". +375290123456 +375297654321
ANSWER поздороваться
NOT IF 3.0 3 TIMER! ожидать THEN ;
(OK)
> ожидать
+375290123456 +375297654321 (OK)
```

Слово "поздороваться" мы уже встречали. Дальше мы определили новое слово "ожидать", которое производит следующие действия: в выходной буфер печатаются номера телефонов через пробел, которые нужны для выполнения слова ANSWER, дальше выполняется слово ANSWER - делается попытка ответить на возможный входной вызов, в конце которой, в случае наличия вызова с указанных телефонов, будет выполнено слово "поздороваться", а сразу после слова ANSWER будет осуществлен анализ выполнения слова ANSWER - если результат выполнения будет НЕ ИСТИНА, то через 3 секунды опять будет выполнено слово "ожидать", а если ИСТИНА, то ничего не будет выполнено. Дальше мы просто выполняем слово "ожидать", запуская процесс ожидания входного вызова вплоть до момента его успешного приема.

Иногда возникает необходимость произвести какие-то действия в ответ лишь на факт существования входного голосового вызова без соединения. Такой способ управления обычно очень ограничен, но является бесплатным. Для приема голосового вызова без соединения, существует слово CLIP (аббревиатура, которая означает идентификацию по входящему вызову) которое работает следующим образом: из выходного буфера считывается одна или несколько строк разделенных пробелом - список номеров телефонов в международном или национальном

формате, из входного текста выбирается следующее слово (введено после слова CLIP) и осуществляется анализ возможного входного вызова на предмет соответствия списку номеров телефонов, а на стек кладется логическое значение - результат выполнения слова CLIP. При этом, логическое значение ИСТИНА означает, что анализ успешен и голосовое соединение разорвано, а НЕ ИСТИНА - что на момент выполнения слова CLIP не было входного вызова или голосовой вызов уже длится или входной вызов осуществлен с телефона, которого не было в списке. Когда анализ успешно завершился, форт-система выполнит указанное после слова CLIP слово. Допустимо также не указывать никаких номеров телефонов, тогда указанное слово будет выполнено в ответ на любой входной вызов, который существует на момент выполнения слова CLIP. Приведем пример:

```
> : вкл_бойлер 1 2 RO! ;
(OK)
> : ожидать ". +375290123456 +375297654321
CLIP вкл_бойлер
NOT IF 3.0 3 TIMER! ожидать THEN ;
(OK)
> ожидать
+375290123456 +375297654321 (OK)
```

Количество телефонов на которых происходит попытка ответить словом ANSWER или CLIP ограничивается длиной выходного буфера. Однако существует другой способ, который позволяет существенно увеличить количество телефонов, с которых ожидаются входящие вызовы. Приведем пример со словом ANSWER:

```
> : hello ". hello.wav " PLAY HOLD DROP ;
(OK)
> : wait1 ". 80156130679 " ANSWER hello DROP 3.0 1 TIMER! wait1 ;
(OK)
> : wait2 ". +375297654321 " ANSWER hello DROP 3.0 2 TIMER! wait2 ;
(OK)
> wait1 wait2
80156130679 +375297654321 (OK)
```

Слово "hello" аналогично слову "поздороваться" из предыдущего примера. Дальше мы определили два новых слова "wait1" и "wait2", которые производят подобные действия: в выходной буфер печатается номер телефона (свой для каждого слова), дальше выполняется слово ANSWER - делается попытка ответить на возможный входной вызов, в конце которой, в случае наличия вызова с указанного телефона, будет выполнено слово "hello", а сразу после слова ANSWER будет выполнено слово DROP - чтобы отбросить со стека результат выполнения слова ANSWER. Через 3 секунды слова "wait1" и "wait2" выполнятся опять. Дальше мы просто выполняем слова "wait1" и "wait2", запуская бесконечный процесс ожидания входного вызова (даже во время существования голосового соединения - робота слова ANSWER предусматривает и такую возможность).

DTMF-сигналы

При создании голосовых меню, необходимым элементом таких меню является программирование действий на нажатие клавиш мобильного телефона - то есть

установка реакции на DTMF-сигналы. Для этого существует три базовых слова: WAITKEY (ожидать клавишу), которое настраивает необходимую реакцию форт-системы на одиночные DTMF-сигналы, WAITPW (ожидать пароль), которое настраивает необходимую реакцию форт-системы на ввод с помощью DTMF -сигналов обще системного пароля доступа и WAITSTR (ожидать строку), которое настраивает необходимую реакцию форт-системы на ввод с помощью DTMF -сигналов строки цифр, причем строка цифр может представлять как целое число так и число с десятичной запятой.
В качестве управляющих голосовым меню клавиш мобильного телефона, для пользователей доступные клавиши цифр от "1" до "9" а также клавиши "0", "*" и "#". В форт-системе им отвечают коды от 1 до 12 (то есть, клавишам от "1" до "9" отвечает код от 1 до 9, клавише "0" - 10, клавише "*" - 11, а клавише "#" - 12). По умолчанию, каждое нажатие клавиши сопровождается соответствующим сигналом в динамике мобильного телефона - это значит, что форт-система восприняла факт нажатия. Такое поведение системы можно выключить с помощью слова NODTMFCONFIRM (выключить подтверждение DTMF-сигналов) или повторно включить с помощью слова DTMFCONFIRM (включить подтверждение DTMF -сигналов), причем данная опция не хранится при выключенном питании. Необходимость в выключении подтверждения DTMF-сигналов может возникнуть в случае проблем с передачей сигналов у некоторых операторов связи и в случае повышенной секретности при наборе команд или кодов.
Также можно в произвольный момент времени существования голосового соединения генерировать в динамике мобильного телефона произвольный тон соответствующей клавиши с помощью слова TONE. Слово TONE работает следующим образом: со стека снимается верхнее значение - код клавиши в диапазоне от 1 до 12 и в динамике мобильного телефона происходит генерация одиночного DTMF-сигнала с тоном, который отвечает коду клавиши.
Слово WAITKEY (ожидать клавишу) работает следующим образом: из входного текста выбирается три следующих слова (введенных после слова WAITKEY) и настраивается реакция форт-системы на DTMF-сигналы, а на стек кладется логическое значение - результат выполнения слова WAITKEY. При этом, логическое значение ИСТИНА означает, что процесс настройки прошел успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). Когда на втором конце телефонной линии будет нажата одна из клавиш мобильного телефона, форт-система выполнит первое из указанных слов, причем, на момент выполнения данного слова на вершине стека данных будет находиться код последнего полученного DTMF-сигнала - то есть код нажатой клавиши; в случае, когда ничего не будет нажато в течение 60 секунд, форт-система выполнит второе из указанных слов; и, наконец, когда связь будет прервана пользователем, форт-система выполнит третье из указанных слов. Приведем пример простого голосового меню, которое состоит из одного пункта:

```
> : анализ_кнопки 1 = IF ". true.wav " PLAY HOLD
ELSE ". by.wav " PLAY HOLD THEN DROP ;
(OK)
> : ожидать_кнопку WAITKEY анализ_кнопки HOLD STOP DROP
2 TONE ;
(OK)
> : поздороваться ". hello.wav " PLAY ожидать_кнопку DROP ;
(OK)
> : голосовое_меню
". +375290123456 " DIAL поздороваться STOP DROP ;
(OK)
> голосовое_меню
+375290123456 (OK)
```

Мы определили новое слово "анализ_кнопки", которое производит следующие действия: сравнивает код последнего полученного DTMF-сигнала с 1, если в результате сравнения на стеке будет логическое значение ИСТИНА, то воспроизводиться сообщение из файла "true.wav" и связь обрывается, иначе воспроизводиться сообщение из файла "by.wav" и связь также обрывается, в каждом случае, будет также выполнено слово DROP - чтобы отбросить со стека результат выполнения любого со слов PLAY, поскольку мы его не анализируем. Дальше мы определили новое слово "ожидать_кнопку", которое настраивает необходимую реакцию форт-системы на DTMF-сигналы и генерирует DTMF-сигнал приглашения: при нажатии любой кнопки, будет выполнено слово "анализ_кнопки", в случае, когда пройдет установленное время реакции 60 секунд, связь будет разорвана форт-системой, а в случае, когда связь будет разорвана пользователем голосового меню, не будет никакой реакции.
Дальше мы определили новое слово "поздороваться", которое воспроизводит сообщение из файла "hello.wav", после чего выполняет слово "ожидать_кнопку". Дальше было определено главное слово - "голосовое_меню", которое осуществляет голосовой звонок на указанный телефон и, в случае успеха, выполняет слово "поздороваться". Выполнение главного слова в следующей строке начинает процесс воспроизведения голосового меню.
Слово WAITPW (ожидать пароль) работает подобным слову WAITKEY образом: из входного текста выбирается три следующих слова (введенных после слова WAITPW) и настраивается реакция форт-системы на ввод обще системного пароля доступа с помощью DTMF-сигналов, а на стек кладется логическое значение - результат выполнения слова WAITPW. При этом, логическое значение ИСТИНА означает, что процесс настройки прошел успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). Когда на втором конце телефонной линии будет введен верный пароль так, как это делается во встроенном голосовом меню (при этом каждую следующую цифру нужно вводить только после сигнала приглашения и завершать введение клавишей "#"), форт-система выполнит первое из указанных слов; в случае, когда ничего не будет нажато в течение 60 секунд или пароль не верный, форт-система выполнит второе из указанных слов; и, наконец, когда связь будет прервана пользователем, форт-система выполнит третье из указанных слов. Изменим предыдущий пример простого голосового меню, введя проверку пароля (нововведения помечены темно-серым цветом):

```
> : анализ_кнопки 1 = IF ". true.wav " PLAY HOLD
ELSE ". by.wav " PLAY HOLD THEN DROP ;
(OK)
> : ожидать_кнопку WAITKEY анализ_кнопки HOLD STOP DROP
2 TONE ;
(OK)
> : поздороваться ". hello.wav " PLAY ожидать_кнопку DROP ;
(OK)
> : ожидать_пароль WAITPW поздороваться HOLD STOP DROP 6 TONE ;
(OK)
> : пароль ". parol.wav " PLAY ожидать_пароль DROP ;
(OK)
> : голосовое_меню
". +375290123456 " DIAL пароль STOP DROP ;
(OK)
> голосовое_меню
+375290123456 (OK)
```

Слово WAITSTR (ожидать строку) также работает подобным слову WAITKEY образом: из входного текста выбирается три следующих слова (введенных после слова WAITSTR) и настраивается реакция форт-системы на ввод строки цифр с помощью DTMF-сигналов, а на стек кладется логическое значение - результат выполнения слова WAITSTR. При этом, логическое значение ИСТИНА означает, что процесс настройки прошел успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). Когда на втором конце телефонной линии будет введена строка цифр, форт-система выполнит первое из указанных слов, причем, на момент выполнения данного слова, введенное число будет находиться на вершине математического стека; в случае, когда ничего не будет нажато в течение 60 секунд, форт-система выполнит второе из указанных слов; и, наконец, когда связь будет прервана пользователем, форт-система выполнит третье из указанных слов. При вводе строки цифр, каждую следующую цифру нужно вводить только после сигнала успешного введения предыдущей цифры, для ввода десятичной запятой используется клавиша "*", а завершать ввод необходимо клавишей "#". Приведем пример простого голосового меню ввода значения (температуры) в математическую переменную:

```
> : устан_темп FDUP 60.0 F<
IF 1 FVAR! ". pidtvr.wav " PLAY HOLD DROP
ELSE ". neverno.wav " PLAY HOLD DROP
THEN;
(OK)
> : ожидать_темп WAITSTR устан_темп HOLD STOP DROP 11 TONE ;
(OK)
> : поздороваться ". zaprosh.wav " PLAY ожидать_темп DROP ;
(OK)
> : голосовое_меню
". +375290123456 " DIAL поздороваться STOP DROP ;
(OK)
> голосовое_меню
+375290123456 (OK)
```

Мы определили новое слово "устан_темп", которое производит следующие действия: делает копию положенного на математический стек числа, сравнивает его значение с числом 60.0 (убирая при этом с математического стека одну из копий), если в результате сравнения на стеке данных будет положено логическое значение ИСТИНА, то происходит присвоение значения температуры математической переменной номер 1 (при этом с математического стека убирается последняя копия) далее воспроизводится сообщение из файла "pidtvr.wav" и связь обрывается, иначе воссоздается сообщение из файла "neverno.wav" и связь также обрывается.
Дальше мы определили новое слово "ожидать_темп", которое настраивает необходимую реакцию форт-системы на введение строки цифр: при нажатии кнопки "#", будет выполнено слово "устан_темп", в случае, когда пройдет установленное время реакции 60 секунд, связь будет разорвана форт-системой, а в случае, когда связь будет разорвана пользователем голосового меню, не будет никакой реакции.
Дальше мы определили новое слово "поздороваться", которое воссоздает сообщение из файла "zaprosh.wav", после чего выполняет слово "ожидать_темп".
Дальше было определено главное слово - "голосовое_меню", которое осуществляет голосовой звонок на указанный телефон и, в случае успеха, выполняет слово "поздороваться". Выполнение главного слова в следующей строке начинает процесс воспроизведения голосового меню.

Номера телефонов

В предыдущих примерах все номера телефонов были статически определены с помощью строк. Для того, чтобы иметь возможность менять номера телефонов без перенастройки всей задачи, в энергонезависимой памяти существуют специальные строчные переменные длиной 15 символов для хранения номеров телефонов пользователей. Для работы с этими переменными введено слово USER (пользователь), которое позволяет использовать при описании задачи телефонные номера пользователей - фактически считывать эти специальные переменные и слово USERPHONE (телефон пользователя) которое позволяет определять и изменять телефонные номера пользователей - то есть записывать новые значения в эти специальные переменные. Максимальное количество специальных строчных переменных для хранения номеров телефонов зависит от аппаратной платформы, на которой функционирует форт-система и приведено в приложении. Причем первые шесть переменных всегда имеют статус переменных для хранения телефонных номеров избранных пользователей, которые устанавливаются с помощью конфигурационного меню и принимают участие в политике безопасности всей системы - при определенных условиях лишь с этих телефонов можно осуществлять дистанционное управление.
Слово USER работает следующим образом: со стека снимается верхнее значение, которое может находиться в диапазоне от 1 до PHONE_MAX - это порядковый номер телефона в списке (или номер пользователя), дальше в выходном буфере и на терминале печатается номер телефона, который хранится в специальной строчной переменной с этим номером.
Слово USERPHONE работает следующим образом: со стека снимается верхнее значение, которое может находиться в диапазоне от 1 до PHONE_MAX - это порядковый номер телефона в списке (или номер пользователя), из выходного буфера считывается строка - номер телефона в международном или национальном

формате и запоминается в специальной строчной переменной с этим номером во внутренней энергонезависимой памяти. Приведем пример применения слов USER и USERPHONE:

```
> ". +375297777777 " 1 USERPHONE
+375297777777 (OK)
> 1 USER
+375297777777 (OK)
```

Также существует слово LAST (последний), которое позволяет использовать при описании задачи номер телефона последнего абонента. Слово LAST печатает в выходном буфере и на терминале номер телефона абонента, который последним дозвонился на контроллер, или послал последнее SMS-сообщение.
Приведем примеры применения последних двух слов:

```
> 3 USER LAST
80156130679 +375297654321 (OK)
```

SMS и USSD

Исходящие SMS представляют один из способов сообщить пользователю о тех или иных событиях (другой способ - это осуществление голосового звонка). Однако, входящие SMS представляют собой не просто возможный ответ пользователя, а намного более мощную концепцию реализованную с помощью языка ForthLogic™. Дело в том, что все входящие SMS непосредственно попадают (за небольшим исключением) во входной буфер текстового интерпретатора форт-системы и, соответственно, интерпретируются и выполняются. Данная концепция чрезвычайно гибкая и мощная, однако, с целью повышения надежности и безопасности всей системы, во входящих SMS заблокировано определение новых слов через двоеточие.
Как и в диалоговом режиме при работе с терминалом (*терминальном режиме*), при получении входного SMS (длина одного стандартного SMS не может превышать 160 символов), автоматически формируется исходящее SMS с ответом форт-системы - это *дистанционный режим* форт-системы. Однако, формирование выходного SMS с ответом форт-системы можно заблокировать (данная возможность уменьшает средства эксплуатации контроллера и полезная для организации взаимодействия между контроллерами - так называемой M2M - межмашинного взаимодействия). Для этого в тексте входного SMS сначала (или после пароля и по крайней мере одного пробела - если активирована защита) надо указать служебное слово NAK, отделенное от остального текста по крайней мере одним пробелом. Кроме того, формирование выходного SMS с ответом форт-системы на входящие SMS с коротких номеров (меньше 8 цифр) также заблокировано. Однако, это не мешает посылать управляющие SMS с бесплатных сайтов операторов мобильной связи и других автоматизированных серверов из сети Интернет - просто такие SMS будут без ответа.
Для организации взаимодействия при помощи SMS с автоматизированными системами мониторинга типа АСКОЕ существует возможность идентифицировать каждое переданное и принятое SMS. Для этого в тексте входного SMS сначала (или после пароля и по крайней мере одного пробела - если активирована защита) надо

указать служебное слово IDxxx, отделенное от остального текста по крайней мере одним пробелом. Параметр xxx это произвольное число в диапазоне от 1 до 65536, которое указывается без пробела и является идентификатором сообщения. Форт-система пошлет в ответ SMS, в конце которого к стандартным словам (OK) или (ERROR - ...) будет прибавлен данный идентификатор в виде (IDxxx OK) и (IDxxx ERROR - ...). Применение служебного слова IDxxx позволяет вести учет управляющих и принятых сообщений в условиях, когда операторы связи не гарантируют очередность передачи SMS через свои сети.
Служебные слова IDxxx и NAK не могут быть применены одновременно - это во-первых не логично, а во-вторых приводит к ошибке интерпретации текста SMS.
Внимание: для данной версии интерпретатора языка ForthLogic™ существует ограничение относительно использования кириллических букв в текстах SMS - можно применять только латинские. Поэтому, при развитии прикладного словаря для конкретной задачи, необходимо учитывать данный факт!
Для передачи текстовых сообщений в виде SMS, существует слово SMS которое работает следующим образом: из выходного буфера считывается номер телефона в международном или национальном формате и текст самого SMS, между которыми должен быть по крайней мере один пробел, дальше осуществляется попытка отправить SMS на указанный номер телефона, а на стек кладется логическое значение - результат выполнения слова SMS. При этом, логическое значение ИСТИНА означает, что попытка успешно завершилась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, выключен GSM-модуль). Совокупная длина текста самого SMS не должна превышать 160 символов, в случае превышения этой длины система автоматически ее ограничит с отбрасыванием лишних букв. Текст длиной больше чем одну строку (77 символов) можно ввести в выходной буфер несколько раз меньшими порциями. Приведем пример:

```
> : послатьSMS 1 USER
". Vitaju! Napruga AI2 = " 2 AI? F. ". AI3 = " 3 AI? F.
SMS DROP ;
(OK)
> 1 FPREC! послатьSMS
+375297654321 Vitaju! Napruga AI2 = 9.1 AI3 = 349.0 #00
```

Мы определили новое слово "послатьSMS", которое печатает в выходной буфер (и на терминал) номер телефона, пробел и определенным образом отформатированный текст самого сообщения, после этого выполняется слово SMS, которое "потребляет" все эти данные и делает попытку послать SMS. Словом DROP мы традиционно отбрасываем результат работы слова SMS, поскольку его не анализируем.
Как видно из примера, в текст сообщения можно включать как строки-константы, так и переменные данные - с помощью слов . (точка), F. (FLOAT - точка) и FE. (FLOAT ENGINEER - точка) могут быть использованы все слова, которые кладут свои данные на стек данных или математический стек. Для форматирования текста сообщения можно использовать слова SPACE и NEWLINE. Приведем пример отправки SMS непосредственно из терминала:

```
> ". +375297654321 Hello World! " SMS .
+375297654321 Hello World! -1 (OK)
```

Одна из форм коротких текстовых сообщений в сетях сотовой связи GSM имеет название USSD (Unstructured supplementary service data) и используется операторами для выполнения разных сервисных заданий (подключение/отключение услуг, пополнение счета, проверка баланса, и т.п.). Для работы из USSD-запросами существует слово USSD которое работает следующим образом: из выходного буфера считывается номер телефона в международном или национальном формате и строка USSD-запроса, между которыми должен быть по крайней мере один пробел, дальше осуществляется попытка отправить USSD -запрос оператору сотовой связи, а на стек кладется логическое значение - результат выполнения слова USSD. При этом, логическое значение ИСТИНА означает, что попытка успешно завершилась, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, выключен GSM-модуль, осуществляется другой USSD-запрос). Ответ оператора на удачный USSD-запрос отправляется в виде SMS на указанный номер телефона. Общая длина USSD-запроса не должна превышать 80 символов, в случае превышения этой длины система автоматически ее ограничит с отбрасыванием лишних букв. Слово USSD можно использовать как дистанционно (в дистанционном режиме работы форт-системы в тексте SMS) так и локально (выполняя по таймеру регулярную проверку баланса). Если при выполнении слова USSD осуществляется голосовой вызов, то действие слова будет задержано вплоть до окончания вызова. Приведем пример дистанционного использования USSD-запроса для пополнения счета и проверки баланса :

1. Сформировать и отправить первое SMS следующего содержания (указав при необходимости пароль и правильную последовательность цифр из ваучера пополнения):

```
PASWORD NAK LAST ". *111*12345678909876# " USSD DROP
```

в ответ придет SMS с ответом оператора на USSD-запрос (формат представления зависит от оператора и может быть пустым).

2. Сформировать и отправить второе SMS следующего содержания (указав при необходимости пароль и верную последовательность цифр для проверки баланса):

```
PASWORD NAK LAST ". *123# " USSD DROP
```

в ответ придет SMS с ответом оператора на USSD-запрос, в котором будет присутствовать информация о текущем балансе (формат представления зависит от оператора). В этом примере, для установки номера телефона, на который будет приходить ответ на USSD-запрос, мы использовали слово LAST, тем самым указав номер телефона с которого отправлялись данные SMS. Телефон также можно указать непосредственно или с помощью слова USER.
По технологическим соображениям, в дистанционном режиме работы форт-системы заблокированы непосредственное выполнение слов SMS и DIAL. Однако, это не касается слова USSD, которое спроектировано соответствующим образом. Слова SMS и DIAL могут быть выполнены в составе других слов как отложенные во

времени задания.
Входящие SMS сообщения и встроенный текстовый интерпретатор форт-системы естественным образом могут быть использованы для организации межмашинного взаимодействия между контроллерами или между контроллером и другими интеллектуальными устройствами, которые могут обрабатывать SMS.
Для межмашинного взаимодействия между контроллерами, одному контроллеру необходимо сформировать и послать другому SMS-сообщение, в котором указать необходимые действия или слова. При этом, в тексте такого SMS, сначала (или после пароля) необходимо указать служебное слово NAK - таким образом, другой контроллер, обработав в текстовом интерпретаторе управляющие слова не сформирует и не отошлет автоматический ответ своей форт-системы, который может быть абсолютно "непонятным" форт-системе контроллера, который инициировал обмен. Если в тексте исходящего SMS не указать служебную команду NAK, другой контроллер сформирует и отошлет автоматический ответ своей форт-системы, однако, такое SMS-сообщение вообще не будет воспринято форт-системой контроллера, который инициировал обмен - это мера пресечения, чтобы не создавались ситуации с цепочкой бесконечных SMS-сообщений об ошибке между двумя контроллерами.

Передача данных CSD

Прием и передача данных с помощью канала CSD происходит по принципу модемного соединения точка-точка. На одном конце канала, при этом, находится контроллер серии MAX Logic со встроенным модулем GSM. На втором конце данного канала может быть или обычный аналоговый модем, который работает в проводных телефонных сетях, или любой GSM-модем, который поддерживает протокол CSD, или другой контроллер серии MAX Logic со встроенным модулем GSM.
Аналогично, как и с входящими SMS, входной поток данных с канала CSD непосредственно попадает во входной буфер текстового интерпретатора форт-системы и, соответственно, интерпретируется и выполняется. Как уже упоминалось, данная концепция чрезвычайно гибкая, однако в данном случае она более мощная, чем аналогичная концепция входящих SMS потому, что во входном потоке не заблокировано определение новых слов через двоеточие. Это сделано потому, что пользователю системы получить доступ к каналу CSD с обычного телефона без компьютера и специального программного обеспечения невозможно, и соответственно вероятность случайно нарушить роботу системы практически нулевая. Однако для систем типа M2M - межмашинного взаимодействия или систем мониторинга типа АСКОЕ такая опция, как дистанционное определение новых слов через двоеточие, может пригодиться.
При получении входного потока данных с помощью канала CSD (длина одной порции данных не может превышать 160 символов), автоматически формируется выходной поток данных с ответом форт-системы - это *CSD режим* форт-системы. Отличие этого режима от режима при работе с терминалом (терминального режима) заключается в том, что не работает слово WORDS, а в исходящем потоке отсутствуют все сообщения об ошибках и информация в квадратных скобках генерируемая некоторыми словами (".S", ".FS" и т.п.), отсутствуют все текстовые сообщения форт-системы в круглых скобках ("OK", "ERROR-."., и т.п.) а также

отсутствует символ приглашения форт-системы (">"). Это позволяет в режиме on-line направлять ответ одной форт-системы непосредственно на вход другой, что упрощает построение систем типа M2M. Однако, в случае возникновения ошибок, сообщения о них отображаются на терминале при условии, что существует активное подключение к терминалу.
Из программы на языке ForthLogic™ можно инициировать подключение к каналу CSD, отправить данные в виде строки текста, инициировать выполнение любого известного слова в случае получения данных через канал, разорвать соединение и посмотреть состояние соединения. Для этого реализован ряд слов, которые представлены далее по тексту.
Для осуществления подключения к каналу CSD существует слово CONNECTCSD (подключиться к каналу CSD) которое работает следующим образом: с выходного буфера считывается строка - номер телефона в международном или национальном формате, из входного текста выбирается два следующих слова (введенных после слова CONNECTCSD) и запускается процесс набора номера телефона, а на стек кладется логическое значение - результат выполнения слова CONNECTCSD. При этом, логическое значение ИСТИНА означает, что процесс подключения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM-модуль выключен или уже осуществляется другой выходной или входной вызов). Когда успешно образуется канал CSD, форт-система выполнит первое из указанных слов, в другом случае, когда канал не сможет образоваться в течение 60 секунд или не сможет образоваться в принципе (попытка образовать канал не с модемом), форт-система выполнит второе из указанных слов.

Для передачи сообщения в канал CSD существует слово SENDCSD (отправить данные в канал CSD) которое работает следующим образом: из выходного буфера считывается строка сообщения, из входного текста выбирается следующее слово (введено после слова SENDCSD) и строка передается через канал CSD, а на стек кладется логическое значение - результат выполнения слова SENDCSD. При этом, логическое значение ИСТИНА означает, что процесс передачи успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM-модуль выключен или канал CSD закрыт). В случае, когда в ответ придет сообщение со второго конца канала CSD, оно попадет сначала в интерпретатор форт-системы (см. выше по тексту), после чего форт-система выполнит указанное после слова SENDCSD слово.
Для разрыва канала CSD в какой либо момент времени существует слово BREAKCSD (разорвать канал CSD), а для получения статуса канала существует слово STATUSCSD (состояние канала CSD), которое возвращает на стек логическое значение ИСТИНА если канал передачи данных CSD открыт и логическое значение НЕ ИСТИНА в противоположном случае.
Приведем пример использования вышеупомянутых слов для построения системы типа M2M - межмашинного взаимодействия, в котором образуется канал CSD между двумя контроллерами, дальше один из них (ГЛАВНЫЙ) посылает запросы и обрабатывает ответы таким образом, что состояние цифровых входов другого контроллера (ПОДЧИНЕННОГО) отображается на цифровых выходах ГЛАВНОГО:

```
 > 0 CONSTANT send_a
(OK)
 > : (send) send_a EXECUTE ;
(OK)
 > : read 4 DO! 3 DO! 2 DO! 1 DO! 0.1 1400 BEEP (send);
(OK)
 > : send ". 5 DI? . 6 DI? . 7 DI? . 8 DI? . 0.1 2400 BEEP "
SENDCSD read  DROP ;
(OK)
 > ". send " FIND TO send_a
(OK)
 >  : start 1 USER CONNECTCSD send STOP DROP ;
(OK)
 >  : stop BREAKCSD ;
(OK)
```

Данная программа записывается в память только одного контроллера - ГЛАВНОГО. Дальше ГЛАВНЫЙ контролер инициирует соединение с ПОДЧИНЕННЫМ и начинает управляемый обмен состоянием цифровых входов. Рассмотрим данный пример детальнее.

Канал данных открывается словом "start", как только это происходит, выполняется слово "send", которое посылает запрос "5 DI? . 6 DI? . 7 DI? . 8 DI? . 0.1 2400 BEEP". Согласно этому запросу, другой контроллер - ПОДЧИНЕННЫЙ выводит в свой выходной буфер состояние цифровых входов в виде цифр 0 или -1 и воспроизводит короткий звуковой сигнал. Дальше все содержимое его выходного буфера попадает в канал CSD в качестве ответа. Этот ответ сначала попадает во входной буфер ГЛАВНОГО контроллера, который интерпретирует цифры и кладет их на свой стек. После этого форт-системой выполняется слово "read", которое снимает цифры со стека и записывает их в соответствующие цифровые выходы. Дальше оно воспроизводит короткий звуковой сигнал и опять выполняет слово "send" (через механизм обратной ссылки, см. раздел "Векторное выполнение") - таким образом, весь цикл запрос-ответ повторяется бесконечное количество раз.

Канал CSD можно использовать для неуправляемого из программы на языке ForthLogic обмена данными с устройствами присоединенными к последовательному интерфейсу RS485. Фактически образуется "прозрачный" модемный канал между устройствами и программой для ЭВМ, которая инициировала данный канал с помощью обычного модема или GSM-модема и не накладываются никакие ограничения на протокол обмена по данному каналу. Однако, следует принять во внимание часовые задержки, которые возникают из природы коммутированных во времени каналов передачи в сетях GSM. Практически, это заключается в том, что в непрерывном потоке данных возможные паузы до 20 мс. Также следует помнить, что скорость обмена фиксирована и составляет 9600 бит/сек.

Для образования модемного канала применяется слово MODEMCSD. Данное слово можно применять лишь тогда, когда предварительно образован канал передачи данных CSD: оно возвращает на стек логическое значение ИСТИНА если модемный канал успешно образовался и логическое значение НЕ ИСТИНА в противоположном случае. Практически, данное слово можно применить как в терминальном режиме (из программы на ForthLogic или непосредственно из терминала) так и в CSD режиме форт-системы (например, из программы для ЭВМ, которая инициирует модемный канал).

Когда образуется модемный канал, все другие способы обмена через GSM (голосовые сообщения, SMS, GPRS) не работают. Канал можно разорвать путем расторжения CSD соединения с противоположного конца линии. При этом возобновляется работа через остальные каналы GSM (голосовые сообщения, SMS, GPRS и CSD в обычном режиме).

Передача данных GPRS

GPRS режим форт-системы

Прием и передача данных с помощью канала GPRS происходит по принципу клиент-сервер с использованием протокола TCP. При обычном клиент-серверном соединении, над протоколом TCP выполняется текстовый протокол обмена с форт-системой подобный обычному терминальному режиму работы: при получении входного потока данных с помощью канала GPRS (длина одной порции данных не может превышать 160 символов), автоматически формируется выходной поток данных с ответом форт-системы - это GPRS режим форт-системы. При этом, в выходном потоке отображаются все ошибки и служебная информация форт-системы что не приемлемо в случае межмашинного взаимодействия (M2M) между двумя контролерами. Для отключения отображения служебной информации и ошибок существует слово M2MTCPON (включить режим M2M через протокол TCP). Повторно включить нормальный режим позволяет слово M2MTCPOFF (выключить режим M2M через протокол TCP). При пропадании питания данная опция не сохраняется, а при подаче питания по умалчиванию устанавливается нормальный режим передачи данных через протокол TCP.

Принципы клиент-серверного соединения

Необходимо обстоятельно остановиться на технологии клиент-сервер. На одном конце канала всегда находится сервер а на втором клиент и потому существует два сценария построения соединения. Первый сценарий заключается в том, что в качестве сервера удобно использовать серверную программу которая работает на ЭВМ, которая имеет фиксированный IP-адрес во всемирной сети INTERNET. В этом случае контролер серии ES-ForthLogic со встроенным модулем GSM выступает в качестве клиента, который инициирует соединение с сервером. Преимущество такого сценария состоит в том, что к серверу одновременно может быть подсоединено очень много клиентов (сотни и тысячи) и нет необходимости в получении специальных SIM-карт с доступными с из вне IP-адресами.
Другой сценарий заключается в том, что в качестве сервера выступает контролер серии ES-ForthLogic со встроенным модулем GSM который может одновременно принять лишь *одно* клиентское соединение. При этом необходимо гарантировать доступность IP-адреса контролера во всемирной сети INTERNET - это достигается при применении специальных SIM-карт и специальных телеметрических тарифов избранного оператора сети GSM. Преимущество данного сценария заключается в том, что на втором конце можно использовать клиентскую программу которая работает на ЭВМ и имеет любой доступ к сети INTERNET. Такая программа должна одновременно образовывать столько клиентских соединений, сколько необходимо.

Слова для роботы с GPRS

Из программы на языке ForthLogic™ можно инициировать подключение к GPRS, установить соединение в режиме клиента или сервера, отправить данные в виде строки текста, разорвать соединение и посмотреть состояние соединения. Для этого реализован ряд слов которые представлены далее по тексту. Также реализована поддержка передачи данных, которые накапливаются во время работы регистратора, через существующее GPRS-соединение по протоколу TCP/IP или на сервер баз данных MYSQL с применением протокола HTTP - более детально это описано в соответствующем разделе. Кроме работы в режиме клиента или сервера с другим контролером или программой которая работает на ПК, существует возможность поддерживать постоянное соединение со специализированным OPC-сервером. При этом применяется первый сценарий технологии клиент-сервер (описанный выше).

Начальные настройки

Для выяснения возможности подключения к сервису GPRS, существует слово STATUSGPRS (состояние сервиса GPRS), которое возвращает на стек логическое значение ИСТИНА, если сервис GPRS для данной SIM-карты активирован. Если сервис GPRS не активируется по умалчиванию (свойство тарифа) или случился сбой сети, то для активации сервиса существует слово ATTACHGPRS (подключить сервис GPRS) которое делает попытку подключиться к сервису GPRS а на стек возвращает логическое значение ИСТИНА, если попытка началась удачно и НЕ ИСТИНА если возникли проблемы (например, GSM модуль выключен). Процесс подключения сервиса может занимать до 10 секунд и проверяется с помощью слова STATUSGPRS.

Для хранения параметров канала GPRS таких как APN и IP в энергонезависимой памяти существуют специальные одноименные строчные переменные. Длина переменных IP и APN составляет 70 символов. Для работы с этими переменными существуют слова IP, APN, SETIP и SETAPN. Слова IP и APN печатают в выходном буфере и на терминале содержимое одноименных переменных. А слова SETIP и SETAPN запоминают текст из выходного буфера в соответствующих переменных IP и APN. В зависимости от режима работы канала GPRS, семантика содержимого этих переменных может изменяться.

Для хранения параметров работы со специализированным OPC-сервером или сервером баз данных MYSQL, в энергонезависимой памяти также существуют специальные строчные переменные ID (идентификатор) и URL (адрес). Переменная ID предназначена для хранения имени и пароля для доступа к удаленному серверу. Имя и пароль должны быть разделенными пробелом и иметь длину не более чем 15 символов. Переменная URL предназначена для указания адреса внутри хоста к которому происходят обращения с применением протокола HTTP. Длина переменной ID составляет 30 символов а переменной URL - 70 символов. Для работы с этими переменными существуют слова ID, URL, SETID и SETURL. Слова ID и URL печатают в выходном буфере и на терминале содержимое одноименных переменных. А слова SETID и SETURL запоминают текст из выходного буфера в соответствующих переменных ID и URL.

Для конфигурации протокола работы TCP и подключение к GPRS существует слово CONFIGTCP (конфигурация протокола TCP) которое используется вместе с

системными константами CLIENT (клиент), CLIENTDB (клиент сервера баз данных MYSQL), CLIENTOPC (клиент OPC-сервера) и SERVER (сервер) и работает следующим образом: со стека снимается число, которое отвечает одной из системных констант CLIENT или SERVER, из выходного буфера считывается строка которая состоит из APN, имени и пароля разделенных пробелами и происходит настройка протокола работы TCP и подключение к GPRS. На стек при этом кладется логическое значение - результат выполнения слова CONFIGTCP: логическое значение ИСТИНА означает, что процесс настройки и подключения осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствующий сервис GPRS). APN который указывается в выходном буфере, предоставляется оператором для вхождения в сеть GPRS. Иногда также предоставляется имя и пароль, однако, если они отсутствуют, то строка состоит только из APN.

Для установления номера порта, который будет "прослушиваться" в режиме работы типа SERVER, существует слово CONFIGPORT, которое работает следующим образом: со стека снимается число в диапазоне от 0 до 65535 и происходит установление номера порта. На стек при этом кладется логическое значение - результат выполнения слова CONFIGPORT: логическое значение ИСТИНА означает, что процесс установления осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). Номер порта по умалчиванию всегда равняется 2020 и его настройка имеет смысл только для работы в режиме SERVER. При пропадании питания номер порта не сохраняется.

К ресурсам в сети INTERNET можно обращаться через их IP-адреса (например 124.34.1.145) или через предназначенные для них URL-адреса (например: www.es.ua). По умолчанию установлен режим IP-адресов - так удобнее для телеметрических систем. Однако, в таких задачах как работа с сервером баз данных MYSQL возникает необходимость указывать URL-адреса. Для установления режима адресации с помощью URL-адресов существует слово DNSMODE (режим имен), которое работает следующим образом: из выходного буфера считывается строка, которая состоит из двух IP-адресов DNS-серверов разделенных пробелами и происходит установление режима URL-адресов. На стек при этом возвращается результат выполнения слова DNSMODE: логическое значение ИСТИНА означает, что процесс установления осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). Допускается не указывать IP-адреса DNS-серверов, тогда для поиска URL-адресов система обращается к стандартным открытым DNS-серверам 208.67.220.220 и 208.67.222.222. Для установления режима адресации с помощью IP-адресов назначено слово IPMODE (режим IP-адресов) которое возвращает на стек логическое значение - результат своего выполнения: логическое значение ИСТИНА означает, что процесс установления осуществился успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). При пропадании питания опция режима адресации не сохраняется.

Установление и разрыв соединения.

Для установления соединения согласно протоколу TCP существует слово OPENTCP (установить соединение по протоколу TCP), которое работает следующим образом:

если протокол был настроен на работу в режиме CLIENT или CLIENTOPC, то из выходного буфера считывается строка которая состоит из IP-адреса и номера порта разделенных пробелом, если протокол был настроен на работу в режиме CLIENTDB, то из выходного буфера считывается строка которая состоит из URL-адреса хоста, на котором установлен специализированный скрипт, из входного текста выбирается два очередных слова (введенных после слова OPENTCP) и происходит установление соединения. На стек при этом кладется логическое значение - результат выполнения слова OPENTCP: логическое значение ИСТИНА означает, что процесс соединения начался успешно, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен или отсутствует сервис GPRS). Когда успешно установится соединение согласно протоколу TCP, форт-система выполнит первое из указанных после слова OPENTCP слов, в противоположном случае, когда соединение не сможет образоваться на протяжении 120 секунд, форт-система выполнит второе из указанных слов. Адреса в выходном буфере указываются только для режимов соединения типа CLIENT, CLIENTDB или CLIENTOPC и являются адресами сервера с которым происходит попытка соединения. Для режима соединения типа SERVER данное слово лишь запускает сервер, который ожидает соединения со стороны клиента.
Для передачи сообщения согласно протоколу TCP в любом обычном режиме соединения типа CLIENT или SERVER существует слово SENDTCP (отправить данные по протоколу TCP) которое работает следующим образом: из выходного буфера считывается строка сообщения и передается по протоколу TCP, а на стек кладется логическое значение - результат выполнения слова SENDTCP. При этом, логическое значение ИСТИНА означает, что процесс передачи успешно осуществился, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, GSM модуль выключен, отсутствует сервис GPRS или канал GPRS закрыт). Максимальная длина сообщения составляет 160 символов.
Передача сообщений в режиме соединения типа CLIENTOPC является невозможной, поскольку обмен данными с OPC-сервером происходит по специализированному внутреннему протоколу. OPC-сервер делает запросы а система автоматически и прозрачно для пользователя осуществляет обработку и формирование ответа. Передача сообщений в режиме соединения типа CLIENTDB также является невозможной, поскольку данные передаются автоматически из внутренней энергонезависимой памяти на сервер баз данных MYSQL с применением протокола HTTP. Данный процесс начинается словом LOG>DB и полностью контролируется системой.
Для разрыва соединения типа CLIENT, CLIENTDB или CLIENTOPC в любой момент времени существует слово CLOSETCP (разорвать соединение согласно протоколу TCP). После разрыва соединения типа CLIENT, CLIENTDB или CLIENTOPC, можно осуществить попытку соединения с другим сервером.
Для отключения от GPRS в любой момент времени существует слово SHUTTCP (разорвать подключение согласно протокола TCP). После отключения от GPRS, необходимо опять подключаться с помощью слова CONFIGTCP.

Статус соединения

Для получения статуса соединения согласно протоколу TCP существует слово STATUSTCP (состояние соединения согласно протоколу TCP), которое возвращает на стек логическое значение ИСТИНА если в режиме соединения типа CLIENT,

CLIENTDB или CLIENTOPC существует физическое соединение с отдаленным сервером. В режиме соединения типа SERVER слово STATUSTCP возвращает на стек логическое значение ИСТИНА если сам сервер является активным (при этом может отсутствовать само соединение с клиентом).
Для получения статуса соединения с клиентом согласно протоколу TCP в режиме соединения типа SERVER существует слово STATUSSERVER (состояние сервера), которое возвращает на стек логическое значение ИСТИНА если в режиме соединения типа SERVER существует физическое соединение с отдаленным клиентом.
Для получения статуса работы с OPC-сервером в режиме соединения типа CLIENTOPC существует слово STATUSOPC (состояние OPC), которое возвращает на стек логическое значение ИСТИНА если в режиме соединения типа CLIENTOPC существует логическое соединение с OPC-сервером. Логическое соединение с OPC-сервером означает постоянный автоматический обмен данными по специализированному внутреннему протоколу.
После подключения к GPRS, можно узнать собственный IP-адрес, который автоматически предоставляет оператор, с помощью слова LOCALIP (собственный IP-адрес). Данное слово печатает в выходном буфере и на терминале строку собственного IP-адреса.
Если в режиме соединения типа SERVER существует физическое соединение с отдаленным клиентом, то можно узнать о IP-адресе клиента с помощью слова REMOTEIP (IP-адрес клиента). Данное слово печатает в выходной буфер и на терминал строку IP-адреса клиента. Также в режиме соединения типа SERVER можно ограничить круг отдаленных клиентов, если записать в энергонезависимую переменную IP IP-адрес клиента которому разрешено подключаться к серверу. Если в эту переменную записать пустую строку, то к серверу могут подключаться все клиенты без ограничений.

Особенности режима CLIENTOPC

Как уже упоминалось, в режиме CLIENTOPC обмен данными с OPC-сервером происходит автоматически по специализированному внутреннему протоколу. OPC-сервер это специализированный сервер который с одной стороны осуществляет обмен данными с контролером через канал GPRS а с другой стороны предоставляет доступ к этим данным OPC-клиентам, в качестве которых чаще всего выступают разнообразные системы дистанционного управления и мониторинга технологических процессов (АСУТП, SCADA). OPC-сервер поддерживает спецификации OPC/DA v1 и v2 и должен работать на ПК, которая имеет фиксированный IP-адрес во всемирной сети INTERNET. Во время работы с OPC-сервером контроллер выступает в качестве клиента и потому можно применить обычные SIM-карты с поддержкой GPRS-соединения. Также можно применять SIM-карты с поддержкой VPN. Инициатором соединения с OPC-сервером выступает контроллер. Для успешного соединения, контроллер должен пройти аутентификацию, которая заключается в совпадении имени и пароля, которые задаются с помощью переменной ID. Одновременно к одному OPC-сервера может быть подсоединено много контролеров, что позволяет строить распределенные системы дистанционного управления и мониторинга технологических процессов.
Принцип обмена данными с OPC-сервером заключается в том, что OPC-сервер делает запросы а система автоматически и прозрачно для пользователя

осуществляет обработку запросов и формирование ответа. При этом OPC-сервер может свободно устанавливать и читать все глобальные переменные, глобальные переменные в формате чисел с плавающей запятой и глобальные битовые переменные. Программа на языке ForthLogic™ не имеет непосредственного доступа к обмену данными с OPC-сервером, однако может взаимодействовать с OPC-сервером (а через него и с системой АСУТП или SCADA) через глобальные переменные.

Особенности режима CLIENTDB

В режиме соединения типа CLIENTDB данные передаются автоматически из внутренней энергонезависимой памяти на сервер баз данных MYSQL с применением протокола HTTP. Как уже упоминалось, данный процесс начинается словом LOG>DB и полностью контролируется системой.
Принцип передачи данных на сервер баз данных MYSQL заключается в том, что система сначала передает файл из внутренней энергонезависимой памяти на HTTP-сервер с помощью метода POST. Этот файл принимается и обрабатывается специализированным скриптом, имя которого необходимо указывать в переменной URL. Дальше этот файл анализируется и данные из него передаются в таблицу на сервер баз данных MYSQL. В режиме соединения типа CLIENTDB контроллер выступает в качестве клиента и потому можно применить обычные SIM-карты с поддержкой GPRS-соединения. Для успешного соединения с HTTP-сервером, контроллер должен пройти аутентификацию, которая заключается в совпадении имени и пароля, которые задаются с помощью переменной ID. Кроме того, программа на языке ForthLogic™ должна обеспечить верный формат и фиксированное количество полей зарегистрированных данных для одной записи в файл регистрации, например:

"1304585524;22.233455;-1;13434", где:

1304585524	22.233455	-1	13434
Поле времени в формате UTC	Поле числа с плавающей запятой, например температура	Поле флажка, например состояние входа	Поле целого числа, например количество импульсов на входе

Разделителем между полями данных должен быть символ ";" (точка с запятой), в конце записи разделитель не применяется. Первым полем должно быть время в формате UTC. Первое поле является обязательным. Остальные поля и их тип должны строго соответствовать параметрам соответствующей таблицы на сервере баз данных MYSQL. Каждому зарегистрированному (такому, который прошел аутентификацию) контроллеру ставится в соответствие одна фиксированная таблица с фиксированной структурой на сервере баз данных MYSQL.

Пример соединения типа SERVER и CLIENT

Приведем пример установления соединения типа SERVER. В этом примере используется SIM-карта с фиксированным IP-адресом и порт сервера по умолчанию (2020). Клиент должен устанавливать соединение с данным адресом и портом. Кроме того, клиент может быть только один и также должен иметь фиксированный IP-адрес.

```
 > ." static.umc.ua " SETAPN ." 192.168.0.1 " SETIP
(OK)
 > 0 CONSTANT checkservice_a
(OK)
 > : (checkservice) checkservice_a EXECUTE ;
(OK)
 > : reconfig SHUTTCP 3.0 1 TIMER! (checkservice) ;
(OK)
 > : task STATUSTCP NOT
 IF reconfig
 ELSE 1.0 1 TIMER! task
 THEN ;
(OK)
 > : openfail 3.0 1 TIMER! (checkservice) ;
(OK)
 > : start OPENTCP task openfail DROP ;
(OK)
 > : checkservice
 STATUSGPRS
 IF APN SERVER CONFIGTCP
   IF 3.0 1 TIMER! start
   ELSE 3.0 1 TIMER! reconfig
   THEN
 ELSE ATTACHGPRS DROP 10.0 1 TIMER! checkservice
 THEN ;
(OK)
 > ." checkservice " FIND TO checkservice_a
(OK)
 > : checknet
 SIGNAL? -1 =
 IF 1.0 1 TIMER! checknet
 ELSE 3.0 1 TIMER! checkservice
 THEN ;
(OK)
 > checknet
(OK)
 >
```

Сначала мы настроили параметры GPRS-сервиса и позволили доступ только для клиента с IP-адресом 192.168.0.1. Слово checknet проверяет состояние GSM-модуля, когда он включится и успешно зарегистрируется в сети GSM, будет выполнено слово checkservice, которое проверяет наличие GPRS-сервиса и осуществляет попытку подключиться к сервису в случае его отсутствия.. В случае его наличия происходит попытка подключить и настроить GPRS в режиме сервера. При успешной попытке будет выполнено слово start. При неудачной попытке будет выполнено слово reconfig, которое отключает GPRS и повторяет попытку путем выполнения слова checkservice. Слово start активирует сервер и в случае его успешной активации запускает на выполнение слово task, иначе все повторяется путем выполнения слова checkservice. Слово task непрерывно проверяет состояние сервера и если он отключился (пропала сеть, и тому подобное) то будет выполнено слово reconfig, которое попробует возобновить работу сервера.

Теперь приведем пример установления соединения типа CLIENT.

```
 > ." static.umc.ua " SETAPN ." 192.168.0.100 2020 " SETIP
(OK)
 > 0 CONSTANT checkservice_a
(OK)
 > : (checkservice) checkservice_a EXECUTE ;
(OK)
 > : reconfig SHUTTCP 3.0 1 TIMER! (checkservice) ;
(OK)
 > : task STATUSTCP NOT
 IF reconfig
 ELSE 1.0 1 TIMER! task
 THEN ;
(OK)
 > : openfail 3.0 1 TIMER! (checkservice) ;
(OK)
 > : start IP OPENTCP task openfail DROP ;
(OK)
 > : checkservice
 STATUSGPRS
 IF APN CLIENT CONFIGTCP
   IF 3.0 1 TIMER! start
   ELSE 3.0 1 TIMER! reconfig
   THEN
 ELSE ATTACHGPRS DROP 10.0 1 TIMER! checkservice
 THEN ;
(OK)
 > ." checkservice " FIND TO checkservice_a
(OK)
 > : checknet
 SIGNAL? -1 =
 IF 1.0 1 TIMER! checknet
 ELSE 3.0 1 TIMER! checkservice
 THEN ;
(OK)
 > checknet
(OK)
 >
```

Сначала мы настроили параметры GPRS-сервиса и указали параметры доступа к серверу. Остальные слова практически аналогичны к предыдущему примеру. Лишь слово start осуществляет попытку подключиться к серверу, а слово checkservice настраивает режим клиента.
После установления соединения в любом обычном режиме типа CLIENT или SERVER форт-система будет работать в GPRS-режиме и можно отсылать из программы строчные данные или данные регистрации (данные регистрации могут отсылаться или в процессе накопления или в виде файла сохраненного во внутренней памяти - см. раздел "Регистратор").

Гарнитура

Для подключения внешней гарнитуры (комплекта оборудования для осуществления разговора) во время осуществления голосового вызова (входного или исходящего) существует слово MIC (гарнитура). Выполнение данного слова приводит к переключению аудио-канала GSM-модуля с цифрового синтезатора на внешнюю гарнитуру присоединенную либо к входам SPK, MIC+ и MIC- либо к разъему аудио MIC/EAR. При этом на стек кладется логическое значение - результат выполнения слова MIC: логическое значение ИСТИНА означает, что процесс переключения прошел успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов). При успешном выполнении слова MIC, воспроизведение голосовых сообщений становится невозможным - аудио-канал GSM-модуля будет работать с внешним источником звука, а во внешнем динамике будет воспроизводится аудио из GSM-модуля. После того, как связь будет прервана (пользователем или форт-системой), аудио-канал GSM-модуля автоматически переключается на цифровой синтезатор - то есть в следующем голосовом вызове опять становится доступным воспроизведение голосовых сообщений, а аудио из GSM-модуля не будет слышно во внешнем динамике.

Аудио-канал GSM-модуля также можно переключить в любой момент времени на цифровой синтезатор с помощью слова VOICE. Выполнение данного слова приводит к переключению аудио-канала GSM-модуля на цифровой синтезатор. При этом на стек кладется логическое значение - результат выполнения слова VOICE: логическое значение ИСТИНА означает, что процесс переключения прошел успешно, а НЕ ИСТИНА - что возникли проблемы (например, на данный момент не осуществляется ни один голосовой вызов).

Прием DTMF-сигналов возможен как при роботе GSM-модуля от цифрового синтезатора, так и при работе от внешней гарнитуры.

Для регулирования чувствительности внешней гарнитуры существует слово MICLEVEL (чувствительность гарнитуры) которое работает следующим образом: с вершины стека данных снимается число - новое условное значение чувствительности в диапазоне от 0 до 15 и устанавливается заданная чувствительность. Увеличение или уменьшение условной чувствительности на 1 приводит к увеличению или уменьшению реальной чувствительности гарнитуры в 1,1885 раз. По умолчанию, условная чувствительность равна 2, а ее новое значение не сохраняется при выключении питания.

Для регулировании громкости в гарнитуре, существует слово SPKLEVEL (громкость гарнитуры), которое работает следующим образом: с вершины стека данных снимается число - новое условное значение громкости в диапазоне от 0 до 100 и устанавливается заданная громкость. Уменьшение условной громкости приводит к уменьшению реальной громкости в динамике и наоборот. По умолчанию, условная громкость равна 100, а ее новое значение не сохраняется при выключении питания.

Воспроизведение сообщений через акустическую систему

Некоторые модели контролеров оборудованы акустической системой, через которую можно воспроизводить сообщение и числа аналогично тому как это

делается при воспроизведении голосового меню.
Для воспроизведения звукового сообщения через аудиосистему в любой момент времени, существует слово AUDIOPLAY (воспроизвести аудио) которое работает следующим образом: из выходного буфера считывается строка - название файла в формате *.wav, который расположен на карте памяти SD/MMC (процедура создания звуковых файлов описана в приложении), из входного текста выбирается очередное слово (введено после слова AUDIOPLAY) и запускается процесс воспроизведения звукового сообщения, а на стек кладется логическое значение - результат выполнения слова AUDIOPLAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы (например, файл или карта памяти отсутствуют или уже осуществляется воспроизведение через аудиосистему и тому подобное). Когда звуковое сообщение будет полностью воспроизведено, форт-система выполнит указанное после слова AUDIOPLAY слово.
Для воспроизведения через аудиосистему словами значения числа с вершины математического стека существует слово AUDIOSAY (сказать) которое работает следующим образом: из выходного буфера считывается строка - путь к набору файлов чисел в формате *.wav, который расположен на карте памяти SD/MMC, с математического стека снимается верхнее значение, которое должно находиться в пределах от -999,9 до +999,9 включительно, из входного текста выбирается очередное слово (введено после слова AUDIOSAY) и запускается процесс воспроизведения словами значения числа, а на стек данных кладется логическое значение - результат выполнения слова AUDIOSAY. При этом, логическое значение ИСТИНА означает, что процесс воспроизведения успешно начался, а НЕ ИСТИНА - что возникли аппаратные проблемы или число находится вне установленных пределов. Когда значение числа будет полностью воспроизведено, форт-система выполнит указанное после слова AUDIOSAY слово. Набор файлов чисел в формате *.wav описан в приложении.
По умалчиванию значение числа с вершины математического стека воспроизводиться с указыванием знака числа и с точностью 1 цифра после десятичной запятой. Воспроизведение знака числа можно отключить с помощью слова NOAUTOSAYPLUS и повторно включить с помощью слова AUTOSAYPLUS.
Для прекращения воспроизведения звукового сообщения через аудиосистему или числа с вершины математического стека существует слово AUDIOMUTE, которое немедленно прекращает воспроизведение звука через аудиосистему и блокирует выполнения заданного после слов AUDIOPLAY или AUDIOSAY слова.

Состояние сети GSM

Для большей гибкости использования GSM-модуля, в форт-системе доступна информация, которая касается состояния сети GSM, состояния связи, и т.п. На данный момент реализованы следующие слова для получения дополнительной информации о состоянии сети.
Слово ROAMING? возвращает на стек логическое значение ИСТИНА, если GSM-модуль зарегистрировался в роуминге, или логическое значение НЕ ИСТИНА если GSM-модуль зарегистрировался в домашней сети. Данное слово может пригодится в приграничных местностях для программного ограничения трафика в случае регистрации в роуминге.

Слово OPERATOR печатает в выходном буфере и на терминале название оператора сети GSM, в котором зарегистрирован GSM-модуль. Если модуль не зарегистрирован, то строка будет пустой.
Слово SIGNAL? возвращает на стек числовое значение уровня сигнала сети GSM. Качество сигнала при успешной регистрации в сети, может принимать значение от 0 до 4. Если возникли проблемы с регистрацией или самим GSM-модулем, то слово SIGNAL? возвращает на стек число -1. Данное слово может служить индикатором начала успешной работы в сети GSM. Информация о качестве сигнала в сети GSM возобновляется форт-системой каждые 3 сек. Приведем пример применения данных слов:

```
> ROAMING? .
0 (OK)
> OPERATOR
Mobile GSM (OK)
> SIGNAL? .
3 (OK)
```

Приложение

Настройка программы-терминала

В качестве терминала в среде Microsoft® Windows®XP, можно использовать программу эмуляции терминала Microsoft®HyperTerminal, которая входит в состав операционной системы.
Перед использованием программы эмуляции терминала, необходимо установить драйвер USB. Для этого необходимо запустить файл драйвера, который находится в папке "USB" дистрибутива, который поставляется в комплекте с контроллером или находится на сайте производителя. После успешной установки, в операционной системе появится новый последовательный порт (его номер можно проверить и поменять в программе "Диспетчер устройств"). После установки драйвера USB, с помощью кабеля из комплекта поставки, можно подсоединять контроллер к свободному порту USB.
Когда контроллер подсоединен к компьютеру, можно запустить программу Microsoft®HyperTerminal (Пуск→Программы→Стандартные→Связь→ HyperTerminal). При первом запуске, необходимо указать название нового подключения. Дальше в окне "Подключение" в списке "Подключаться через" выбрать номер последовательного порта, который был установлен драйвером USB. Дальше в окне "Свойства:COM" указать параметры связи: "Скорость" - 19200 для контроллеров с дисплеем и 57600 для контроллеров без дисплея; "Биты данных" - 8; "Четность" - нет; "Стоповые биты" - 1; "Управление потоком" - нет и подтвердить кнопкой "ОК". Дальше необходимо зайти в меню "Свойства" окна программы Microsoft®HyperTerminal (Файл→Свойства), выбрать закладку "Параметры" и в разделе "Эмуляция терминала" выбрать "TTY", дальше нажать кнопку "Параметры ASCII", снять все галочки и подтвердить все настройки кнопкой "ОК". После этого, следует сохранить выбранные параметры, выполнив команду "Сохранить" в окне программы Microsoft®HyperTerminal (Файл→Сохранить).
Если запускать программу Microsoft®HyperTerminal через имя подключения, которое

было сохранено при первом запуске, то все настройки автоматически вступают в силу и больше ничего настраивать не надо.

Создание звуковых файлов

Для реализации голосовых сообщений и меню, необходимо создать набор звуковых фрагментов и записать их в соответствующем формате на карту памяти SD/MMC. Для этого лучше всего подходит программа Microsoft®Звукозапись (Пуск→ Программы→Стандартные→Развлечения→Звукозапись) из состава операционной системы Microsoft® Windows®XP. Каждый записанный звуковой фрагмент необходимо сохранить в файл с расширением "wav" в формате PCM с атрибутами "8,000 кГц; 8 бит; Моно 7КБ/с". Для этого в окне программы Microsoft®Звукозапись выбрать меню "Сохранить как..." (Файл → Сохранить как..), в окне "Сохранение файла" в поле "Формат:" нажать кнопку "Изменить", дальше в окне "Выбор звука" в поле "Формат:" выбрать PCM, а в поле "Атрибуты:" выбрать "8,000 кГц; 8 бит; Моно 7КБ/с" и подтвердить все настройки кнопкой "ОК". Дальше указать название файла и сохранить его.
Сохраненные таким образом файлы, необходимо переписать в корневую директорию на карту памяти SD/MMC и вставить ее в контроллер. В дальнейшем, при выполнении встроенного слова PLAY, данные файлы будут использованы для воспроизведения звуковых фрагментов.

Набор файлов чисел

Для воспроизведения через аудиосистему словами чисел необходимо обеспечить на карте памяти SD/MMC набор *.wav файлов которые должны иметь формат аналогичный к описанному в предыдущем разделе и иметь фиксированные названия. В следующей таблице представлен перечень набора файлов.

Название файла	Голосовое содержимое файла	Название файла	Голосовое содержимое файла
_0.wav	ноль	_40.wav	сорок
_1.wav	один	_400.wav	четыреста
_10.wav	десять	_5.wav	пять
_100.wav	сто	_50.wav	пятьдесят
_11.wav	одинадцать	_500.wav	пятьсот
_12.wav	двенадцать	_6.wav	шесть
_13.wav	тринадцать	_60.wav	шестьдесят
_14.wav	четырнадцать	_600.wav	шестьсот
_15.wav	пятнадцать	_7.wav	сем
_16.wav	шестнадцать	_70.wav	семьдесят
_17.wav	семнадцать	_700.wav	семьсот
_18.wav	восемнадцать	_8.wav	восемь
_19.wav	девятнадцать	_80.wav	восемьдесят
_2.wav	два	_800.wav	восемьсот
_20.wav	двадцать	_9.wav	девять
_200.wav	двести	_90.wav	девяносто
_3.wav	три	_900.wav	девятьсот
_30.wav	тридцать	comma.wav	запятая
_300.wav	триста	minus.wav	минус

_4.wav	четыре	plus.wav	плюс

Данный набор файлов должен находиться в одной папке с любым названием на карте памяти SD/MMC.

Создание файлов-скриптов на языке ForthLogic™

Непосредственно вводить с терминала большие тексты неудобно, потому их сохраняют в текстовые файлы и переносят в форт-систему. Эта возможность также может быть полезной при тиражировании программ-скриптов и для резервного хранения созданного для конкретной задачи скрипта (это особенно актуально, когда осуществляется обновление программной прошивки контроллера - из словаря форт-системы удаляются все введенные пользователем слова).

Правила создания

Процедура создания файлов-скриптов чрезвычайно простая. В любом текстовом редакторе (например, Microsoft®Блокнот (Пуск→Программы→Стандартные→ Блокнот) из состава операционной системы Microsoft® Windows®XP) необходимо набрать текст скрипта и сохранить его в файл "forthdic.txt". При этом необходимо следить, чтобы длина строк не превышала 77 символов и в конце файла была, по крайней мере, одна пустая строка.

Правила интерпретации с карты памяти

Для интерпретации с карты памяти, файл скрипта "forthdic.txt" необходимо переписать в корневую директорию на карту памяти SD/MMC, вставить карту в контроллер и выполнить пункт конфигурационного меню "SD/MMC" → "ВЫПОЛНИТЬ:" → "ВЫПОЛН.СКРИПТ!". При этом данный файл будет построчно интерпретирован и выполнен текстовым интерпретатором форт-системы. При выявлении интерпретатором любой ошибки, его работа будет прекращена и на встроенном дисплее будет отображено сообщение о коде ошибки и номере строки файла "forthdic.txt", в котором возникла ошибка.
Если текст скрипта сохранить в корневой директории карты памяти SD/MMC в файле "autorun.txt", то при установке карты в контроллер, данный файл будет автоматически интерпретирован и выполнен текстовым интерпретатором форт-системы. При выявлении интерпретатором любой ошибки, его работа будет прекращена, потому важно, чтобы данный скрипт был предварительно отлажен в диалоговом режиме. После успешной интерпретации, данный файл будет автоматически удален с карты памяти.
Также существует возможность интерпретации произвольного файла, расположенного на карте памяти SD/MMC. Для этого, в терминальном режиме работы текстового интерпретатора форт-системы необходимо с новой строки без предварительных пробелов указать команду COMPILE FILE <название файла>. При выявлении интерпретатором любой ошибки, его работа будет прекращена и в окне терминала будет отображено сообщение об ошибке и номере строки файла, в которой возникла ошибка.

Правила интерпретации через Гипертерминал

Текстовый интерпретатор форт-системы может интерпретировать файлы не только с карты памяти SD/MMC. Если в терминальном режиме работы указать с новой строки без предварительных пробелов команду RECEIVE FILE, то появится приглашение на пересылку текстового файла по протоколу CRC Xmodem. При этом, в течение 30 сек. необходимо инициировать отправку файла. В программе Microsoft®HyperTerminal для этого необходимо зайти в окно "Отправка файла" (Передача → Отправить файл...), выбрать с помощью кнопки "Обзор" имя файла, указать "Протокол" - Xmodem и нажать кнопку "Отправить". При выявлении интерпретатором любой ошибки, его работа будет прекращена, окно отправления файла закроется и в окне терминала будет отображено сообщение об ошибке и номере строки файла, в которой возникла ошибка.

Принципы эффективной работы

При создании файлов-скриптов на языке ForthLogic™, следует четко понимать отличие таких файлов от обычных файлов программ в других традиционных языках программирования. Файл-скрипт может состоять из определений новых слов и констант, а также из других слов и чисел, которые непосредственно обрабатываются и выполняются в момент интерпретации данного файла. Этот мощный механизм позволяет автоматизировать практически любые операции (особенно с помощью карт памяти) и является главным отличием файлов-скриптов.

Создание программ

Написание программ - это по большей части интерактивный процесс: создание текста - проверка в реальных условиях - коррекция текста - проверка в реальных условиях - коррекция текста - .. Учитывая упомянутые особенности файлов-скриптов, можно порекомендовать следующий алгоритм при написании и отладке программ :

1. Создать текстовый файл программы.
2. В начале файла добавить следующие строки:

```
2 ERRORLEVEL
FORGET myprogram
0 ERRORLEVEL
: myprogram ". Короткое описание программы и ее версия " ;
```

3. Описать все необходимые слова и константы программы, при необходимости вставить слова, которые будут непосредственно выполнены во время интерпретации.
4. В конце файла описать "главное слово" программы (например, main) и после описания добавить строки:

```
". main " BOOT
3.0 1 TIMER! main
```

5. Добавить еще одну пустую строку и сохранить файл.

6. Отправить файл на интерпретацию через Гипертерминал.
7. Если в процессе интерпретации не оказалось синтаксических или других ошибок, то, приблизительно через 3 секунды после того как окно отправки файла закроется, программа будет запущена на выполнение и можно перейти к п. 10.
8. При выявлении интерпретатором любой ошибки, его работа будет прекращена, окно отправки файла закроется и в окне терминала будет отображено сообщение об ошибке и номере строки файла, в которой возникла ошибка.
9. После исправления ошибки необходимо повторить п.6.
10. После проверки работы программы, внести необходимые коррекции в файл и повторить п.6. Если вносить коррекции уже не нужно, то программа готова и файл является пригодным для передачи заказчику или долговременного хранения.

И наконец, традиционная при написании программ рекомендация: максимально используйте комментарии в тексте файлов-скриптов (с помощью скобок).

Отладка программ

Традиционные способы отладки программ заключаются в возможности остановки программы, просмотра и изменения контекста во время остановки, пошагового выполнения, и т.п. В силу ограничений аппаратной платформы и особенностей реализации многозадачности на языке ForthLogic™, не все классические средства и способы возможны для реализации. Однако, интерпретирующий характер языка позволяет определенную степень свободы и предоставляет возможности неосуществимые для других программно-аппаратных платформ (главным образом основанных на компиляторах). Например, отсутствие этапа компиляция ускоряет процесс создания и отладки отдельных программных процедур (слов в терминах форт-системы): задав слово в терминальном режиме, его можно сразу проверить "подставив" через стек необходимые для его работы параметры. Доступ к интерпретатору в терминальном режиме во время работы программы также позволяет определенным образом вмешиваться в ее работу: стеки и глобальные переменные доступны для модификации в произвольный момент времени. Однако, для дополнительного удобства, существуют слова, которые позволяют более эффективно осуществлять процесс отладки. Рассмотрим их.
Чтобы "увидеть" содержимое обоих стеков, не изменяя их состояние, используются слова .S (точка-стек) и .FS (точка-математический стек), которые соответственно печатают на терминале содержимое целочисленного и математического стека в квадратных скобках в том порядке, как это принято для стековой нотации - верхние значения на стеке (те, которые были прибавлены последними) находятся правее. Значение с математического стека печатаются в инженерном/научном представлении с 6 знаками после десятичной запятой. Данные слова можно использовать лишь в режиме интерпретации (их нельзя использовать в определении других слов).
Реализация многозадачности с помощью таймеров является простой и мощной концепцией. Иногда бывает необходимо остановить все задачи, которые привязаны к разным таймерам. Вручную останавливать каждый таймер долго и не эффективно. Это можно сделать с помощью слова STOPALL (остановить все). При

этом обнуляются промежутки времени и назначенные слова всех таймеров, останавливаются все настроенные обмены по протоколу MODBUS, останавливается воспроизведение сообщений и любая активность в канале GSM/GPRS/CSD. Стеки, глобальные переменные и состояния выходов остаются без изменений. Для обнуления стеков, глобальных переменных и выходных сигналов существует слово CLEARSYS (очистить систему). Более "глубокую очистку" можно выполнить с помощью перезапуска системы. Слово RESTART (перезапуск) выполняет перезагрузку всей системы и аналогично холодному старту (выключение-включение питания системы). Данные слова можно использовать лишь в режиме интерпретации.

Для реализации механизма точек остановки программы, существуют слова BREAK (остановка) и DEBUGLEVEL (уровень отладки). Слово BREAK предназначено для остановки форт-системы в процессе отладки и работает следующим образом: со стека снимается верхнее значение - номер точки остановки, и если этот номер меньший или равный глобальной системной переменной DEBUGLEVEL, то происходит остановка форт-системы. При этом останавливаются все таймеры, останавливаются все настроенные обмены по протоколу MODBUS, а на терминале в квадратных скобках печатается информация о номере точки остановки, содержимое всех стеков (целочисленного, математического и возвратов) и выходного буфера на момент остановки. После этого все стеки и буферы обнуляются, а глобальные переменные остаются без изменений.

Слово DEBUGLEVEL предназначено для установки одноименной системной переменной и работает следующим образом: со стека снимается верхнее значение и запоминается в глобальной системной переменной DEBUGLEVEL. Начальное значение этой переменной равняется нулю и она не есть энергонезависимая. Содержание данной переменной можно интерпретировать как "уровень отладки", которая активирует разные группы точек остановки. Например, если в программе существует несколько групп точек остановки, которые имеют номера от 100 до 200 и от 200 до 300, то присвоение системной переменной DEBUGLEVEL значения 200 блокирует вторую группу точек остановки (форт-система их игнорирует), а присвоение системной переменной DEBUGLEVEL значения 0 (значение по умолчанию) блокирует все точки остановки и программа работает в обычном режиме. Таким образом, слова BREAK могут оставаться в штатной программе постоянно при условии, что их номера больше чем ноль. Слово DEBUGLEVEL можно использовать лишь в режиме интерпретации.

В процессе создания программы возможны определенные типы ошибок, которые приводят к перезапуску всей системы. Если в системе настроен автоматический запуск главного слова (с помощью меню или слова BOOT), то при возникновении вышеупомянутых ошибок система будет непрерывно перезагружаться. Для избежания такого явления реализован механизм блокирования выполнения главного слова в терминальном режиме работы. То есть, при подключенном кабеле USB и работе через терминал автоматический запуск не происходит. В этом случае запустить программу можно вручную набрав ее главное слово в терминале.

Ошибки языка ForthLogic™

В терминальном и дистанционном режиме работы форт-системы все ошибки, которые возникают во время диалога с системой, по умолчанию отображаются в

круглых скобках в виде текста на английском языке. Во время интерпретации файла "forthdic.txt", в случае возникновения ошибки, на дисплее будет отображено сообщение о коде ошибки, а не сам текст ошибки. Дополнительно по умолчанию, в терминальном режиме работы в квадратных скобках отображаются все ошибки, которые возникают во время работы всей форт-системы (это касается и тех частей, которые выполняют слова назначенные таймерам и другим отложенным во времени процессам так и частям, которые интерпретируют слова в процессе диалога). При этом, в квадратных скобках отображается код ошибки и состояние стека возвратов на момент возникновения ошибки, при этом адреса со стека возвратов расположены в соответствии с принятой стековой нотацией (справа расположены самые последние выполненные адреса) и могут быть интерпретированы с помощью слова NAME. Данная информация позволяет реконструировать место возникновения ошибки. Отражение ошибок по умолчанию можно отключать и включать с помощью слова ERRORLEVEL (уровень отображения ошибок), которое снимает со стека числовой параметр уровня отображения ошибок и настраивает соответственно поведение форт-системы. Данный параметр не сохраняется во время пропадания питания и по умалчиванию равняется 0 - что означает отображение всех ошибок. Если установить данный параметр ровным 1, то отображаются только ошибки в квадратных скобках в терминальном режиме работы. Если установить данный параметр ровным 2, то форт-система вообще не отображает никаких ошибок - такой режим может пригодиться во время отладки программ.

Дальше представлены все известные форт-системе ошибки, их коды и объяснения:

Код	Текст ошибки	Объяснение
1	UNKNOWN WORD	Неизвестно форт-системе слово
2	ILLEGAL USAGE	Применение слова в неверном контексте (выполнение или компиляции)
3	ILLEGAL PARAMETER	Неверный параметр слова (вне разрешенных пределов)
4	INSUFFICIENT PARAMETERS	Недостаточное количество параметров
5	DATA STACK EMPTY	Стек данных пустой
6	DATA STACK FULL	Переполнение стека данных
7	RETURN STACK EMPTY	Стек возвратов пустой
8	RETURN STACK FULL	Переполнение стека возвратов
9	OUT OF MEMORY	Недостаточно памяти в словаре
10	MATHEMATIC STACK EMPTY	Математический стек пустой
11	MATHEMATIC STACK FULL	Переполнение математического стека
12	SD CARD NOT FOUND	Карта памяти SD/MMC отсутствует
13	FILE NOT FOUND	Файл не найден
14	INPUT BUFFER OVERFLOW	Переполнение входного буфера
15	FILE SYSTEM BUSY	Файловая система занята (например, воспроизводится звуковой файл)
16	FILE EMPTY	Файл пустой
17	WRONG CONSTRUCTION	Неполная конструкция управления выполнением (IF - ELSE - THEN)
18	FILE TRANSFER ERROR	Ошибка во время приема файла по протоколу Xmodem
19	FILE TRANSFER TIMOUT	Инициированный приему файла по протоколу Xmodem не начался в течение 30 сек.

Код	Текст ошибки	Объяснение
20	COMPILE MODE NOT ALLOWED	Запрещение режима компиляции (касается текстов SMS)

Аппаратная платформа языка ForthLogic™

Аппаратная платформа, на которой реализована конкретная форт-система с точки зрения программной модели, характеризуется разным количеством программно-доступных ресурсов и разным набором базовых слов.

Для всех аппаратных платформ общие следующие ресурсы и параметры. Все целочисленные арифметические и логические операции осуществляются над стеком данных глубиной 16 элементов. Диапазон представления целых чисел от -2147483648 до +2147483647. Все математические операции с числами с плавающей запятой осуществляются над математическим стеком глубиной 16 элементов. Диапазон представления чисел с плавающей запятой отвечает одинарной точности согласно стандарта IEEE-754: ±(1.4x10-45...3.4x1038). В логических операциях ИСТИНА означает произвольное число не ровное 0 (обычно -1), НЕ ИСТИНА означает 0. Операции, которые возвращают логическое значение ИСТИНА, возвращают его в виде -1. Информация в выходном буфере дублируется или отображением в окне терминала в терминальном режиме форт-системы или отображением в тексте SMS в дистанционном режиме форт-системы для аппаратных платформ, которые имеют модуль GSM. Глубина стека возвратов равняется 64 уровням.

В следующей таблице версий представленные ресурсы форт-системы, которые зависят от аппаратной платформы и версии прошивок, для которых актуально данное руководство по программированию:

Ресурс	Короткое описание	H01	H03	H02	H04	SG
FIRMWARE	Версия прошивки (микропрограммы)	1.16(B)	1.16(D)	1.16(GII)	4.50	1.08(SG)
D_MAX	Количество глобальных переменных для хранения целых чисел	64	64	64	128	256
F_MAX	Количество глобальных математических переменных для хранения чисел в формате с плавающей запятой	32	32	32	64	128
B_MAX	Количество глобальных однобитных переменных	128	128	128	128	256
S_MAX	Количество строчных переменных	-	4	4	8	16
T_MAX	Количество независимых таймеров	32	32	32	64	128
DI_MAX	Количество цифровых входов	8	8	8	8	6
AI_MAX	Количество аналоговых входов	4	4	4	4	2
DO_MAX	Количество цифровых выходов	4	4	4	4	6
RO_MAX	Количество релейных выходов	3	3	3	3	-
MENU_MAX	Количество пунктов меню пользователя	-	4	-	8	-
ROW_MAX	Количество строк текста дисплея	-	7	-	6	-
COL_MAX	Количество колонок текста дисплея	-	15	-	15	-
PHONE_MAX	Количество строчных переменных для хранения номеров телефонов	-	-	40	40	50
OUTBUF_MAX	Размер выходного буфера	120	120	180	180	180

Ресурс	Короткое описание	H01	H03	H02	H04	SG
LOGBUF_MAX	Размер буфера регистрации	-	-	512	256	1024

Встроенные слова языка ForthLogic™

Дальше представлены все известные форт-системе встроенные слова с коротким описанием и ссылкой на соответствующую страницу документа с детальным описанием:

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
1	входы/выходы	AI?	кладет на математический стек состояние аналогового входа	+	+	+	+	+
2	входы/выходы	AIS?	кладет на математический стек масштабируемое состояние аналогового входа				+	
3	входы/выходы	BAT?	кладет на математический стек напряжение питания аккумулятора	+	+	+	+	
4	входы/выходы	POW?	кладет на математический стек напряжение основного питания	+	+	+	+	
5	входы/выходы	DI?	кладет на стек состояние цифрового входа	+	+	+	+	+
6	входы/выходы	DO!	устанавливает состояние цифрового выхода равным значению со стека	+	+	+	+	+
7	входы/выходы	DO?	кладет на стек состояние цифрового выхода	+	+	+	+	+
8	входы/выходы	RO!	устанавливает состояние релейного выхода равным значению со стека	+	+	+	+	
9	входы/выходы	RO?	кладет на стек состояние релейного выхода	+	+	+	+	
10	GSM	OPERATOR	печать в выходном буфере оператора сети			+	+	+
11	GSM	ROAMING?	кладет на стек признак работы в роуминге			+	+	+
12	GSM	SIGNAL?	кладет на стек уровень сигнала сети			+	+	+
13	GSM	LAST	печатает в вых. буфере номер телефон последнего абонента			+	+	+
14	GSM	USER	печатает в вых. буфере номер телефона пользователя			+	+	+
15	GSM	USERPHONE	устанавливает номер телефона пользователя			+	+	+
16	GSM/текст	SMS	отправляет SMS			+	+	+
17	GSM/текст	USSD	отправление USSD запроса			+	+	+
18	GSM/голос	ANSWER	ответ на голосовой звонок				+	+
19	GSM/голос	CLIP	выполнение действия в ответ на голосовой звонок			+	+	+
20	GSM/голос	DIAL	осуществление голосового звонка				+	+
21	GSM/голос	HOLD	прекращение голосового звонка				+	+
22	GSM/голос	HOOK?	кладет на стек статус голосового звонка				+	+
23	GSM/голос	MUTE	прекращение воспроизведения звука				+	+
24	GSM/голос	PLAY	воспроизведение звукового файла с карты памяти SD/MMC				+	+
25	GSM/голос	SAY	воспроизведение значения с вершины математического стека				+	+
26	GSM/голос	MIC	подключение к каналу GSM внешней аудиосистемы и динамика				+	+
27	GSM/голос	VOICE	подключение к каналу GSM внутреннего синтезатора				+	+
28	GSM/голос	MICLEVEL	установка чувствительности внешней аудиосистемы				+	+
29	GSM/голос	SPKLEVEL	установка усиления внешнего динамика				+	+
30	GSM/голос	AUTOSAYPLUS	включает автоматическое воспроизведение знака плюс				+	+
31	GSM/голос	NOAUTOSAYPLUS	выключает автоматическое воспроизведение знака плюс				+	+
32	GSM/CSD	CONNECTCSD	установка канала данных CSD			+		+
33	GSM/CSD	SENDCSD	отправка строчных данных через канал CSD			+		+
34	GSM/CSD	BREAKCSD	разрыв канала данных CSD			+		+
35	GSM/CSD	STATUSCSD	кладет на стек состояние канала данных CSD			+		+
36	GSM/CSD	MODEMCSD	активирует прозрачный модемный режим			+		+

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
37	GSM/DTMF	TONE	генерация сигналов DTMF				+	+
38	GSM/DTMF	WAITKEY	ввод одиночных сигналов DTMF				+	+
39	GSM/DTMF	WAITPW	ввод пароля с помощью сигналов DTMF				+	+
40	GSM/DTMF	WAITSTR	ввод строки цифр с помощью сигналов DTMF				+	+
41	GSM/DTMF	DTMFCONFIRM	включает автоматическое подтверждение принятого сигнала DTMF				+	+
42	GSM/DTMF	NODTMFCONFIRM	выключает автоматическое подтверждение принятого сигнала DTMF				+	+
43	GSM/GPRS	APN	печатает в вых. буф. содержимое переменной APN			+		+
44	GSM/GPRS	SETAPN	устанавливает содержимое переменной APN			+		+
45	GSM/GPRS	IP	печатает в вых. буф. содержимое переменной IP			+		+
46	GSM/GPRS	SETIP	устанавливает содержимое переменной IP			+		+
47	GSM/GPRS	ID	печатает в вых. буф. содержимое переменной ID			+		+
48	GSM/GPRS	SETID	устанавливает содержимое переменной ID			+		+
49	GSM/GPRS	URL	печатает в вых. буф. содержимое переменной URL					+
50	GSM/GPRS	SETURL	устанавливает содержимое переменной URL					+
51	GSM/GPRS	ATTACHGPRS	подключение к сервису GPRS			+		+
52	GSM/GPRS	CONFIGTCP	настройка GPRS соединения			+		+
53	GSM/GPRS	CLIENT	системная константа для слова CONFIGTCP			+		+
54	GSM/GPRS	CLIENTDB	системная константа для слова CONFIGTCP					+
55	GSM/GPRS	CLIENTOPC	системная константа для слова CONFIGTCP			+		+
56	GSM/GPRS	SERVER	системная константа для слова CONFIGTCP			+		+
57	GSM/GPRS	CONFIGPORT	установка порта в режиме сервера			+		+
58	GSM/GPRS	DNSMODE	установка режима URL-адресов			+		+
59	GSM/GPRS	IPMODE	установка режима IP-адресов			+		+
60	GSM/GPRS	OPENTCP	установка соединения по протоколу TCP			+		+
61	GSM/GPRS	CLOSETCP	розрыв соединения по протоколу TCP			+		+
62	GSM/GPRS	SHUTTCP	отключение от GPRS			+		+
63	GSM/GPRS	SENDTCP	передача строковых параметров по протоколу TCP			+		+
64	GSM/GPRS	STATUSGPRS	возвращает на стек состояние сервиса GPRS			+		+
65	GSM/GPRS	STATUSSERVER	возвращает на стек состояние сервера			+		+
66	GSM/GPRS	STATUSTCP	возвращает на стек состояние протокола TCP			+		+
67	GSM/GPRS	STATUSOPC	возвращает на стек статус соединения с OPC-сервером			+		+
68	GSM/GPRS	REMOTEIP	печатает в вых. буф. IP-адрес клиента в режиме сервера			+		+
69	GSM/GPRS	LOCALIP	печатает в вых. буф. собственный IP-адрес			+		+
70	GSM/GPRS	M2MTCPON	включает режим без ответа форт-системы			+		+
71	GSM/GPRS	M2MTCPOFF	включает нормальный режим ответа форт-системы			+		+
72	меню	FOCUS	устанавливает фокус на пункт меню пользователя		+		+	
73	меню	HIDE	делает невидимым пункт меню пользователя		+		+	
74	меню	INFO	печатает текст в информационном поле меню пользователя				+	
75	меню	LASTMENU?	кладет на стек номер последнего задействованного пункта меню пользователя		+		+	
76	меню	MENU	настраивает и делает видимым пункт меню пользователя		+		+	
77	дисплей	BLACK	изменение цвета шрифта на черный		+		+	
78	дисплей	BLUE	изменение цвета шрифта на голубой		+		+	
79	дисплей	DEEPBLUE	изменение цвета шрифта на синий		+		+	
80	дисплей	GREEN	изменение цвета шрифта на зеленый		+		+	
81	дисплей	RED	изменение цвета шрифта на красный		+		+	
82	дисплей	ORANGE	изменение цвета шрифта на оранжевый		+		+	
83	дисплей	VIOLET	изменение цвета шрифта на фиолетовый		+		+	
84	дисплей	WHITE	изменение цвета шрифта на белый		+		+	

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
85	дисплей	YELLOW	изменение цвета шрифта на желтый		+		+	
86	дисплей	INVERT	инверсия цветов шрифта и фона		+		+	
87	дисплей	CLEAR	очистка дисплея		+		+	
88	дисплей	PRINT	печать строки из выходного буфера на дисплее		+		+	
89	дисплей	GET	отображение на дисплее окна для ввода числовых значений		+		+	
90	дисплей	GETS	отображение на дисплее окна для ввода строчных значений		+		+	
91	клавиши	BUTTON	настраивает функцию клавиши		+		+	
92	клавиши	F1	системная константа для слова BUTTON		+		+	
93	клавиши	F2	системная константа для слова BUTTON		+		+	
94	клавиши	DOWN	системная константа для слова BUTTON		+		+	
95	клавиши	LEFT	системная константа для слова BUTTON		+		+	
96	клавиши	OK	системная константа для слова BUTTON		+		+	
97	клавиши	RIGHT	системная константа для слова BUTTON		+		+	
98	клавиши	UP	системная константа для слова BUTTON		+		+	
99	сигнал	BEEP	генерация звукового сигнала	+	+	+	+	+
100	аудіо	AUDIOMUTE	прекращение воспроизведения звука через аудиосистему					+
101	аудіо	AUDIOPLAY	воспроизведение звукового файла с карты памяти SD/MMC через аудиосистему					+
102	аудіо	AUDIOSAY	воспроизведение значения с вершины математического стека через аудиосистему					+
103	сист. время	>DATE	превращает время UTC в общепринятую дату		+	+	+	+
104	сист. время	>TIME	превращает время UTC в общепринятое время		+	+	+	+
105	сист. время	>UTC	превращает общепринятое время во время UTC	+	+	+	+	+
106	сист. время	>WDAY	превращает время UTC в общепринятый день недели		+	+	+	+
107	сист. время	DATE?	кладет на стек значение годов, месяцев, дней	+	+	+	+	+
108	сист. время	TIME?	кладет на стек значение секунд, минут, часов	+	+	+	+	+
109	сист. время	UTC?	кладет на стек секунды UTC так, как это принято в ОС UNIX	+	+	+	+	+
110	сист. время	WDAY?	кладет на стек значение дней недели	+	+	+	+	+
111	сист. час	DST?	кладет на стек признак действия летнего времени			+		+
112	сист. час	TIMESTAMP	печатает в вых. буф. календарное представление времени			+		+
113	MODBUS	MODBUSSTART	формируется и активируется пакет протокола MODBUS	+	+	+	+	+
114	MODBUS	MODBUSSTOP	прекращается формирование пакета протокола MODBUS	+	+	+	+	+
115	MODBUS	CYCLIC_ACCESS	системная константа для слова MODBUSSTART	+	+	+	+	+
116	MODBUS	SINGLE_ACCESS	системная константа для слова MODBUSSTART	+	+	+	+	+
117	MODBUS	READ_COILS	системная константа для слова MODBUSSTART	+	+	+	+	+
118	MODBUS	READ_HOLDREGS	системная константа для слова MODBUSSTART	+	+	+	+	+
119	MODBUS	READ_INPUTREGS	системная константа для слова MODBUSSTART	+	+	+	+	+
120	MODBUS	READ_INPUTS	системная константа для слова MODBUSSTART	+	+	+	+	+
121	MODBUS	WRITE_COIL	системная константа для слова MODBUSSTART	+	+	+	+	+
122	MODBUS	WRITE_COILS	системная константа для слова MODBUSSTART	+	+	+	+	+
123	MODBUS	WRITE_REG	системная константа для слова MODBUSSTART	+	+	+	+	+
124	MODBUS	WRITE_REGS	системная константа для слова MODBUSSTART	+	+	+	+	+
125	MODBUS	MODBUSCALLBACK	настраивает автоматический обратной вызов протокола MODBUS	+	+	+	+	+
126	MODBUS	MODBUSPARAM	устанавливает параметры последовательного канала протокола MODBUS	+	+	+	+	+
127	MODBUS	EVEN	системная константа для слова MODBUSPARAM	+	+	+	+	+
128	MODBUS	NONE	системная константа для слова MODBUSPARAM	+	+	+	+	+
129	MODBUS	ODD	системная константа для слова MODBUSPARAM	+	+	+	+	+
130	MODBUS	MODBUSSTATUS?	состояние обмена по протоколу MODBUS	+	+	+	+	+
131	MODBUS	MODBUSTIMEOUT!	устанавливает время TIMEOUT протокола MODBUS	+	+	+	+	+
132	журнал.	LOGON	запускает и настраивает процесс регистрации	+	+	+	+	+

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
133	журнал.	LOGRUN	запускает процесс регистрации с существующими настройками	+	+	+	+	+
134	журнал.	LOGOFF	останавливает процесс регистрации	+	+	+	+	+
135	журнал.	ALL_DATA	системная константа для слова LOGON	+	+	+	+	+
136	журнал.	EVENTS_MODE	системная константа для слова LOGON	+	+	+	+	+
137	журнал.	USER_MODE	системная константа для слова LOGON	+	+	+	+	+
138	журнал.	INPUTS	системная константа для слова LOGON	+	+	+	+	+
139	журнал.	INTERVAL_MODE	системная константа для слова LOGON	+	+	+	+	+
140	журнал.	OUTPUTS	системная константа для слова LOGON	+	+	+	+	+
141	журнал.	TO_FLASH	системная константа для слова LOGON	+	+	+	+	+
142	журнал.	TO_SD	системная константа для слова LOGON	+	+	+	+	+
143	журнал.	TO_TCP	системная константа для слова LOGON			+		+
144	журнал.	LOG	переписывает выходной буфер в системный журнал	+	+	+	+	+
145	журнал.	LOG?	кладет на стек состояние процесса регистрации	+	+	+	+	+
146	журнал.	LOGSEND?	кладет на стек состояние процесса переноса данных	+	+	+	+	+
147	журнал.	LOGFILE?	кладет на стек статус системного журнала	+	+	+	+	+
148	журнал.	LOG>SD	переносит данные с журнала на карту памяти SD/MMC	+	+	+	+	+
149	журнал.	LOG>TCP	переносит данные с журнала через канал GPRS			+		+
150	реєстр.	LOG>DB	переносит данные с журнала через канал GPRS в базу MySQL					+
151	журнал.	FREELOG?	кладет на стек количество свободного места во внутренней памяти регистрации	+	+	+	+	+
152	журнал.	LOGFORMAT	форматирует внутреннюю память регистрации	+	+	+	+	+
153	журнал.	LOGTITLEOFF	выключает автоматическое оглавление во время регистрации	+	+	+	+	+
154	журнал.	LOGTITLEON	включает автоматическое оглавление во время регистрации	+	+	+	+	+
155	служеб.	:	начало определения нового слова	+	+	+	+	+
156	служеб.	;	окончание определения нового слова	+	+	+	+	+
157	служеб.	CONSTANT	определяет числовую константу	+	+	+	+	+
158	служеб.	TO	записывает вершину стека в числовую константу	+	+	+	+	+
159	служеб.	FCONSTANT	определяет числовую математическую константу	+	+	+	+	+
160	служеб.	TOF	записывает вершину математического стека в числовую математическую константу	+	+	+	+	+
161	служеб.	IF	слово условного оператора управления выполнением алгоритма	+	+	+	+	+
162	служеб.	ELSE	слово условного оператора управления выполнением алгоритма	+	+	+	+	+
163	служеб.	THEN	слово условного оператора управления выполнением алгоритма	+	+	+	+	+
164	служеб.	EXECUTE	выполнение адреса с вершины стека	+	+	+	+	+
165	служеб.	FIND	кладет на стек исполнительный адрес слова из выходного буфера	+	+	+	+	+
166	служеб.	NAME	отображение в выходном буфере слова, заданного адресом	+	+	+	+	+
167	служеб.	WORDS	отображение на экране в терминальном режиме словаря	+	+	+	+	+
168	служеб.	FORGET	удаление слова из словаря	+	+	+	+	+
169	служеб.	UNUSED	кладет на стек свободное место в словаре в байтах	+	+	+	+	+
170	служеб.	RESTART	перезапуск системы	+	+	+	+	+
171	служеб.	STOP	слово, которое ничего не выполняет (ну очень полезное!)	+	+	+	+	+
172	служеб.	CLEARSYS	очистка системы	+	+	+	+	+
173	служеб.	STOPALL	останавливает все таймеры	+	+	+	+	+
174	служеб.	TIMER!	запуск таймера, который после заданного времени выполнит заданное слово	+	+	+	+	+
175	служеб.	TIMER?	кладет на стек адрес слова, которое будет выполнено таймером	+	+	+	+	+

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
176	служеб.	AUTOSPACE	режим отображения в вых. буфере с автоматическим пробелом	+	+	+	+	+
177	служеб.	NOAUTOSPACE	режим отображения в вых. буфере без автоматического пробела	+	+	+	+	+
178	служеб.	FPREC!	установление точности отображения математического стека	+	+	+	+	+
179	служеб.	VERSION	отображение в выходном буфере версии прошивки	+	+	+	+	+
180	служеб.	(	начало комментария - конец или символ) или конец строки ввода	+	+	+	+	+
181	служеб.	.FS	неразрушающее отображение на терминале всего математического стека	+	+	+	+	+
182	служеб.	.S	неразрушающее отображение на терминале всего стека	+	+	+	+	+
183	служеб.	BREAK	точка остановки форт-системы	+	+	+	+	+
184	служеб.	DEBUGLEVEL	устанавливает уровень отладки для точек остановки	+	+	+	+	+
185	служеб.	ERRORLEVEL	устанавливает уровень отображения ошибок	+	+	+	+	+
186	конфиг.	BOOT	устанавливает скрипт автозапуска системы	+	+	+	+	+
187	конфиг.	BOOT.	печатает в выходном буфере скрипт автозапуска системы	+	+	+	+	+
188	конфиг.	CALBAT	калибровка аккумуляторной батареи	+		+		
189	конфиг.	CALI	калибровка комбинированного входа по току	+		+		+
190	конфиг.	CALPOW	калибровка напряжения питания	+		+		
191	конфиг.	CALV	калибровка комбинированного входа по напряжению	+		+		
192	конфиг.	CALZ	калибровка нуля аналогового входа					+
193	конфиг.	PASSWORD	устанавливает системный пароль	+		+		+
194	конфиг.	PROTECTPARAM	возвращает на стек параметры защиты системы			+		+
195	конфиг.	PASSWORD.	печатает в вых. буф. системный пароль			+		+
196	конфиг.	DISABLE	системная константа для слова PASSWORD	+		+		+
197	конфиг.	PROTECT_BY	системная константа для слова PASSWORD	+		+		+
198	конфиг.	CONTROL	устанавливает права доступа для дистанционного режима			+		+
199	конфиг.	CONTROLPARAM	возвращает на стек параметры доступа к дистанционному режиму			+		+
200	конфиг.	FOR_ALL	системная константа для слова CONTROL			+		+
201	конфиг.	FOR_LOYAL	системная константа для слова CONTROL			+		+
202	конфиг.	LOCAL	системная константа для слова CONTROL			+		+
203	конфиг.	REMOTE	системная константа для слова CONTROL			+		+
204	конфиг.	PIN	устанавливает пин-код			+		+
205	конфиг.	PIN.	печатает в вых. буф. пин-код			+		+
206	конфиг.	DIAI	устанавливает параметры комбинированного входа	+		+		
207	конфиг.	SET_TO_D	системная константа для слова DIAI	+		+		
208	конфиг.	SET_TO_I	системная константа для слова DIAI	+		+		
209	конфиг.	SET_TO_V	системная константа для слова DIAI	+		+		
210	конфиг.	DIAIPARAM	возвращает на стек параметры комбинированного входа	+		+		
211	конфиг.	SETWATCH	устанавливает параметры системных часов	+		+		+
212	конфиг.	SUMMER_OFF	системная константа для слова SETWATCH	+		+		+
213	конфиг.	SUMMER_ON	системная константа для слова SETWATCH	+		+		+
214	конфиг.	WATCH!	устанавливает системные часы	+	+	+	+	+
215	конфиг.	WATCHPARAM	возвращает на стек параметры системных часов			+		+
216	строки	."	отображение в выходном буфере строки, которая ограничена символом " или концом строки ввода	+	+	+	+	+
217	строки	FLUSH	очистка выходного буфера	+	+	+	+	+
218	строки	LENGTH	кладет на стек длину строки текста в выходном буфере		+	+	+	+
219	строки	NEWLINE	перенесение текста в выходном буфере на новую строку		+	+	+	+
220	строки	SPACE	отображение в выходном буфере пробела	+	+	+	+	+
221	строки	QUOTE	отображение в выходном буфере кавычек	+	+	+	+	+

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
222	строки	STRING!	записывает в строчную переменную текст из выходного буфера		+	+	+	+
223	строки	STRING?	кладет в выходной буфер строку текста из строчной переменной		+	+	+	+
224	мат.опер.	D>F	перенесение вершины стека данных на вершину математ. стека	+	+	+	+	+
225	мат.опер.	F -	вычитание на математическом стеке	+	+	+	+	+
226	мат.опер.	F*	умножение на математическом стеке	+	+	+	+	+
227	мат.опер.	F**	функция возведения в степень на математическом стеке	+	+	+	+	+
228	мат.опер.	F.	отображение в выходном буфере вершины мат. стека в формате с фиксированной запятой	+	+	+	+	+
229	мат.опер.	F/	деление на математическом стеке	+	+	+	+	+
230	мат.опер.	F+	сложение на математическом стеке	+	+	+	+	+
231	мат.опер.	F<	логическая операция МЕНЬШЕ на математическом стеке	+	+	+	+	+
232	мат.опер.	F>	логическая операция БОЛЬШЕ на математическом стеке	+	+	+	+	+
233	мат.опер.	F>D	перенос вершины мат. стека на вершину стека данных с округлением результата	+	+	+	+	+
234	мат.опер.	F>US	превращение "математический формат в без знаковый"	+	+	+	+	+
235	мат.опер.	FABS	получение абсолютного значения на математическом стеке	+	+	+	+	+
236	мат.опер.	FACOS	функция арккосинус на математическом стеке	+	+	+	+	+
237	мат.опер.	FASIN	функция арксинус на математическом стеке	+	+	+	+	+
238	мат.опер.	FATAN	функция арктангенс на математическом стеке	+	+	+	+	+
239	мат.опер.	FCOS	функция косинус угла на математическом стеке	+	+	+	+	+
240	мат.опер.	FCOSH	функция косинус гиперболический на математическом стеке	+	+	+	+	+
241	мат.опер.	FDEPTH	кладет на стек количество элементов математического стека	+	+	+	+	+
242	мат.опер.	FDROP	отбрасывание вершины математического стека	+	+	+	+	+
243	мат.опер.	FDUP	дублирование вершины математического стека	+	+	+	+	+
244	мат.опер.	FE.	отображение в выходном буфере вершины математического стека в научном формате	+	+	+	+	+
245	мат.опер.	FEXP	функция экспонента на математическом стеке	+	+	+	+	+
246	мат.опер.	FLN	функция логарифм натуральный на математическом стеке	+	+	+	+	+
247	мат.опер.	FLOG	функция логарифм десятичный на математическом стеке	+	+	+	+	+
248	мат.опер.	FNEGATE	изменение знака числа на математическом стеке	+	+	+	+	+
249	мат.опер.	FOVER	дублирование значения мат. стека которое лежит под вершиной на вершину мат. стека	+	+	+	+	+
250	мат.опер.	FPICK	дублирование произвольного значения математического стека на вершину математического стека	+	+	+	+	+
251	мат.опер.	FROLL	циклическое вращение произвольного количества верхних значений матем. стека	+	+	+	+	+
252	мат.опер.	FROT	циклическое вращение трех верхних значений математического стека	+	+	+	+	+
253	мат.опер.	FSIN	функция синус угла на математическом стеке	+	+	+	+	+
254	мат.опер.	FSINH	функция синус гиперболический на математическом стеке	+	+	+	+	+
255	мат.опер.	FSQRT	функция корень квадратный на математическом стеке	+	+	+	+	+
256	мат.опер.	FSWAP	перестановка двух верхних значений математического стека	+	+	+	+	+
257	мат.опер.	FTAN	функция тангенс угла на математическом стеке	+	+	+	+	+
258	мат.опер.	FTANH	функция тангенс гиперболический на математическом стеке	+	+	+	+	+
259	мат.опер.	FVAR!	записывает в математическую переменную значение с математического стека	+	+	+	+	+
260	мат.опер.	FVAR?	кладет на математический стек значение математической переменной	+	+	+	+	+
261	мат.опер.	US>F	превращение "без знаковый формат в математический"	+	+	+	+	+
262	целые опер.	-	операция вычитания	+	+	+	+	+
263	целые опер.	*	операция умножения	+	+	+	+	+

№	Условная группа	Слово	Короткое описание	H01	H03	H02	H04	SG
264	целые опер.	.	отображение в выходном буфере вершины стека	+	+	+	+	+
265	целые опер.	/	операция деления с отбрасыванием остатка	+	+	+	+	+
266	целые опер.	+	операция сложения	+	+	+	+	+
267	целые опер.	<	логическая операция МЕНЬШЕ	+	+	+	+	+
268	целые опер.	<=	логическая операция МЕНЬШЕ РАВНО	+	+	+	+	+
269	целые опер.	<>	логическая операция НЕ РАВНЯЕТСЯ	+	+	+	+	+
270	целые опер.	=	логическая операция РАВНЯЕТСЯ	+	+	+	+	+
271	целые опер.	>	логическая операция БОЛЬШЕ	+	+	+	+	+
272	целые опер.	>=	логическая операция БОЛЬШЕ РАВНО	+	+	+	+	+
273	целые опер.	ABS	операция получения абсолютного значения	+	+	+	+	+
274	целые опер.	AND	операция логического И	+	+	+	+	+
275	целые опер.	DEPTH	кладет на стек количество элементов стека данных	+	+	+	+	+
276	целые опер.	DROP	отбрасывание вершины стека	+	+	+	+	+
277	целые опер.	DUP	дублирование вершины стека	+	+	+	+	+
278	целые опер.	FALSE	кладет на стек логическое значение НЕ ИСТИНА	+	+	+	+	+
279	целые опер.	FLAG!	записывает в битовую переменную значение со стека	+	+	+	+	+
280	целые опер.	FLAG?	кладет на стек значение битовой переменной	+	+	+	+	+
281	целые опер.	ISNOW	попарно сравнивает шесть значений на вершине стека	+	+	+	+	+
282	целые опер.	LSHIFT	операция логического сдвига влево	+	+	+	+	+
283	целые опер.	MOD	операция деления с получением остатка	+	+	+	+	+
284	целые опер.	NEGATE	операция изменения знака числа	+	+	+	+	+
285	целые опер.	NOT	операция логического ОТРИЦАНИЯ	+	+	+	+	+
286	целые опер.	OR	операция логического ИЛИ	+	+	+	+	+
287	целые опер.	OVER	дублирование значения, которое лежит под вершиной стека на вершину стека	+	+	+	+	+
288	целые опер.	PICK	дублирование произвольного значения стека на вершину стека	+	+	+	+	+
289	целые опер.	ROLL	циклическое вращение произвольного количества верхних значений на стеке	+	+	+	+	+
290	целые опер.	ROT	циклическое вращение трех верхних значений на стеке	+	+	+	+	+
291	целые опер.	RSHIFT	операция логического сдвига вправо	+	+	+	+	+
292	целые опер.	SWAP	перестановка двух верхних значений стека	+	+	+	+	+
293	целые опер.	TRUE	кладет на стек логическое значение ИСТИНА	+	+	+	+	+
294	целые опер.	US>S	превращение "без знаковый формат в знаковый"	+	+	+	+	+
295	целые опер.	VAR!	записывает в переменную значение со стека	+	+	+	+	+
296	целые опер.	VAR?	кладет на стек значение переменной	+	+	+	+	+
297	целые опер.	XOR	операция логического ИСКЛЮЧИТЕЛЬНОГО ИЛИ	+	+	+	+	+

Внесенные изменения и дополнения

Версия документа 2.3

- добавлен раздел "Передача данных GPRS";
- добавлен раздел "Воспроизведение сообщений через акустическую систему";
- в приложении добавлен раздел "Набор файлов чисел";
- в разделе "Последовательный порт RS485" дополнено описание слова MODBUSPARAM;
- в раздел "Управление системой" добавлено описание слов WATCHPARAM, CALZ, CONTROLPARAM, PROTECTPARAM, PASSWORD. (пароль-точка), PIN. (пин-точка);
- в раздел "Системное время" добавлено описание слов DST?, TIMESTAMP;
- в разделе "Воспроизведение сообщений" дополнено описание слова SAY;
- в раздел " Воспроизведение сообщений" добавлено описание слов AUTOSAYPLUS, NOAUTOSAYPLUS;
- в раздел "Передача данных CSD" добавлено описание слова MODEMCSD;
- в раздел "Регистратор" добавлено описание слов TO_TCP и LOG>TCP;
- в таблице ресурсов и таблице слов прибавлен столбик версии контролера ES-ForthLogic-SG.

Версия документа 2.4

- в разделе "Последовательный порт RS485" дополнено описание слова MODBUSTIMEOUT!;
- в разделе "Передача данных GPRS" добавлено описание слов M2MTCPON, M2MTCPOFF, ATTACHGPRS и дополнено описание слов SETIP, SETAPN;
- в разделе "Регистратор" добавлено описание слов USER_MODE, LOGSEND?, LOGFILE? и дополнено описание слов LOG?, LOG;
- в разделе "Создание программ" изменено описание образца файла-скрипта;
- в разделе "Отладка программ" добавлено описание слова CLEARSYS;
- в разделе "Ошибки языка ForthLogic™" добавлено описание слова ERRORLEVEL;

Версия документа 2.5

- в разделе "Система" добавлено описание команды RESTORE DEFAULTS;
- в разделе "Константы, строки и переменные" добавлено описание слова QUOTE;
- в разделе "Регистратор" дополнено описание слова LOGFILE?;
- в разделе "Управления системой" дополнено описание слов SETWATCH и PASSWORD;
- в разделе "Управления системой" добавлено описание слова DIAIPARAM;

Версия документа 2.6

- в разделе "Регистратор" добавлено описание слова LOG>DB и дополнено описание процесса регистрации;
- раздел "Акустическая система" дополнено и переименовано на "Гарнитура";

- в разделе "Передача данных GPRS" добавлено описание слов ID, URL, SETID, SETURL, DNSMODE, IPMODE, CLIENTDB, CLIENTOPC, STATUSOPC