

***Настанова з програмування
на мові ForthLogic™***

2014

Зміст

Програмування на мові ForthLogic™	2
Вступ	2
Основні положення	2
Робота в діалоговому режимі	3
Стек даних та обчислення	4
Введення нових слів	9
Константи, рядки та змінні	12
Логічні операції	16
Структури управління	18
Таймери та багатозадачність	20
Векторне виконання	23
Програмування апаратних засобів	24
Входи	24
Виходи	28
Послідовний порт RS485	29
Система	37
Ідентифікація системи	37
Керування системою	37
Реєстратор	41
Підсистема живлення	46
Системний час	46
Інтерфейс користувача	49
Клавіатура	49
Дисплей	50
Виведення інформації	50
Введення інформації	51
Меню користувача	52
Звук	53
Індикація	53
Акустична система	53
Модуль GSM/GPRS	54
Стан мережі GSM	54
Номери телефонів	55
SMS та USSD	56
Голосові дзвінки	58
Відтворення повідомлень	59
Керування викликом	59
Здійснення виклику	59
Прийом виклику	60
DTMF сигнали	62
Гарнітура	65
Передача даних CSD	66
Передача даних GPRS	68
GPRS режим форт-системи	68
Засади клієнт-серверного з'єднання	68
Слова для роботи з GPRS	68
Модуль ETHERNET	75
Засади клієнт-серверного з'єднання ETHERNET	75
Слова для роботи з ETHERNET	75
Початкові налаштування	75
Протокол сервера MODBUS TCP	76
Протокол передачі даних на сервер баз даних MySQL	77
Додаток	78
Налаштування програми-терміналу	78
Створення звукових файлів	79
Набір файлів чисел	79
Створення файлів-скриптів на мові ForthLogic™	80
Правила створення	80
Правила інтерпретації з карти пам'яті	81
Правила інтерпретації через гіпертермінал	81
Засади ефективної роботи	82
Створення програм	82
Відлагодження програм	83
Помилки мови ForthLogic™	84
Апаратна платформа мови ForthLogic™	85
Вбудовані слова мови ForthLogic™	86
Внесені зміни та доповнення	91

Програмування на мові ForthLogic™

Вступ

Мова програмування Форт (англ. forth вперед і одночасне скорочення від fourth четвертий), яка лежить в основі мови ForthLogic™, з'явилася на початку 1970-х рр. в США. Її винахідник Чарльз Мур спочатку застосував її для розробки програмного забезпечення мікро-ЕОМ, яка керувала роботою радіотелескопу. Переваги роботи з мовою Форт були настільки великі, що незабаром її почали використовувати і на інших спеціалізованих ЕОМ.

Ефективність застосування мови Форт підтверджується тим, що вона використовується найбільш відомими грандами індустрії: корпорація Boeing використовує вбудований інтерпретатор мови Форт в системі авіоники (бортового обладнання) літака Boeing 777; корпорація Tektronix використовує мову Форт для серії аналізаторів мережевих протоколів K1297 та K1205; корпорація Lockheed Martin використовує мову Форт в бортовому обладнанні наземної телеметричної системи SMART для балістичної ракети Trident 2 D5; корпорація FedEx використовує ручний зчитувач штрихкодів SuperTracker з вбудованою Форт-системою в своєму програмно-апаратному комплексі електронного контролю посилок COSMOS II; корпорація General Electric використовує мову Форт для серії SONET-комутаторів JungleMUX; корпорація Euroray використовує мову Форт при створенні програмного забезпечення для універсальних касових терміналів / зчитувачів смарт-карт з архітектурою Open Terminal Architecture (OTA); корпорація Sun Microsystems з 1989 р. використовує загрузчик OpenBoot (програма типу BIOS) з вбудованим інтерпретатором мови Форт в своїх комп'ютерах SparcStation та серверах SPARCServer, а корпорація Apple Inc. використовує аналогічний загрузчик Open Firmware в своїх комп'ютерах Power Macintosh. Крім того, мова Форт є стандартною мовою керування обладнанням телескопів як на землі так і в космосі.

Основні положення

Синтаксис мови ForthLogic™ максимально простий. Запис кожної інструкції (команди) складається з одного слова, в якості якого може виступати послідовність будь-яких символів, що не містить пробілу. В мові ForthLogic™ такі інструкції (команди) так і називаються - *слова*. Простота синтаксису є наслідком того, що у якості обчислювальної моделі використовується стекова машина. В більшості випадків слова-команди цієї машини знімають необхідні операнди зі стеку і залишають свої результати (якщо вони є) також на стеку. Таким чином, програма, написана на мові ForthLogic™, виглядає як послідовність слів, кожне з яких має на увазі виконання тих або інших дій. Слова розділяються будь-яким числом пробілів; обмеження накладається тільки на довжину слова - воно повинне містити не більше 14 символів. Стандартно визначено порівняно невеликий набір "вбудованих" слів. Серед них є слова, що дозволяють визначати нові через вже існуючі і тим самим розширювати початковий набір слів-команд в потрібному для даного завдання напрямі.

Обчислювальна модель, яка лежить в основі мови ForthLogic™, складається зі стеку даних, стеку чисел з плаваючою комою, глобальних змінних, глобальних змінних в форматі чисел з плаваючою комою, глобальних бітових змінних, глобальних рядкових змінних, словника, і вихідного буфера для виведення результатів на термінал або обміну рядковими даними між словами. Мова ForthLogic™ дозволяє описувати паралельно виконувани процеси і функціонує в багатозадачному середовищі.

Стек даних (далі по тексту - або просто *стек* або *стек даних*) розташований в оперативній пам'яті і використовується для передачі числових параметрів та результатів між словами, виконуваними в межах однієї задачі. Його елементами є чотирибайтові значення, які розглядаються як цілі числа із

знаком в діапазоні від -2147483648 до +2147483647. В процесі виконання слів значення поміщаються на стек і знімаються з нього. Переповнення і вичерпання стеку перевіряється і повідомляється як помилка; його максимальний об'єм дорівнює 16 елементам.

Стек чисел з плаваючою комою (далі по тексту - *математичний стек*) також розташований в оперативній пам'яті і використовується для математичних обчислень над числами з плаваючою комою. Його елементами є чотирибайтові представлення чисел з плаваючою комою одинарної точності згідно стандарту IEEE-754, які можуть приймати значення в діапазоні $\pm(1,4 \times 10^{-45} \dots 3,4 \times 10^{38})$. В процесі виконання математичних операцій над числами з плаваючою комою, значення поміщаються на математичний стек і знімаються з нього. Переповнення і вичерпання математичного стеку перевіряється і повідомляється як помилка; його максимальний об'єм дорівнює 16 елементам.

Глобальні змінні є звичайними статичними змінними, які розташовані в оперативній пам'яті. Їх елементами є чотирибайтові значення, які розглядаються як цілі числа із знаком. Глобальні змінні звичайно використовуються для передачі числових параметрів і результатів цілочислових обчислень між словами виконуваними в різних задачах або для довготривалого збереження цілочислових параметрів і результатів в рамках однієї задачі. Кількість глобальних змінних залежить від апаратної платформи.

Глобальні змінні в форматі чисел з плаваючою комою (далі по тексту - *математичні змінні*) є статичними змінними, які розташовані в оперативній пам'яті. Їх елементами є чотирибайтові представлення чисел з плаваючою комою одинарної точності згідно стандарту IEEE-754. Математичні змінні звичайно використовуються для збереження проміжних математичних значень і результатів обчислень. Кількість глобальних математичних змінних залежить від апаратної платформи.

Глобальні бітові змінні є статичними змінними, які розташовані в оперативній пам'яті. Їх елементами є однібітові числа які приймають значення 0 або 1. Бітові змінні звичайно використовуються як різноманітні прапорці та в задачах логічного керування - вони природнім чином узгоджуються з унітарними сигналами контролера. Кількість глобальних бітових змінних залежить від апаратної платформи.

Глобальні рядкові змінні є статичними масивами довжиною 15 елементів кожний, які розташовані в оперативній пам'яті. Їх елементами є однібайтові значення, які представляють коди символів. Рядкові змінні призначені для оперативного зберігання рядків тексту довжиною до 15 символів, в яких можуть бути присутні пробіли. Кількість глобальних рядкових змінних залежить від апаратної платформи.

Початкову частину енергонезалежної пам'яті займає *словник* - сховище слів, констант і рядкових даних. По мірі розширення початкового набору слів, словник росте у бік збільшення адрес. Існують спеціальні слова, які дозволяють працювати зі словником (наприклад, з'ясовувати вільне місце у словнику, видаляти слова, тощо). Обсяг енергонезалежної пам'яті та розмір початкового словника залежить від апаратної платформи.

Вихідний буфер є звичайним статичним буфером, який розташований в оперативній пам'яті. Його елементами є однібайтові значення, які представляють коди символів. В буфері розташовуються рядкові дані для обміну з терміналом або між окремими словами. Розмір буферу залежить від апаратної платформи.

Робота в діалоговому режимі

Програмування на мові ForthLogic™ є діалоговим процесом. Працюючи за терміналом, користувач вводить слова-команди, а *форт-система*, тобто програмно-апаратна реалізація мови ForthLogic™, негайно виконує дії, що позначаються цими словами - це *термінальний режим* роботи форт-системи. Про свою готовність до обробки чергового рядка тексту форт-система повідомляє користувача запрошенням ">", яке друкується на терміналі. Отримавши таке запрошення, користувач набирає на

терміналі чергову порцію тексту, закінчуючи її клавішею "Enter". Отримавши сигнал про завершення введення, форт-система починає обробку введеного тексту (він розміщується у вхідному буфері для введення з терміналу), виділяючи в ньому слова-команди і виконуючи їх. Успішно обробивши весь введений текст, форт-система знову запрошує користувача до введення, і описаний цикл діалогу повторюється. Після успішного завершення обробки введеного тексту, форт-система виводить на термінал підтверджуюче повідомлення (OK) (від англійського o'kaу - "все гаразд"). Якщо під час обробки введеного тексту виявляється будь-яка помилка (наприклад, зустрівся невідоме форт-системі слово), то на термінал виводиться пояснююче повідомлення, обробка введеного тексту припиняється і форт-система запрошує користувача до введення нового тексту. Відомі форт-системі помилки та їх коди описані в додатку.

Максимальна довжина рядка тексту, яка приймається для обробки, складає 80 символів. При роботі з терміналом, вбудований редактор форт-системи автоматично обмежує довжину рядка до 77 символів і сповіщає про це звуковим сигналом. При наборі чергової порції тексту, крім клавіш літер та цифр, допускається застосовувати лише клавіші "Backspace", "Space"(пробіл) та "Enter".

Під словом термінал, ми розуміємо будь-яку з доступних програм, які здійснюють емуляцію терміналу. В середовищах різних операційних систем таких програм є дуже багато, причому як платних так і безплатних - процедура налаштування програми-терміналу в середовищі Microsoft® Windows®XP описана в додатку.

Безпосередньо вводити з терміналу великі тексти незручно, тому їх зберігають у текстові файли та за допомогою карти пам'яті SD/MMC переносять у форт-систему (процедура створення файлів та переносу їх на карту пам'яті SD/MMC описана в додатку). Також текстовий файл можна перенести у форт-систему за допомогою протоколу Xmodem безпосередньо з терміналу - дана процедура також описана в додатку.

Контролери без вбудованого дисплея мають спеціальний режим відображення статусної інформації про систему у вікні терміналу. Для входу та виходу з цього режиму існують відповідно команди SHOW STATUS та HIDE STATUS, які слід вводити з нового рядка терміналу без передуючих їм пробілів.

Стек даних та обчислення

Як уже згадувалося, в основі обчислювальної моделі мови ForthLogic™ лежить стекова машина. На даний момент, форт-система містить два програмно доступних стека: стек даних та математичний стек. Механізми роботи цих стеків подібні - різниця полягає в представленні даних та в наборі слів для роботи з даними у цих стеках.

Стек являє собою програмний об'єкт, який функціонує по принципу "останній зайшов - перший вийшов". Стек є єдиним місцем проведення математичних та логічних обчислень та є надзвичайно простим та надійним механізмом передачі числових параметрів та даних між окремими словами. В якості аналогії зі стеком можна привести колоду карт: покласти нове значення на стек можна порівняти з тим, як ми кладемо нову карту на верх колоди; зняти число зі стеку аналогічно до дії, коли ми знімаємо карту з колоди. Слова на мові ForthLogic™ зазвичай використовують в якості операндів верхні елементи стеків, прибираючи їх із стека і повертаючи результати (якщо вони є) на місце операндів. Як правило, слова використовують одне-два верхніх значення на стеку. Для їх опису застосовуватимемо наступну діаграму стекової нотації та кольорове виділення (це стосується обох стеків):

ім'я	вершина стеку до	---	вершина стеку після
слова	виконання слова		виконання слова

При цьому вважаємо, що саме верхнє значення в стеку (те яке було додане останнім) знаходиться справа. Додатково, перед значеннями, які знаходяться на математичному стеку, будемо вказувати тип стеку за допомогою літери *F* та двокрапки.

Для роботи з вершиною стека даних існують наступні 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 (від DUPLICATE дублювати) дублює вершину стека даних, додаючи в стек ще одне значення, рівне тому, яке було до цього верхнім. Слово DROP (скинути) прибирає верхнє значення. Слово OVER (через) дублює значення, яке розташоване на стеку даних безпосередньо під верхнім. Слово ROT (від ROTATE обертати) циклічно переставляє за годинниковою стрілкою три верхні значення в стеку даних. Нарешті, слово SWAP (обміняти) міняє місцями два верхні значення.

Можна працювати з будь-яким елементом стека даних за допомогою слів:

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

Слово PICK (взяти) дублює n-й елемент стека даних (рахуючи від нуля), так що 0 PICK тотожно DUP, а 1 PICK тотожно OVER. Слово ROLL (повернути) циклічно переставляє n верхніх елементів стека (теж рахуючи від нуля) за годинниковою стрілкою, так що 2 ROLL тотожно ROT, 1 ROLL тотожно SWAP, а 0 ROLL є порожньою операцією.

Щоб "побачити" верхнє значення на стеку даних, використовується слово . (крапка), яке знімає значення з вершини стека і друкує його у вихідному буфері та на терміналі як ціле число у вільному форматі (тобто без провідних нулів і зі знаком мінус, якщо число негативне). Якщо користувач хоче, щоб надруковане значення залишилося на стеку, він повинен виконати наступний текст:

```
> DUP .
```

Слово DUP створить копію верхнього значення, а крапка його роздрукує і прибере із стека.

Для роботи з вершиною математичного стека існують наступні 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 (від FLOAT DUPLICATE дублювати) дублює вершину математичного стека, додаючи в стек ще одне значення, рівне тому, яке було до цього верхнім. Слово FDROP (скинути) прибирає верхнє значення. Слово FOVER (через) дублює значення, яке розташоване на математичному стеку безпосередньо під верхнім. Слово FROT (від FLOAT ROTATE обертати) циклічно переставляє за годинниковою стрілкою три верхні значення в математичному стеку. Нарешті, слово FSWAP (обміняти)

мінняє місцями два верхні значення.

Також можна працювати з будь-яким елементом математичного стека за допомогою слів:

FPICK	$F: A_n, A_{n-1}, \dots, A_0, n$	--->	$F: A_n, A_{n-1}, \dots, A_0, A_n$
FROLL	$F: A_n, A_{n-1}, \dots, A_0, n$	--->	$F: A_{n-1}, \dots, A_0, A_n$

Слово FPICK (взяти) дублює n-й елемент математичного стека (рахуючи від нуля), так що 0 FPICK тотожно FDUP, а 1 FPICK тотожно FOVER. Слово FROLL (повернути) циклічно переставляє n верхніх елементів математичного стека (теж рахуючи від нуля) за годинниковою стрілкою, так що 2 FROLL тотожно FROT, 1 FROLL тотожно FSWAP, а 0 FROLL є порожньою операцією.

Щоб "побачити" верхнє значення на математичному стеку, використовуються слова F. (FLOAT - крапка) або FE. (FLOAT ENGINEER - крапка), які знімають значення з вершини математичного стека і друкують його у вихідному буфері та на терміналі. У першому випадку число друкується з фіксованою комою у вільному форматі з шістьма знаками після коми (тобто без провідних нулів і зі знаком мінус, якщо число негативне). У другому випадку число друкується в інженерному/науковому представленні з мантиєю, основою 10 та експонентою (наприклад -1,234E-02; 1,98E+12). Точність представлення чисел при роздруковуванні (тобто кількість цифр після десяткової коми) можна встановити за допомогою системної змінної FPREC, яка, по замовчуванню, рівна 6. Для запису нового значення в системну змінну FPREC існує слово FPREC!, яке працює наступним чином: з вершини стеку даних знімається число (нове значення змінної FPREC в діапазоні від 0 до 6) і присвоюється цій змінній. Точність рівна 0 означає відсутність десяткової коми взагалі, що тотожно перетворенню до цілого. Значення системної змінної FPREC не зберігається при виключенні живлення, тому, при використанні іншої, ніж по замовчуванню, точності представлення чисел при роздруковуванні, дану змінну слід ініціалізувати.

Як уже згадувалося, стек являє собою структуру даних, яка функціонує по принципу "останній зайшов - перший вийшов". Для побудови деяких алгоритмів обробки даних така структура може стати в нагоді. Для цього необхідно знати кількість елементів даних на стеку, щоб мати змогу виконати цикл обробки цих даних певну кількість разів. Для визначення кількості елементів на стеку даних та математичному стеку, існують слова DEPTH (глибина) та FDEPTH (FLOAT глибина), які кладуть на вершину *стеку даних* кількість елементів на стеку даних та на математичному стеку відповідно:

DEPTH	$A_n, A_{n-1}, \dots, A_1$	--->	$A_n, A_{n-1}, \dots, A_1, n$
FDEPTH	-	--->	n
	$F: A_n, A_{n-1}, \dots, A_1$	--->	$F: A_n, A_{n-1}, \dots, A_1$

Перераховані вище слова працюють зі значеннями, що вже знаходяться в стеку. А як занести значення в стек? Мова ForthLogic™ має наступне *правило по замовчуванню*: якщо введене слово форт-системи не відоме, то перш ніж повідомляти користувача про помилку, форт-система намагається зрозуміти це слово як запис числа. Якщо слово-число складається з одних цифр з можливим початковим знаком мінус, то помилки немає: слово вважається відомим і його дія полягає в тому, що дане число кладеться на вершину стека даних. Якщо слово-число складається з одних цифр розділених десятковою комою (насправді це літера . (крапка)) з можливим початковим знаком мінус та можливим науковим представленням експоненти числа (за допомогою літер е або Е та можливим знаком експоненти мінус), то слово теж вважається відомим і його дія полягає в тому, що дане число кладеться на вершину математичного стека.

Для безпосереднього переносу чисел з одного стеку в інший існують слова D>F та F>D. Слово D>F

знімає верхнє значення з вершини стеку даних та переносить на вершину математичного стеку з відповідним перетворенням представлення числа. Слово F>D знімає верхнє значення з вершини математичного стеку та переносить на вершину стеку даних з округленням до найближчого цілого числа та відповідним перетворенням представлення числа. У стековій нотації для даних слів необхідно відображати стан обох стеків:

D>F	A	---> -
	F: -	---> F:перетворене значення A
F>D	F:A	---> F: -
	-	---> округлене значення A

Тепер у нас достатньо засобів, щоб привести приклади діалогу. Розглянемо наступний протокол роботи (далі по тексту будемо виділяти протокол роботи з терміналом сірим кольором):

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

У відповідь на запрошення до введення (символ ">", друкований системою) користувач вводить три числа: 5, 6 і 7. Обробляючи введений текст, форт-система кладе ці числа у вказаному порядку на стек даних і після закінчення обробки виводить підтверджуюче повідомлення OK і знов запрошує користувача до введення. Далі користувач вводить текст з чотирьох слів: SWAP і три крапки. Виконуючи ці слова-команди, форт-система міняє місцями два верхні елементи стека даних (5, 6, 7 -> 5, 7, 6) і потім по черзі три рази знімає верхнє значення із стека даних та друкує його. В результаті на терміналі з'являється текст 6 7 5 і повідомлення OK, вказуючи на завершення обробки, після чого система знов видає користувачеві запрошення на введення. У відповідь на запрошення до введення користувач вводить два числа з фіксованою та плаваючою комою та слова для відображення двох значень з вершини математичного стеку в різному форматі. Обробляючи введений текст, форт-система кладе ці числа у вказаному порядку на математичний стек а потім знімає їх зі стеку друкуючи у вказаному форматі.

В попередньому прикладі, при виведенні значень зі стеку на термінал (і у вихідний буфер), автоматично друкується один пробіл. Така поведінка форт-системи встановлюється по замовчуванню, проте, можна довільним чином активувати чи деактивувати автоматичний друк одного пробілу після будь-якого виводу на термінал (і у вихідний буфер). Для цього існують слова AUTOSPACE - активація автоматичного друку одного пробілу, та NOAUTOSPACE - деактивація автоматичного друку одного пробілу.

Щоб збільшити кількість пробілів (для форматування виводу) можна застосувати слово SPACE (пробіл). Дане слово друкує у вихідному буфері та на терміналі один пробіл:

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

```
> AUTOSPACE 5 6 7
(OK)
> . . .
7 6 5 (OK)
>
```

Щоб перенести текст на новий рядок (також для форматування виводу) можна застосувати слово NEWLINE (новий рядок). Дане слово переносить текст у вихідному буфері та на терміналі на новий рядок - проте, практичне застосування цього слова має зміст при виведенні тексту на вбудований дисплей та для форматування текстів SMS (див. наступні розділи).

Слова-команди, що виконують цілочислові арифметичні операції над числами зі стеку даних, є загальноприйнятими математичними позначеннями:

+	A,B	---> сума A+B
-	A,B	---> різниця A-B
*	A,B	---> добуток A*B
/	A,B	---> частка від ділення A/B
MOD	A,B	---> залишок від ділення A/B
ABS	A	---> абсолютна величина A
NEGATE	A	---> значення із зворотним знаком -A

Слова-команди, що виконують математичні операції над числами у форматі з плаваючою комою з математичного стеку, відрізняються від загальноприйнятих долученням літери F:

F+	F:A,B	---> F:сума A+B
F-	F:A,B	---> F:різниця A-B
F*	F:A,B	---> F:добуток A*B
F/	F:A,B	---> F:ділення A/B
FABS	F:A	---> F:абсолютна величина A
FNEGATE	F:A	---> F:значення із зворотним знаком -A

Слова-команди, що виконують обчислення математичних функцій над числами у форматі з плаваючою комою з математичного стеку:

FSIN	F:A	---> F:синус кута A в радіанах $\sin(A)$
FCOS	F:A	---> F:косинус кута A в радіанах $\cos(A)$
FTAN	F:A	---> F:тангенс кута A в радіанах $\tan(A)$
FSINH	F:A	---> F:синус гіперболічний A $\sinh(A)$
FCOSH	F:A	---> F:косинус гіперболічний A $\cosh(A)$
FTANH	F:A	---> F:тангенс гіперболічний A $\tanh(A)$
FASIN	F:A	---> F:арксинус A $\arcsin(A)$, $-1.0 \leq A \leq 1.0$
FACOS	F:A	---> F:арккосинус A $\arccos(A)$, $-1.0 \leq A \leq 1.0$
FATAN	F:A	---> F:арктангенс A $\arctg(A)$
FLOG	F:A	---> F:логарифм десятковий A $\log_{10}(A)$
FLN	F:A	---> F:логарифм натуральний A $\ln(A)$
FEXP	F:A	---> F:експонента A e^A
F**	F:A,B	---> F:піднесення A до степені B A^B
FSQRT	F:A	---> F:корінь квадратний з A $\sqrt{A}$

Використання стека для зберігання проміжних значень природним чином приводить до так званої "зворотної польської форми" - одному із способів бездужкового запису арифметичних виразів, що має на увазі постановку знаку операції після операндів. Наприклад, вираз $(A/B+C)*(D*E-F*(G-H))$ записується таким чином: A B / C + D E * F G H - * - *. Таким чином, форт-систему можна використовувати як калькулятор цілочислових значень. Щоб обчислити, наприклад, значення $(25+18+32)*5$, досить ввести такий текст:

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

У відповідь система надрукує (виконуючи крапку) бажану відповідь. Також, форт-систему можна використовувати як калькулятор для чисел з плаваючою комою. Щоб обчислити значення $1.2e-2*\sin(25.23+0.18*3.1415)$, можна ввести текст:

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

Введення нових слів

Базова властивість мови ForthLogic™ - це можливість вводити нові слова, розширюючи тим самим набір команд в потрібному для користувача напрямі. Для введення нових слів найчастіше використовується визначення через двокрапку - визначення нового слова через вже відомі. Таке визначення починається словом : (двокрапка) і закінчується словом ; (крапка з комою). Відразу після двокрапки йде слово, що визначається, а за ним послідовність слів, через які воно визначається. Слово яке визначається має містити не більше 14 будь-яких символів та не містити пробілів. Наприклад, текст:

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

визначає слово S2, що обчислює суму квадратів двох чисел, що знімаються з вершини стека, стекова нотація має вигляд:

S2	A, B	---	A**2+B**2
----	------	-----	-----------

Після введення даного опису, слово S2 можна виконувати і включати в описи інших слів. При створенні таких визначень рекомендується ретельно відслідковувати всі зміни стека.

Перепишемо приведенне вище визначення слова S2 з коментарями, показуючи в дужках стан вершини стека після виконання кожного рядка. Слово ((ліва дужка) призначене для коментарів - форт-система ігнорує всі слова після лівої дужки або до кінця рядка або до першої закриваючої правої дужки, перед якою має бути принаймні один пробіл:

```
: S2      ( A, B ---> A**2+B**2 сума квадратів )
  DUP    ( A, B, B )
```

```

* SWAP      ( B**2,A )
DUP *      ( B**2,A**2 )
+ ;        ( A**2+B**2 )
(OK)
>

```

Розглянемо докладніше роботу форт-системи під час визначення нових слів. Ми вже знаємо, що отримавши від користувача чергову порцію вхідного тексту, форт-система виділяє в ній окремі слова і шукає їх в своєму словнику. Цю роботу виконує вбудований текстовий інтерпретатор форт-системи. Якщо слово в словнику не знайдене, то текстовий інтерпретатор намагається зрозуміти його як число, використовуючи описане вище правило по замовчуванню. Якщо слово знайдене або виявилось записом числа, то подальші дії інтерпретатора залежать від його поточного стану. У кожен момент часу текстовий інтерпретатор знаходиться в одному з двох станів: в стані *виконання* або в стані *компіляції*. В стані виконання знайдене слово виконується (тобто виконується дія, що становить його семантику), а число кладеться на стек. Якщо ж інтерпретатор знаходиться в стані компіляції, то знайдене слово не виконується, а компілюється, тобто включається в створювану послідовність дій для слова, що визначається в даний момент. Знайдене і скомпільоване таким чином слово буде виконано разом з іншими такими словами під час виконання визначеного через них слова. Якщо потрібно скомпільовати число, то текстовий інтерпретатор компілює особливий код, який під час виконання покладе значення даного числа на стек (ця дія абсолютно прозора для користувача). Скомпільовані слова виконуються надзвичайно швидко: різниця між часом інтерпретації певної послідовності дій та часом виконання тої самої скомпільованої послідовності дій сягає трьох порядків - це величезна перевага мови ForthLogic™ над іншими інтерпретуючими мовами.

Простежимо за роботою текстового інтерпретатора по обробці вже розглянутого визначення слова S2. Припустимо, що перед початком обробки введеного рядка інтерпретатор знаходиться в стані виконання. Першим словом є : (двокрапка), яке виконується. Його семантика полягає в тому, що з вхідного рядка вибирається чергове слово і запам'ятовується як слово, що визначається, а інтерпретатор перемикається в стан компіляції. Наступні слова, які інтерпретатор вибиратиме з вхідного рядка не враховуючи коментарі (DUP *, SWAP, тощо), компілюватимуться, а не виконуватися, оскільки інтерпретатор знаходиться в стані компіляції. В результаті, зі словом S2 зв'язується послідовність дій, що відповідає цим словам. Процес виділення і компіляції слів продовжуватиметься до тих пір, поки не зустрінеться ; (крапка з комою). Це слово особливе, воно має так звану *ознаку негайного виконання*. Слова з такою ознакою виконуються незалежно від поточного стану текстового інтерпретатора, тому крапка з комою буде другим виконаним словом після двокрапки. Семантика крапки з комою полягає в тому, що побудова визначення, початого двокрапкою, завершується і інтерпретатор знов перемикається в стан виконання. Тому після введення визначення слова S2, ми відразу можемо перевірити, як воно працює на конкретних значеннях:

```

> 5 4 S2 .
41 (OK)
>

```

Слова з ознакою негайного виконання виконуються по-різному залежно від стану текстового інтерпретатора. Деякі з них навіть використовуються тільки в одному з цих станів. Так, слово ; допустимо застосовувати тільки в стані компіляції. Воно завершує побудову нового визначення і перемикає текстовий інтерпретатор в стан виконання.

Введені слова можна виключити із словника за допомогою слова FORGET (забути), яке вибирає з вхідного рядка наступне слово (введене після слова FORGET) і виключає його із словника разом зі

всіма словами, визначеними пізніше. Такий спосіб виключення слів виглядає природнім при описуванні задачі методом "знизу-вверх", який використовується в мові ForthLogic™.

Розберемо наступний протокол діалогу:

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

Спочатку користувач обчислює результат від множення 2 на 2 і отримує відповідь 4. Ввівши по тому визначення слова 2 як число 3, він надалі отримує вже іншу відповідь. Виключивши це визначення слова 2 через FORGET, він повертається до колишньої семантики слова 2 - тобто просто цифри.

Для роботи з новими словами та словником, існують також слова WORDS (слова) та UNUSED (не використано). Слово WORDS друкує на терміналі список всіх доступних слів в словнику в порядку їх визначення (якщо список не поміщається на екрані, то робиться зупинка до моменту, коли користувач натисне клавішу "Space" (пробіл)). Слово UNUSED кладе на стек вільне місце в словнику в байтах. Далі, за допомогою слова . (крапка), це значення можна відобразити на терміналі. Також існує можливість повернення словника до первинного стану, коли в ньому присутні лише вбудовані слова. Для цього, в термінальному режимі роботи текстового інтерпретатора форт-системи необхідно з нового рядка вказати команду BUILD DICTIONARY без передуючих пробілів. Дана команда повністю перебудовує словник на основі внутрішньої інформації і застосовується в разі поважних збоїв у роботі вбудованих слів словника. Для простого витирання всіх слів користувача існує слово ERASE (витерти). Конкретний вміст словника залежить від версії програмної прошивки. Дізнатися про версію можна за допомогою слова VERSION (версія) яке друкує у вихідний буфер та на термінал рядок версії даної програмної прошивки.

При визначенні нових слів через двокрапку, іноді трапляється так, що не вистачає місця в межах одного рядка щоб закінчити побудову нового слова. У таких випадках, коли рядок перевищує встановлену довжину, допускається переходити на інший рядок за допомогою клавіші "Enter" і продовжувати визначення в новому рядку. Цей процес можна повторювати аж до закінчення визначення - тобто до слова ; (крапка з комою), при цьому форт-система не повинна виконувати жодних сторонніх завдань або програм.

У визначенні нових слів через двокрапку, можна застосовувати як вбудовані слова форт-системи так і інші попередньо визначені через двокрапку слова. Глибина вкладень визначень через двокрапку може сягати 64 рівнів і тісно пов'язана з програмно-недоступним *стеком повернень*. Стек повернень використовується для тимчасового зберігання адрес та деякої службової інформації. Під час виконання визначення будь-якого слова, на вершині стеку повернень буде знаходитись адреса того слова, з якого дане визначення було викликано (*адреса повернень*). Адреси повернень кладуться на стек повернень та знімаються з нього абсолютно прозоро для користувача, проте, переповнення та вичерпання стеку повернень перевіряється і повідомляється як помилка.

Загальна ідеологія використання власних слів при написанні програми на мові ForthLogic™ полягає в тому, що спочатку створюються самі прості "базові" слова які оперують найменшими порціями даних

або виконують найпростіші дії. Їх роботу звичайно можна негайно перевірити та відкоригувати в термінальному режимі шляхом імітації "робочого" оточення (підставивши на стек або у вихідний буфер необхідні дані, подавши на вхід необхідний сигнал, тощо). Переконавшись в коректності цих слів, на їх основі можна створювати і перевіряти більш складні слова, на їх основі ще більш складні і так далі. Цей процес в кінці зводиться до створення власного словника задачі і одного *головного слова*, яке визначає і запускає на виконання всю програму. Зрозуміло, що таких програм в словнику може бути декілька і кожна з них обов'язково має власне головне слово. Виконання того чи іншого головного слова може повністю змінити поведінку системи. Даний факт є унікальним в середовищі програмованих контролерів. Також важливим є згаданий вище спосіб написання програм на мові ForthLogic™ знизу-вверх, який, до того ж, є єдино можливим. Такі програми містять найменшу можливу кількість помилок.

Константи, рядки та змінні

Користувачу часто буває зручно працювати не з "анонімними" значеннями, а з поіменованими. По аналогії із засобами інших мов ці засоби мови ForthLogic™ називаються константами та змінними. Також існують рядки - об'єкти, які використовуються для виведення тексту у вихідний буфер та на термінал.

Константи, як випливає з їх назви, є поіменованим незмінними величинами. Фізично константи розташовуються у словнику серед інших слів, а оскільки словник знаходиться в енергонезалежній пам'яті то і константи також мають властивість зберігати своє значення при виключенні живлення. Проте існує можливість міняти значення констант, що дозволяє реалізувати довготривале енергонезалежне зберігання будь-яких числових параметрів конкретної задачі. Треба лише пам'ятати, що для контролерів зі старою апаратною платформою (повна версія, B, D, G версій 1.xx) а також серії M кількість змін однієї константи обмежене на рівні, приблизно, 10^5 разів і сам процес зміни є досить довготривалим (4-10 мсек). Для інших контролерів кількість змін однієї константи обмежене на рівні, приблизно, 10^{10} разів а процес запису спів розмірний по часу з процесом читання і може не прийматися до уваги. Константи можуть бути *звичайні* - це ті які були визначені користувачем і можуть бути переозначенні, та *системні* - це константи з базового набору слів, які спрощують виконання певних завдань і не можуть бути переозначенні.

Розглянемо слова для роботи з константами. Слово CONSTANT (константа) працює наступним чином. Зі стека даних знімається верхнє значення, а з вхідного тексту вибирається чергове слово (введене після слова CONSTANT) і запам'ятовується в словнику як нова команда - назва константи цілого типу. Її дія полягає в наступному: покласти на стек даних значення, зняте із стека в момент її визначення. Слово TO (в) працює наступним чином. Зі стека даних знімається верхнє значення, а з вхідного тексту вибирається чергове слово (введене після слова TO) - назва константи цілого типу. Далі здійснюється модифікація даної константи відповідно до введеного значення.

Слово FCONSTANT (FLOAT-константа) працює аналогічним чином - лише стосується математичного стеку. З математичного стека знімається верхнє значення, а з вхідного тексту вибирається чергове слово (введене після слова FCONSTANT) і запам'ятовується в словнику як нова команда - назва константи плаваючого типу. Її дія полягає в наступному: покласти на математичний стек значення, зняте із стека в момент її визначення. Слово TOF (в) працює наступним чином. З математичного стека даних знімається верхнє значення, а з вхідного тексту вибирається чергове слово (введене після слова TOF) - назва константи плаваючого типу. Далі здійснюється модифікація даної константи відповідно до введеного значення.

Назви констант мають містити не більше 14 будь-яких символів та не містити пробілів.

Розглянемо приклад:

```

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

```

Надалі, при виконання слова ДОБРЕ, на стек даних буде покладено число 5, а при виконанні слова PI - на математичний стек буде покладено число 3,14156.

Крім поодиноких констант існує можливість створювати та працювати з масивом констант цілого типу. Проголошення масиву констант здійснюється за допомогою слова ARRAY. Слово ARRAY (масив) працює наступним чином. Зі стека даних знімається верхнє значення - розмір проголошеного масиву, а з вхідного тексту вибирається чергове слово - ім'я масиву (введене після слова ARRAY) і запам'ятовується в словнику як нова команда. Її дія полягає в наступному: покласти на стек даних значення яке відповідає певному елементу масиву. Під час проголошення всі члени масиву приймають нульові значення. Існує обмеження на максимальний розмір проголошуваних масивів - 256. Нумерація елементів масиву починається від числа 1. Ім'я масиву має містити не більше 14 будь-яких символів та не містити пробілів.

Для доступу до елементів масиву використовується номер елемента, який знімається зі стека даних, та саме ім'я масиву. При цьому на стек повертається значення елемента масиву. Для зміни елемента масиву існує слово SET (встановити), яке працює наступним чином: зі стека даних знімається два верхніх значення - номер елемента та нове значення елемента, а з вхідного тексту вибирається чергове слово - ім'я масиву (введене після слова SET). Далі здійснюється модифікація даного елемента масиву.

Також для масивів запроваджена операція пошуку першого від початку елемента масиву рівного певному значенню - слово FINDINDEX. Воно працює наступним чином: зі стека даних знімається верхнє значення - величина яка шукається, а з вхідного тексту вибирається чергове слово - ім'я масиву (введене після слова FINDINDEX), далі здійснюється пошук і на стек повертається перший від початку номер елемента масиву рівного заданій величині. Якщо пошук не вдалий то на стек повертається нульове значення.

Змінні це поіменовані комірки оперативної пам'яті. Фізична природа цієї пам'яті дозволяє здійснювати модифікацію змінних нескінченну кількість разів, процес модифікації надзвичайно швидкий (100 нсек), проте, оперативна пам'ять не є енергонезалежною.

Користувачу доступні D_MAX глобальних змінних для зберігання цілих чисел імена яких - це номери в діапазоні від 1 до D_MAX. Кількість глобальних змінних D_MAX залежить від апаратної платформи на якій функціонує Форт-система і приведено в додатку. Для роботи з глобальними змінними існують слова VAR? та VAR!. Слово VAR? працює наступним чином. Зі стека даних знімається верхнє значення - номер змінної, а на стек даних кладеться копія вмісту відповідної змінної. Слово VAR! працює іншим чином. Зі стека даних знімається два верхніх значення - номер змінної та нове значення, яке запам'ятовується у відповідній змінній. Приведемо приклад роботи з цими словами:

```

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

```

```
-237889 (OK)
>
```

Користувачу також доступні F_MAX глобальних математичних змінних для зберігання чисел у форматі з плаваючою комою імена яких - це номери в діапазоні від 1 до F_MAX. Кількість глобальних математичних змінних F_MAX залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Для роботи з глобальними математичними змінними існують слова FVAR? та FVAR!. Слово FVAR? працює наступним чином. Зі стека даних знімається верхнє значення - номер математичної змінної, а на математичний стек кладеться копія вмісту відповідної змінної. Слово FVAR! працює іншим чином. Зі стека даних знімається верхнє значення - номер змінної, з математичного стека знімається верхнє значення та запам'ятовується у відповідній математичній змінній. Приведемо приклад роботи з даними словами:

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

Цілі та математичні змінні мають досить широкий діапазон представлення чисел і, відповідно, займають в пам'яті досить багато місця. Проте, для задач керування часто необхідно оперувати окремими бітами та прапорцями. Звичайно, для цього можна використовувати цілі числа та цілі змінні, але це є неефективно з точки зору затрат пам'яті та часу виконання. Для роботи з окремими бітами та прапорцями користувачу доступні B_MAX глобальних однобітових змінних імена яких - це номери в діапазоні від 1 до B_MAX. Кількість глобальних однобітових змінних B_MAX залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Для роботи з однобітовими змінними існують слова FLAG? та FLAG!. Слово FLAG? працює наступним чином. Зі стека даних знімається верхнє значення - номер змінної, а на стек даних кладеться копія вмісту відповідної змінної, яка приймає значення ІСТИНА або НЕ ІСТИНА (з даними визначеннями можна ознайомитися у параграфі "Логічні операції"). Слово FLAG! працює іншим чином. Зі стека даних знімається два верхніх значення - номер бітової змінної та нове значення, яке інтерпретується як ІСТИНА (для чисел не рівних нулю) або НЕ ІСТИНА (для чисел рівних нулю) та запам'ятовується у відповідній бітової змінній у вигляді 1 або 0. Приведемо приклад роботи з даними словами:

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

Рядки тексту допомагають користувачу при програмуванні специфічних команд, таких як здійснення та прийом голосових дзвінків, відтворення звукових файлів, виведення на дисплей, тощо. Ці команди використовують форматований вміст вихідного буфера. При роботі в термінальному режимі форт-системи, виведення у вихідний буфер одночасно дублюється виведенням на термінал. Розмір вихідного буфера складає OUTBUF_MAX. При переповненні вихідного буфера відбувається автоматичне відкидання даних які в нього не вміщаються і це не спричинює жодних помилок.

Для визначення рядка тексту, який буде друкуватися у вихідному буфері, використовується слово "." (крапка-лапки), після якого вводиться будь-який необхідний текст з можливими пробілами. В кінці тексту необхідно ввести пробіл та слово " (лапки), або можна просто закінчити текст клавішею "Enter". Визначення рядка тексту за допомогою слів "." (крапка-лапки) та " (лапки) не дозволяє застосовувати в тексті самі лапки якщо перед ними має бути пробіл. Якщо виникає необхідність друкувати у вихідному буфері текст з лапками, то для цього можна використати слово QUOTE (лапки). Наведемо приклад:

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

Таке визначення рядків тексту можна використовувати як всередині визначення через двокрапку, що має велике практичне значення, так і в діалоговому режимі. Рядок тексту, визначений всередині визначення через двокрапку, має ознаку константного рядка, не підлягає подальшій модифікації та є енергонезалежним. Наведемо приклади використання рядків:

```
> ." +380501234567 "
+380501234567 (OK)
> : Hi! ." Hello world! " ;
(OK)
> Hi!
> Hello world! (OK)
> ." Вас вітає система X
Вас вітає система X (OK)
>
```

Для визначення того що рядок тексту у вихідному буфері не є пустим, існує слово LENGTH (довжина), яке кладе на стек довжину рядку тексту (включно з пробілами), що знаходиться у вихідному буфері. На відміну від інших слів, які використовують вміст вихідного буфера і *очищають* його при цьому, слово LENGTH залишає вихідний буфер без змін. Для безпосередньої чистки вихідного буфера застосовується слово FLUSH (змивати).

Для оперативного зберігання рядків символів довжиною до 15 літер в оперативній пам'яті існують статичні рядкові змінні. Користувачу доступні S_MAX рядкових змінних імена яких - це номери в діапазоні від 1 до S_MAX. Кількість рядкових змінних S_MAX залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Для роботи з рядковими змінними існують слова STRING? та STRING!. Слово STRING? працює наступним чином. Зі стека даних знімається верхнє значення - номер змінної і у вихідний буфер друкується вміст даної змінної. Слово STRING! працює зворотнім чином. Зі стека даних знімається верхнє значення - номер змінної а з вихідного буфера зчитується рядок символів та запам'ятовується у відповідній змінній. Вміст вихідного буфера при цьому очищається. Приведемо приклад роботи з цими словами:

```
> ." +380501234567 " 1 STRING!
+380501234567 (OK)
> 1 STRING?
+380501234567 (OK)
>
```

Імена змінних у вигляді цілих чисел природнім чином дозволяють реалізувати *індексний доступ* до

комірок пам'яті різного типу. Зокрема, організація масивів є природнім розвитком такого доступу. Розглянемо наступний приклад: необхідно організувати масив з 10 цілих чисел з відповідним доступом до нього. Масив розташуємо починаючи з номера змінної записаної в константі `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 даному прикладі ми також використали опосередковану адресацію змінних, коли номер змінної всередині масиву визначався шляхом математичних обчислень. Номер змінної будь-якого типу також може зберігатися в цілочисловій змінній і це дає ще одну перевагу індексного доступу до оперативних даних.

Логічні операції

У мові ForthLogic™ число 0, в двійковому еквіваленті якого всі розряди нулі, представляє логічне значення НЕ ІСТИНА, а будь-яке інше 32-розрядне число сприймається як ІСТИНА. Разом з тим стандартні слова, які повинні повертати в якості результату логічне значення, зі всіх можливих представлень значення ІСТИНА використовують тільки одне: число -1, у двійковому еквіваленті якого всі розряди одиниці. Це пов'язано з тим, що логічні операції кон'юнкції, диз'юнкції і заперечення виконуються у мові ForthLogic™ *порозрядно* над всіма розрядами операндів і в цьому випадку відповідно трактуються як *порозрядні логічні операції*. Проте, їх можна трактувати і як традиційні логічні операції, але тільки над величинами ІСТИНА та НЕ ІСТИНА у згаданому вище представленні. Для зручності при здійсненні традиційних логічних операцій існують спеціальні системні константи TRUE (істина) та FALSE (не істина), які кладуть на стек логічні значення ІСТИНА та НЕ ІСТИНА відповідно:

TRUE	–	--->	-1
FALSE	–	--->	0

Слова-команди, що виконують логічні обчислення над значеннями стеку даних мають традиційний синтаксис та працюють або як звичайні логічні операції або як порозрядні логічні операції:

AND	A, B	--->	порозрядне логічне І $A \cdot B$
OR	A, B	--->	порозрядне логічне АБО $A + B$
XOR	A, B	--->	порозрядне логічне ВИКЛЮЧНЕ АБО $A \oplus B$
NOT	A	--->	порозрядне логічне ЗАПЕРЕЧЕННЯ $\neg A$

Логічні значення також виникають в операціях цілочислового порівняння, які входять в базовий набір слів і мають загальноприйняті позначення:

<	A, B	---	A < B	менше
=	A, B	---	A = B	рівно
>	A, B	---	A > B	більше
<=	A, B	---	A <= B	менше рівно
<>	A, B	---	A != B	не рівно
>=	A, B	---	A >= B	більше рівно

Всі ці операції порівняння знімають із стека даних два верхніх значення, порівнюють їх як числа зі знаком і повертають на стек даних результат порівняння у вигляді значень ІСТИНА або НЕ ІСТИНА в описаному вище представленні.

Логічні значення також виникають в операціях порівняння чисел з математичного стеку. Проте, ці операції знімають два верхніх значення з математичного стека, порівнюють їх як числа зі знаком і повертають результат порівняння на стек даних у вигляді значень ІСТИНА або НЕ ІСТИНА в описаному вище представленні:

F<	F: A, B	---	F: -	
	-	---	A < B	менше
F=	F: A, B	---	F: -	
	-	---	A = B	рівно
F>	F: A, B	---	F: -	
	-	---	A > B	більше
F<=	F: A, B	---	F: -	
	-	---	A <= B	менше рівно
F<>	F: A, B	---	F: -	
	-	---	A != B	не рівно
F>=	F: A, B	---	F: -	
	-	---	A >= B	більше рівно

Наведемо приклад застосування логічних обчислень:

```
> 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)
>
```

До логічних операцій над стеком даних також відносяться слова логічного зсуву вліво LSHIFT та вправо RSHIFT. Слово LSHIFT повертає на стек значення яке дістаємо з A зсувом його на B розрядів ліворуч і заповненням нулями звільнених праворуч розрядів. Аналогічно, слово RSHIFT повертає на

стек значення яке дістаємо з A зсувом його на B розрядів праворуч і заповненням нулями звільнених ліворуч розрядів:

LSHIFT	A,B	---	(A << B)	логічний зсув ліворуч
RSHIFT	A,B	---	(A >> B)	логічний зсув праворуч

Наведемо приклад застосування операцій зсуву:

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

Операції зсуву можна використовувати для обчислення бітових масок, які необхідні при встановленні окремих бітів у порозрядних логічних операціях. Наведемо приклад застосування операцій зсуву для обчислення бітових 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` обчислює бітову маску для відповідного номера біта. Слова `SetBit`, `ClearBit` та `InvertBit` здійснюють відповідну маніпуляцію над окремими бітами. Наприклад, `1 VAR? 10 SetBit 1 VAR!` встановлює 10 біт в 1 цілочисловій змінній.

Структури управління

У мові ForthLogic™ є лише один тип структур управління процесом виконання алгоритму - умовний оператор. Всі інші типи структур управління, такі як багатоваріантний вибір, умовні цикли, цикли з лічильником, тощо можна побудувати за допомогою умовного оператора та концепції багатозадачності

на основі таймерів (див. параграф "Таймери та багатозадачність"). Такий підхід гарантує (особливо для різноманітних безмежних циклів) розділення процесорного часу між окремими задачами та неможливість його монополізації однією задачею в умовах кооперативної багатозадачності в якій форт-система функціонує як одна із задач.

Умовний оператор будується за допомогою слів IF, ELSE та THEN. Ці слова використовуються лише всередині визначень через двокрапку і розділяють тіло визначення на відрізки. Відрізок тексту IF <частина-то> ELSE <частина-інакше> THEN задає наступну послідовність дій. Слово IF (якщо) знімає значення з вершини стека і розглядає його як логічне. Якщо це ІСТИНА (будь-яке ненульове значення), то виконується <частина-то> - слова, що знаходяться між IF та ELSE, а якщо НЕ ІСТИНА (рівно нулю), то виконується <частина-інакше> - слова між ELSE та THEN. Самі слова ELSE (інакше) і THEN (то) грають роль обмежувачів для слова IF і самостійного значення не мають.

Інша форма умовного оператора полягає в тому, що <частина-інакше> разом із словом ELSE може бути відсутньою, і тоді умовний оператор має скорочену форму IF <частина-то> THEN. Якщо логічне значення, що знімається зі стека словом IF, є ІСТИНА, то виконуються слова, які складають <частина-то>, а якщо НЕ ІСТИНА, то даний оператор не виконує ніяких дій. Зверніть увагу, що в обох випадках умова для слова IF на вершині стеку обчислюється попередніми словами.

Приведемо наприклад визначення корисного слова ?DUP, яке дублює верхнє значення стеку, якщо це не нуль, і залишає стек в попередньому стані, якщо на вершині нуль:

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

Специфікація даного слова може мати такий вигляд (коса риска розділяє два варіанти результату, який це слово може залишити на стеку):

?DUP	A	---	>	A, A (A не рівне нулю) / A
------	---	-----	---	----------------------------

Приведемо наприклад визначення іншого корисного слова LOGIC, яке перетворює верхнє значення стеку даних у логічне значення придатне для подальших логічних операцій:

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

Специфікація даного слова може мати такий вигляд:

LOGIC	A	---	>	-1 (A не рівне нулю) / 0
-------	---	-----	---	--------------------------

Розглянемо приклад побудови багатоваріантного вибору типу SWITCH-CASE на основі умовного оператора:

```

> : ТЕМПЕРАТУРА ( температура -- )
DUP 18 < IF ." Помірно " ELSE
DUP 21 < IF ." Нормально " ELSE
DUP 24 < IF ." Комфортно " ELSE
DUP 27 < IF ." Жарко " ELSE
DUP 40 < IF ." Гаряче " ELSE
      ." Пожежа! "
THEN THEN THEN THEN THEN DROP 1 STRING! ;
(OK)
>

```

В даному прикладі в залежності від значення на вершині стеку у рядкову змінну 1 записується відповідне слово. Зверніть увагу на однакову кількість слів DUP, IF, ELSE та THEN.

Перевага застосування умовного оператора для побудови багатоваріантного вибору полягає в тому, що порівняння та вибір не обмежене лише цілими числами - можна порівнювати математичні та логічні величини, можна навіть порівнювати різні типи даних в межах однієї такої складеної конструкції.

Таймери та багатозадачність

Як уже згадувалося раніше, мова ForthLogic™ дозволяє описувати паралельно виконувани процеси а сама форт-система функціонує в багатозадачному середовищі. Така її властивість тісно пов'язана з поняттям *таймер*. Таймер - це програмний об'єкт, що дозволяє реалізувати часовий інтервал, після закінчення якого може бути виконане будь яке відоме форт-системі слово.

Користувачу доступні T_MAX незалежних таймера, імена яких - це номери в діапазоні від 1 до T_MAX . Кількість таймерів T_MAX залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Для конфігурації таймерів призначене слово TIMER!. Слово TIMER! працює наступним чином: з математичного стеку знімається верхнє значення - часова затримка в секундах, зі стеку даних знімається верхнє значення - номер таймера, з вхідного тексту вибирається чергове слово (введене після слова TIMER!) і запускається таймер з відповідним номером. Зміст виконання слова TIMER! полягає в наступному: після закінчення інтервалу часу, який дорівнює значенню часової затримки, виконати слово, яке було вказане під час запуску таймера (введене після слова TIMER!).

Дискретність встановлення часу затримки спрацювання таймерів складає 0,01 сек. Внутрішня розрядна сітка представлення часу дозволяє реалізувати часові інтервали від 0 сек. до 21474836,47 сек. з кроком 0,01 сек. Тобто, цілком реально запустити таймер на 248 днів! Особливий випадок часу затримки спрацювання - це 0 сек (записується як 0.0). Затримка 0 сек означає виконання призначеного даному таймеру слова в наступному такті роботи всієї системи - фактичне негайне виконання. Справа в тому, що форт-система функціонує в умовах кооперативної багатозадачності в складі програмно-апаратної системи контролера, в якій існують інші програмні модулі та підсистеми. Всі вони (так само як і форт-система) виконуються по черзі у відведених для них тактах. Таким чином, коли виконується будь-яке слово на мові ForthLogic™, то воно виконується в межах такту роботи форт-системи і згаданий вище випадок з таймером, який має затримку 0 сек. означає, що в наступному такті всієї системи буде знову запущено форт-систему яка виконає призначене таймеру слово. Будь-яка інша часова затримка для таймера означає постановку форт-системи в чергу відкладених завдань і виконання лише через певний проміжок часу, який відповідає часовій затримці.

Наведемо приклад використання таймера:

```

> : встанов_вих 1 1 RO! ;
(OK)

```

```
> 4.0 1 TIMER! встанов_вих  
(OK)  
>
```

В даному прикладі, ми визначили нове слово встанов_вих, яке записує логічну одиницю у релейний вихід S1 (перемикає контакти реле - про це трохи пізніше). Далі ми запустили таймер 1 на час 4 секунди, після закінчення якого буде виконане слово встанов_вих (зверніть увагу на десяткову кому у представленні часу затримки). Отже, контакти реле перемкнуться через 4 секунди починаючи від моменту часу успішної обробки останнього рядка текстовим інтерпретатором форт-системи (після виведення повідомлення (OK)).

Слово TIMER! можна застосовувати і всередині означень через двокрапку, тоді з'являється можливість створювати послідовність дій, які виконуються через певні проміжки часу. Одна з форм такої послідовності - це створення дій, що циклічно виконуються нескінченну кількість разів - тобто, по суті, це механізм створення *паралельно виконуваних процесів*. Наведемо приклад:

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

В даному прикладі, ми визначили нове слово сирена з наступними змістом виконання: на стек кладеться логічний стан релейного виходу S1, здійснюється логічна інверсія вершини стеку, після чого значення зі стеку записується у релейний вихід S1, далі запускається таймер 1 на час 2 секунди, після закінчення якого буде виконане слово сирена - тобто весь процес виконання слова сирена знову повториться. Після визначення слова, ми виконали слово сирена, таким чином запустивши нескінченну кількість разів щойно описаний процес і отримавши неперервне перемикання контактів реле S1.

Як уже згадувалось, реалізовані в системі таймери є незалежними, отже є можливість запустити аж до T_MAX паралельних завдань, кожне з яких описується своїм словом. В таких багатозадачних системах необхідно якимось чином адмініструвати окремі задачі. Оскільки, механізм багатозадачності реалізовано за допомогою таймерів, то керуючи таймерами можна керувати задачами. Для цього призначене слово TIMER? та слово NAME (ім'я).

Слово TIMER? працює наступним чином: зі стека знімається верхнє значення - номер таймера а на стек кладеться або адреса слова, яке буде виконане відповідним таймером після заданого проміжку часу, або 0, якщо слово не задано (таймер не визначений).

Адреса слова - це адреса слова в словнику і для її інтерпретації призначено слово NAME, яке працює наступним чином: зі стека знімається верхнє значення та інтерпретується як адреса, далі здійснюється пошук слова, у визначенні якого міститься дана адреса (кожне визначене слово займає в словнику низку адрес) і, якщо слово знайдено, то воно друкується у вихідному буфері та на терміналі, інакше не друкується нічого. З врахуванням попереднього прикладу, наведемо протокол наступного діалогу:

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

Оскільки в кінці виконання слова сирена, таймер 1 запускається знову, то в будь-який момент часу, під

час виконання слова `TIMER?`, на стек буде покладено адресу слова `сирена` (показане значення адреси наведено для прикладу). Зверніть увагу, за допомогою слів `DUP . NAME` ми одночасно вивели на термінал адресу та назву слова.

В процесі роботи таймера (коли триває зворотній відлік часу) його можна перезапустити на виконання іншого слова через інший проміжок часу. При цьому попереднє слово та попередній проміжок часу анулюється. Зокрема, якщо перезапустити таймер із спеціальним словом `STOP` (зупинити), яке не робить жодної дії, можна фактично зупинити задачу, яка циклічно запускається даним таймером. З врахуванням попередніх прикладів, наведемо протокол наступного діалогу:

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

Переозначивши слово таймера `1` зі слова `сирена` на слово `STOP`, ми зупинили нескінченну дію слова `сирена`. Затримка `0` секунд, яку ми вказали в прикладі, в даному випадку означає негайне виконання слова `STOP` (в описаному вище розумінні), проте, може бути довільною, адже попереднє слово та попередній проміжок часу таймера `1` вже анульовані.

Як вже згадувалось, для побудови деяких типів структур управління можна використовувати умовний оператор та таймери. Зокрема, це стосується різноманітних умовних циклів та циклів з лічильником. Розглянемо приклад побудови на основі умовного оператора умовного циклу типу `BEGIN-UNTIL` який існує в мові Форт, проте відсутній в мові `ForthLogic™`. Формально, синтаксис побудови такого циклу може виглядати так:

`BEGIN <тіло-циклу> UNTIL`

Такий цикл називається умовним циклом з перевіркою в кінці. Після виконання слів, які утворюють його тіло, на стеку залишається логічне значення - умова завершення циклу. Слово `UNTIL` (доки не) знімає це значення та аналізує його. Якщо це `ІСТИНА`, то виконання циклу припиняється. Слово `UNTIL` можна замінити умовним оператором та таймером, який з прийнятною для даної задачі періодичністю буде запускати на виконання слова, які утворюють тіло циклу:

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

В даному прикладі умовного циклу слово `<тіло-циклу>` залишає на стеку логічне значення, яке аналізується в слові `BEGIN`, і якщо це `НЕ ІСТИНА`, то слово `BEGIN` запускається на виконання через `0,1` сек, інакше слово `BEGIN` не запускається і дія умовного циклу "припиняється". Таким чином, слово `UNTIL` було замінено низкою слів: `NOT IF 0.1 1 TIMER! BEGIN THEN`.

Розглянемо приклад побудови на основі умовного оператора іншого умовного циклу - циклу з лічильником типу `DO-LOOP` який також існує в мові Форт, проте відсутній в мові `ForthLogic™`. Формально, синтаксис побудови такого циклу може виглядати так:

DO <тіло-циклу> LOOP

Цикл починається зі слова DO (робити), яке знімає зі стеку два значення: початкове значення (на вершині стеку) та кінцеве значення та запам'ятовує їх. Поточне значення лічильника встановлюється рівним початковому значенню та виконується тіло циклу. Слово LOOP збільшує значення лічильника на 1 та перевіряє умову завершення циклу. Умовою завершення циклу є перевищення кінцевого значення на 1. Слово LOOP можна замінити операторами збільшення та перевірки лічильника та таймером, який з прийнятною для даної задачі періодичністю буде запускати на виконання слова, які утворюють цикл:

```
> : <тіло-циклу> 1 RO? NOT 1 RO! ; ( ---> )
(OK)
> : DO <тіло-циклу> 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 даному прикладі було замінено низкою слів: 1 VAR? 1 + DUP 1 VAR! 2 VAR? <= IF 0.1 1 TIMER! DO THEN. Змінна 1 VAR використовується як лічильник - вона, при необхідності, доступна також в тілі циклу, а змінна 2 VAR зберігає максимальне значення лічильника. Під час роботи лічильник збільшується на 1 та перевіряється умова завершення циклу. Перед викликом слова DO в змінні 1 VAR та 2 VAR необхідно записати мінімальне та максимальне значення лічильника.

Векторне виконання

При визначенні нових слів в мові ForthLogic™ дозволяється застосовувати лише вже відомі форт-системі слова. Проте, існує механізм *зворотнього посилання*, який фактично дозволяє реалізувати визначення слів після того як їх було попередньо застосовано в визначеннях інших слів. Цей механізм тісно пов'язаний з поняттям векторного виконання і ґрунтується на словах EXECUTE та FIND. Слово EXECUTE (виконати) працює наступним чином: зі стеку знімається верхнє значення - *виконавча адреса* будь-якого слова в словнику і дане слово негайно виконується так, ніби його виконали безпосередньо ввівши в форт-систему.

Увага: слід бути надзвичайно обережними з виконавчими адресами - якщо число на вершині стеку під час виконання слова EXECUTE не є такою адресою, то наслідки виконання слова EXECUTE будуть непередбачуваними!

Завдяки слову EXECUTE з'являється можливість передавати через стек виконавчі адреси однотипних слів і таким чином реалізовувати векторне (тобто, залежне від контексту) виконання програми.

Дізнатися виконавчу адресу будь-якого слова в словнику можна за допомогою слова FIND (знайти). Слово FIND працює наступним чином: з вихідного буфера зчитується рядок до першого пробілу - слово для якого необхідно знайти виконавчу адресу і, якщо слово відоме форт-системі, на стек кладеться виконавча адреса цього слова. У зворотньому випадку на стек кладеться значення НЕ ІСТИНА (або 0).

Наведемо приклад застосування згаданих слів для реалізації зворотнього посилання:

```
> 0 CONSTANT lbeep_addr
```

```

(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)
>

```

Спочатку ми створили константу `1beep_adr` для збереження виконавчої адреси (однаковою мірою ми могли би застосувати для цього цілочислову змінну). Далі ми створили слово-транслятор (`1beep`) яке просто виконує адресу `1beep_adr`. Це слово вже можна використовувати там де планувалося використовувати ще не визначене слово `1beep` - як, наприклад, у слові `2beep`. І нарешті, ми визначили саме слово `1beep`. Потім ми на постійно запам'ятали виконавчу адресу слова `1beep` в константі `1beep_adr`, тим самим "замикаючи" нескінченну послідовність виконання слів **`1beep-2beep-(1beep)-1beep ...`**

Програмування апаратних засобів

Для роботи з апаратними засобами платформи, на якій реалізовано форт-систему, у мові ForthLogic™ існує низка базових слів, які можна розділити по відповідним категоріям: входи, виходи, система, голосові дзвінки, SMS, тощо. Список слів для роботи з апаратними засобами може постійно розширюватись, відображаючи можливості конкретної апаратури та потреби кінцевих користувачів.

Входи

Значення з аналогових та цифрових входів перед тим як потрапити в систему фільтруються. Тип та властивості програмних аналогових фільтрів залежать від апаратної платформи. Цифрові входи фільтруються методом накопичення активного/пасивного стану сигналу з подальшим порівнянням з пороговими значеннями. Параметри даного алгоритму фільтрації дозволяють ефективно поглинати перехідні процеси тривалістю до 40 мс.

Звертання до входів здійснюється через їхні номери. Максимальне значення відповідного номера залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку.

Для роботи з цифровими входами існує слово `DI?`, яке знімає зі стеку верхнє значення - номер входу в діапазоні від 1 до `DI_MAX` і кладе на стек логічний стан відповідного цифрового входу (`DI/Alx`, `Dlx`, `Dx`). Логічний стан ІСТИНА означає замикання відповідного входу до будь-якого з контактів GND (спільний). У випадку комбінованих входів `DI/Alx`, якщо відповідний вхід за допомогою внутрішньої перемички та налаштувань системи сконфігуровано як аналоговий, то логічний стан цього входу завжди буде НЕ ІСТИНА.

Для роботи з комбінованими входами `DI/Alx`, які перемичками та налаштуваннями системи сконфігуровано як аналогові, існує слово `AI?`, яке знімає зі стеку верхнє значення - номер входу в діапазоні від 1 до `AI_MAX` і кладе на математичний стек стан відповідного комбінованого входу. При цьому, якщо в системі вхід визначено як цифровий, то значення відповідає абсолютному стану 10-розрядного АЦП, якщо вхід визначено як вхід по струму, то значення відповідає струму в міліамперах, і нарешті, якщо вхід визначено як вхід по напрузі, то значення відповідає напрузі у вольтях. Наведемо

приклад застосування згаданих слів:

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

Для зручності роботи з комбінованими входами DI/AIх в повній версії контролера в якій реалізовано конфігураційне меню та фіксований алгоритм роботи, існує слово AIS?, яке знімає зі стеку верхнє значення - номер входу в діапазоні від 1 до AI_MAX і кладе на математичний стек стан відповідного комбінованого входу. При цьому, якщо в конфігураційному меню вхід визначено як цифровий, то значення відповідає абсолютному стану 10-розрядного АЦП, якщо вхід визначено як вхід по струму або напрузі, то значення відповідає масштабованій фізичній величині. Масштабні коефіцієнти приймаються зі значень вказаних у відповідних пунктах конфігураційного меню. Таким чином слово AIS? можна використовувати для отримання реальних показників фізичних величин (наприклад температури) від датчиків з уніфікованим аналоговим вихідним сигналом.

Для версій контролерів з універсальними цифровими входами/виходами реалізована підтримка лічильників імпульсів. Дана функція працює зі всіма входами/виходами і налаштовується індивідуально для кожного входу/виходу. Якщо функція лічильника дозволена, то вміст лічильника імпульсів збільшується на 1 кожного разу при замиканні відповідного входу до будь-якого з контактів GND (спільний). Максимальне значення лічильника імпульсів відповідає цілому числу зі знаком +2147483647, у випадку його перевищення рахунок продовжується від числа -2147483648 в сторону збільшення до 0 і далі знову до +2147483647.

Для підтримки функції лічильника реалізовано три слова: COUNTERSTART налаштовує та активує лічильник, COUNTERSTOP зупиняє лічильник а COUNTERGET дозволяє отримати поточне значення лічильника. Слово COUNTERSTART працює наступним чином: зі стеку даних знімається верхнє значення - номер лічильника від 1 до DI_MAX, з математичного стеку даних знімається верхнє значення - часова затримка в секундах, з вхідного тексту вибирається чергове слово (введене після слова COUNTERSTART) і відбувається налаштування автоматичного зворотнього виклику цього слова у випадку якщо імпульси на даному лічильнику не з'являються на протязі часу більшого від часової затримки. Перед початком виконання слова автоматичного зворотнього виклику, на стек даних форт-системою буде покладено вміст даного лічильника імпульсів а сам лічильник обнулиться. Допускається вказувати часову затримку рівною 0.0 секунд, тоді механізм виконання слова автоматичного зворотнього виклику буде скасовано і лічильник буде неперервно рахувати вхідні імпульси. Слово COUNTERSTOP працює наступним чином: зі стеку даних знімається верхнє значення - номер лічильника від 1 до DI_MAX і відбувається зупинка роботи даного лічильника без його обнулення. Слово COUNTERGET працює наступним чином: зі стеку даних знімається верхнє значення - номер лічильника від 1 до DI_MAX, на стек даних кладеться вміст даного лічильника імпульсів а сам лічильник обнуляється. Роботу лічильників необхідно налаштовувати кожного разу при старті форт-системи.

Далі наводиться приклад налаштування роботи 1 лічильника на підрахунок імпульсів з контролера приймача купюр торгового автомата. При послідовному вкладенні купюр у приймач їх вартість у вигляді імпульсів накопичується у 1 цілочисловій змінній:

```
> : cb_billaccep 1 VAR? + 1 VAR! ;
```

```
(OK)
> 0.2 1 COUNTERSTART cb_billaccep
(OK)
>
```

Для версій контролерів з універсальними цифровими входами/виходами реалізована підтримка протоколу зчитувачів ідентифікаторів (карток) Wiegand-26, широко вживаного в системах контролю доступу. Існує можливість підключення до трьох зчитувачів карток безпосередньо до універсальних цифрових входів/виходів контролера. Зчитувачі можуть мати номери від 1 до 3, причому існує наступна відповідність сигналів зчитувача та входів/виходів контролера: сигнал "Data0" зчитувача N приєднується до $(N*2-1)$ входу/виходу контролера а сигнал "Data1" зчитувача N - до $(N*2)$ входу/виходу контролера.

Для підтримки протоколу реалізовано автоматичне ядро протоколу зчитувачів Wiegand-26 та два слова: WIEGANDSTART запускає ядро для одного із зчитувачів, а WIEGANDSTOP його зупиняє. Слово WIEGANDSTART працює наступним чином: зі стеку даних знімається верхнє значення - номер зчитувача в діапазоні від 1 до 3, з вхідного тексту вибирається чергове слово (введене після слова WIEGANDSTART) і відбувається налаштування автоматичного зворотнього виклику цього слова у випадку успішного отримання даних зі зчитувача. Перед початком виконання слова автоматичного зворотнього виклику, на стек даних форт-системою буде покладено номер карти, який був отриманий від зчитувача. Слово WIEGANDSTOP працює наступним чином: зі стеку даних знімається верхнє значення - номер зчитувача в діапазоні від 1 до 3 і відбувається зупинка роботи ядра протоколу для даного зчитувача. Ядро протоколу зчитувачів Wiegand-26 необхідно налаштовувати кожного разу при старті форт-системи.

Далі наводиться приклад програми простого контролера доступу з можливістю додавати та вилучати користувачів за допомогою двох "майстер" карт. В даному прикладі Data0 зчитувача приєднується до D1 контролера, Data1 зчитувача - до D2 контролера а контролер та зчитувач живляться від спільного джерела:

```
( array to hold user cards
100 ARRAY cards

6524593 CONSTANT mastercard+ ( add user master card
6524595 CONSTANT mastercard- ( delete user master card

( sounds
: 4b 0.15 1250 BEEP ;
: 3b 0.15 1000 BEEP 0.2 10 TIMER! 4b ;
: 2b 0.15 750 BEEP 0.2 10 TIMER! 3b ;
: correct 0.15 500 BEEP 0.2 10 TIMER! 2b ;
: wrong 0.5 100 BEEP ;

( temporary storages
: lastc! 2 VAR! ;
: lastc? 2 VAR? ;
: tempc! 1 VAR! ;
: tempc? 1 VAR? ;

( door utilities
```

```

: closedoor FALSE 3 DO! ;
: opendoor TRUE 3 DO! 3.0 12 TIMER! closedoor ;

( compare card with user array
: compcard
  DUP lastc! FINDINDEX cards
  IF
    correct opendoor
  ELSE
    wrong
  THEN
;

( add card to user array
: addcard
  0 FINDINDEX cards DUP
  IF
    SET cards
  ELSE
    DROP DROP
  THEN
;

( delete card from user array
: remcard
  FINDINDEX cards DUP
  IF
    0 SWAP SET cards
  ELSE
    DROP
  THEN
;

( wiegand callback
: cb
  tempc? mastercard+ =
  IF
    addcard 0 tempc!
  ELSE
    tempc? mastercard- =
    IF
      remcard 0 tempc!
    ELSE
      DUP DUP mastercard+ = SWAP mastercard- = OR
      IF
        tempc!
      ELSE
        compcard
      THEN

```

```

    THEN
  THEN
;

( sound task
: mastersound
  tempc? mastercard+ = IF 0.2 2400 BEEP THEN
  tempc? mastercard- = IF 0.2 400 BEEP THEN
  0.4 11 TIMER! mastersound
;

( main word
: main mastersound 1 WIEGANDSTART cb ;

." main " BOOT
main

```

Виходи

Звертання до виходів здійснюється через їхні номери. Максимальне значення відповідного номера залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку.

Для роботи з цифровими виходами типу "відкритий колектор" існує слово `DO?`, яке знімає зі стеку верхнє значення - номер виходу в діапазоні від 1 до `DO_MAX` і кладе на стек логічний стан відповідного цифрового виходу (`DOx`, `Dx`). Логічний стан `ІСТИНА` означає, що транзистор відповідного виходу знаходиться в провідному стані.

Щоб встановити цифровий вихід в потрібний стан, існує слово `DO!`, яке знімає зі стеку два верхніх значення - номер виходу в діапазоні від 1 до `DO_MAX` та стан виходу. Далі, розглядаючи цей стан як логічне значення, встановлюється відповідним чином логічний стан відповідного цифрового виходу (`DOx`, `Dx`). Логічний стан `ІСТИНА` означає, що транзистор відповідного виходу буде встановлено в провідний стан.

Для роботи з релейними виходами існує слово `RO?`, яке знімає зі стеку верхнє значення - номер виходу в діапазоні від 1 до `RO_MAX` і кладе на стек логічний стан відповідного релейного виходу `Sx`. Логічний стан `ІСТИНА` означає, що центральний контакт відповідного релейного виходу замкнутий на нормально розімкнутий контакт даного виходу.

Щоб встановити релейний вихід в потрібний стан, існує слово `RO!`, яке знімає зі стеку два верхніх значення - номер виходу в діапазоні від 1 до `RO_MAX` та стан виходу. Далі, розглядаючи цей стан як логічне значення, встановлюється відповідним чином логічний стан відповідного релейного виходу `Sx`. Логічний стан `ІСТИНА` означає, що центральний контакт відповідного релейного виходу буде замкнено на нормально розімкнутий контакт даного виходу.

```

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

```

Для версій контролерів з універсальними цифровими входами/виходами реалізована підтримка двох каналів генерування широтно-імпульсно модульованого сигналу (ШИМ - англ. PWM). Дана функція підтримується на апаратному рівні, тому її реалізація можлива лише для деяких універсальних цифрових входів/виходів, а саме: 1 канал ШИМ прив'язаний до входу/виходу D5, 2 канал ШИМ прив'язаний до входу/виходу D6. Робочий цикл сигналу, або коефіцієнт заповнення, відповідає розімкнутому стану вихідного транзистора універсального цифрового входу/виходу і є відношенням часу розімкнутого стану до періоду генерування ШИМ. Тобто при коефіцієнті заповнення 0,99 транзистор більшість часу практично закритий.

Для роботи з ШИМ призначено два слова PWMSTART та PWMSTOP. Слово PWMSTART працює наступним чином: з математичного стеку знімається верхнє значення - коефіцієнт заповнення сигналу ШИМ в діапазоні від 0,001 до 0,999, зі стеку даних знімається два верхніх значення - номер каналу ШИМ в діапазоні від 1 до 2 та частота сигналу ШИМ в діапазоні від 100 Гц до 10000 Гц і в заданому каналі розпочинається неперервна генерація сигналу ШИМ із заданою частотою та заповненням. Слово PWMSTOP працює наступним чином: зі стеку даних знімається верхнє значення - номер каналу ШИМ в діапазоні від 1 до 2 і в заданому каналі припиняється неперервна генерація сигналу ШИМ. Роботу генератора ШИМ необхідно налаштовувати кожного разу при старті форт-системи.

Наведемо приклад застосування слів PWMSTART та PWMSTOP для генерації сигналу ШИМ частотою 1000 Гц та заповненням 30% в першому каналі (D5):

```
> 0.30 1000 1 PWMSTART
(OK)
> 1 PWMSTOP
(OK)
>
```

Послідовний порт RS485

Обмін по послідовному порту RS485 відбувається згідно комунікаційного протоколу MODBUS RTU. Пристрої під'єднуються до порту RS485 паралельно і утворюють мережу. Обмін в мережі MODBUS RTU здійснюється між *головним* пристроєм (MASTER) і *підпорядкованими* пристроями (SLAVE). При цьому процес обміну може ініціювати лише головний пристрій типу MASTER, натомість підпорядковані пристрої типу SLAVE можуть лише відповідати на запити. Контролери серії ES-ForthLogic™ можуть працювати *лише* в режимі головного пристрою.

По замовчуванню, дані між головним і підпорядкованими пристроями пересилаються у вигляді байтів, упакованих в 11-бітові посилки на швидкості 9600 бітів/сек. Посилка починається із стартового біта (зі значенням 0), далі посилається байт з даними (8 бітів) а на кінці посилається два стоп біти (зі значенням 1). Окремі посилки складаються в *пакети повідомлень* з визначеною структурою.

Пристрій типу MASTER починає всі пакети даних з адреси пристрою типу SLAVE, до якого адресується пакет. Кожний пристрій типу SLAVE мусить мати унікальну адресу з діапазону від 1 до 247. В разі коли пристрій типу SLAVE висилає відповідь на запит, то в полі адреси він розміщує свою власну мережеву адресу. Це дає можливість пристрою типу MASTER визначити звідки надійшла відповідь.

Наступний байт пакету даних, висланого пристроєм типу MASTER, утворює *код доручення* (функції), яке має бути виконане пристроєм типу SLAVE. У відповідь SLAVE висилає пакет з таким же кодом доручення. Контролери серії ES-ForthLogic™ можуть формувати та обслуговувати коди наступних доручень:

- 01 (0x01) Read Coils (Читання статусу дискретних виходів)

- 02 (0x02) Read Discrete Inputs (Читання стану дискретних входів)
- 03 (0x03) Read Holding Registers (Читання статусу регістрів)
- 04 (0x04) Read Input Registers (Читання стану вхідних регістрів)
- 05 (0x05) Write Single Coil (Запис окремого дискретного виходу)
- 06 (0x06) Write Single Register (Запис окремого регістра)
- 15 (0x0F) Write Multiple Coils (Запис декількох дискретних виходів)
- 16 (0x10) Write Multiple registers (Запис декількох регістрів)

У подальших байтах пакету висилаються або приймаються дані. Кількість даних, що пересилаються, залежать від коду доручення.

Після відсилання/отримання всіх даних, до пакету приєднується два байти з контрольною сумою CRC. Призначенням контрольної суми CRC є виключення різноманітних помилок, які можуть виникнути під час пересилки даних, наприклад, як наслідок впливу сильних електромагнітних завад. Контрольну суму завжди підраховує пристрій, який висилає пакет даних. Далі пристрій, який приймає пакет, ще раз підраховує CRC з отриманих даних і порівнює її з прийнятим значенням. Якщо обидві суми збігаються, то пристрій приступає до обробки доручення яке міститься в пакеті, якщо ні - пакет ігнорується.

Пакети повідомлень з переліченими вище кодами доручень формуються та обробляються контролерами серії ES-ForthLogic™ автоматично відповідно до коду доручення. За це відповідає ядро протоколу MODBUS RTU. При цьому, дані які висилаються або приймаються, розташовуються у глобальних цілочислових та глобальних однобітових змінних контролера, що так само залежить від коду доручення. Пакети повідомлень можуть формуватися та висилатися в двох режимах: *циклічному* зі встановленим періодом, та *одиначному*. В циклічному режимі паралельно можуть формуватися до 4 різних пакетів повідомлень пронумерованих від 1 до 4 (не плутати цей номер з номерами функцій, адресами підпорядкованих пристроїв, тощо - це внутрішній для контролера логічний номер структури даних, яка описує всі аспекти обміну), причому це можуть бути повідомлення як для різних пристроїв так і для одного але з різними кодами доручень. Така кількість пакетів повідомлень, що можуть формуватися паралельно, ніяким чином не обмежує кількість пристроїв, що можуть бути адресовані в мережі. Застосувавши одиначний режим формування пакетів повідомлень можна кожного разу довільно обирати адресу підпорядкованого пристрою при формуванні пакету з тим самим номером. Для такого режиму формування пакетів дуже зручно застосовувати автоматичні виклики слів, які відбуваються в кінці одиначного обміну - так звані callback (зворотній виклик).

Для формування та активації пакетів повідомлень існує слово MODBUSSTART (почати обмін згідно протоколу MODBUS). Слово MODBUSSTART вживається разом із числовими параметрами та системними константами CYCLIC_ACCESS (циклічний режим обміну), SINGLE_ACCESS (одиначний режим обміну), READ_COILS (читання статусу дискретних виходів - функція 01), READ_INPUTS (читання стану дискретних входів - функція 02), READ_HOLDREGS (читання статусу регістрів - функція 03), READ_INPUTREGS (читання стану вхідних регістрів - функція 04), WRITE_COIL (запис окремого дискретного виходу - функція 05), WRITE_REG (запис окремого регістра - функція 06), WRITE_COILS (запис декількох дискретних виходів - функція 15) та WRITE_REGS (запис декількох регістрів - функція 16) і призначено для формування та активації пакетів повідомлень згідно комунікаційного протоколу MODBUS RTU. Оскільки системні константи просто кладуть на стек певні числа, то формально синтаксис застосування слова MODBUSSTART можна представити у вигляді діаграми стекової нотації проте, враховуючи велику кількість параметрів, більш наочно це можна зробити показавши послідовність числових параметрів та системних констант під час виклику даного слова (порядок слів має значення і не слід забувати про пробіли!):

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

- формується та активується пакет повідомлень в циклічному режимі;

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

- формується та активується пакет повідомлень в одиночному режимі;

Числові параметри в кутових дужках є обов'язковими:

- <period>** - період формування пакету в секундах, може приймати значення від 0,1 сек. до 600,0 сек. з кроком 0,01 сек;
- <slave>** - мережева адреса підпорядкованого пристрою, має бути унікальною в межах цілої мережі і може приймати значення від 0 до 247;
- <addr>** - адреса даних підпорядкованого пристрою, дана адреса в залежності від коду доручення може бути як адресою дискретного типу даних так і адресою регістрового типу даних і може приймати значення від 0 до 65535;
- <amount>** - кількість даних підпорядкованого пристрою, кількість в залежності від коду доручення може бути як кількістю даних дискретного типу так і кількістю даних регістрового типу і може приймати значення від 1 до 64;
- <index>** - номер глобальної цілочислової або однокітрової змінної, з якої розташовуються дані для запису або результати зчитування, кількість задіяних змінних дорівнює параметру <amount>;
- <func>** - код доручення, може приймати значення READ_COILS, READ_INPUTS, READ_HOLDREGS, READ_INPUTREGS, WRITE_COIL, WRITE_REG, WRITE_COILS, WRITE_REGS (див. коди доручень);
- <packet>** - номер пакета повідомлення, може приймати значення від 1 до 4.

Звичайно, замість конкретних значень числових параметрів можуть бути використані значення змінних або результати математичних операцій, треба лише правильно розташувати їх на стеку. В стековій нотації синтаксис слова MODBUSSTART має дві форми в залежності від параметра режиму доступу який представлено системними константами CYCLIC_ACCESS та SINGLE_ACCESS (цифри 1 та 2 відповідно):

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

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

Протоколом MODBUS визначено два типи даних:

- дискретний тип - однокіттова величина, яка може приймати значення 0 або 1 і в дорученнях використовується для опису дискретних входів та виходів (див. коди доручень);
- регістровий тип - 16-бітова величина, яка може приймати значення від 0 до 65535 і в дорученнях використовується для опису вхідних та внутрішніх (holding) регістрів.

В тих дорученнях, в яких згадуються регістрові дані, їх вміст відповідає вмісту цілочислових глобальних змінних контролера - це стосується доручень як запису так і читання. Тобто, для доручення запису регістрів, значення для запису копіюються із вмісту відповідних цілочислових глобальних

змінних, а для доручення читання регістрів прочитані значення записуються у відповідні цілочислові глобальні змінні. Відповідність між регістрами підпорядкованого пристрою та цілочисловими глобальними змінними контролера встановлюється за допомогою параметрів <addr>, <amount> та <index>. Те саме стосується і дискретного типу даних, тільки йому у відповідність ставляться одnobітові глобальні змінні контролера.

Для налаштування автоматичного виклику слів в одиночному режимі обміну існує слово MODBUSCALLBACK (зворотній виклик протоколу MODBUS). Слово MODBUSCALLBACK працює наступним чином: зі стеку даних знімається верхнє значення - номер пакета повідомлення, з вхідного тексту вибирається чергове слово (введене після слова MODBUSCALLBACK) і відбувається налаштування автоматичного зворотнього виклику цього слова в кінці одиночного обміну. Виконання даного слова є "одноразовим", тобто в кінці наступного одиночного обміну воно не буде виконано якщо не буде нового налаштування за допомогою слова MODBUSCALLBACK. Покажемо спосіб виклику даного слова:

<packet> MODBUSCALLBACK <word>

- налаштування автоматичного одноразового зворотнього виклику в одиночному режимі;

Параметри в кутових дужках є обов'язковим:

<packet> - номер пакета повідомлення, може приймати значення від 1 до 4.

<word> - відоме форт-системі слово, яке буде виконано в кінці одиночного обміну.

Для припинення формування пакетів повідомлень в циклічному режимі існує слово MODBUSSTOP (зупинити обмін згідно протоколу MODBUS). Слово MODBUSSTOP вживається разом із числовим параметром номер пакета і призначено для зупинки неперервного процесу формування цього пакету повідомлення в циклічному режимі. Покажемо спосіб виклику даного слова:

<packet> MODBUSSTOP

- припиняється формування пакету повідомлення в циклічному режимі;

Числовий параметр в кутових дужках є обов'язковим:

<packet> - номер пакета повідомлення, може приймати значення від 1 до 4.

В стековій нотації синтаксис слова MODBUSSTOP має дуже просту форму:

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

Для визначення стану процесу обміну існує слово MODBUSSTATUS? (стан обміну згідно протоколу MODBUS). Слово MODBUSSTATUS? знімає зі стеку номер пакета, який може приймати значення від 1 до 4, та повертає на стек код стану обміну для даного пакета повідомлення. В стековій нотації синтаксис слова MODBUSSTATUS? має форму:

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

Коди стану обміну поділяються на *коди помилок головного пристрою* які виникають в головному

пристрої (тобто контролері) та *коди помилок підпорядкованого пристрою*, які приходять у відповіді на пакет повідомлення. Дані коди та їх опис представлено в двох наступних таблицях:

Код	Опис кодів помилок головного пристрою
0	Немає жодних помилок обміну з підпорядкованим пристроєм
16	Підпорядкований пристрій не відповів на доручення на протязі заданого інтервалу часу
17	У відповіді підпорядкованого пристрою міститься помилка контрольної суми CRC
18	У відповіді підпорядкованого пристрою міститься помилка формату повідомлення
19	Внутрішня помилка головного пристрою

Код	Назва помилки протоколу MODBUS	Опис кодів помилок підпорядкованого пристрою
1	ILLEGAL FUNCTION	Прийнятий код доручення не може бути оброблений підпорядкованим пристроєм.
2	ILLEGAL DATA ADDRESS	Адреса даних вказана в пакеті повідомлення є не допустимою для даного підпорядкованого пристрою.
3	ILLEGAL DATA VALUE	Величина що міститься в полі даних пакета повідомлення є не допустимою величиною для даного підпорядкованого пристрою.
4	SLAVE DEVICE FAILURE	Невідновна помилка мала місце коли підпорядкований пристрій намагався виконати доручення.
5	ACKNOWLEDGE	Підпорядкований пристрій прийняв доручення і обробляє його, але це вимагає багато часу.
6	SLAVE DEVICE BUSY	Підпорядкований пристрій зайнятий обробкою доручення. Головний пристрій повинен повторити повідомлення пізніше, коли підпорядкований пристрій звільниться.

Якщо обмін для даного пакета відбувається в циклічному режимі, то код стану свідчить про можливі помилки обміну головного пристрою або, у випадку кодів помилок підпорядкованого пристрою, про те що обмін *припинився взагалі*. Якщо обмін для даного пакета відбувається в одиночному режимі, то код стану свідчить про можливі помилки після того як обмін завершився (при умові, що запит стану здійснено вже після обміну).

Код помилки 16 виникає коли підпорядкований пристрій не відповів на доручення на протязі заданого інтервалу часу TIMEOUT. Даний інтервал часу по замовчуванню дорівнює 1,0 сек. Проте за допомогою слова MODBUSTIMEOUT! (встановити час TIMEOUT протоколу MODBUS) цей час можна змінити. Слово MODBUSTIMEOUT! працює наступним чином: з вершини математичного стеку знімається число - значення часу в діапазоні від 0,1 сек. до 60,0 сек. з кроком 0,01 сек., і час TIMEOUT встановлюється рівним даному значенню. В стековій нотації синтаксис слова MODBUSTIMEOUT! має наступну форму:

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

В деяких випадках існує необхідність змінити параметри обміну по послідовному каналу з тих які прийняті по замовчуванню на інші (іноді навіть нестандартні). Для цього існує слово MODBUSPARAM (встановити параметри послідовного порту) яке вживається разом із числовим параметром та системними константами NONE (жодний), EVEN (парний), ODD (непарний) і призначено для зміни параметрів обміну. Покажемо спосіб виклику даного слова:

<parity> <stopbits> <baudrate> MODBUSPARAM

- встановлює параметри обміну;

Числові параметри в кутових дужках є обов'язковими:

- <baudrate>** - встановлює швидкість обміну, може приймати значення з ряду 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200
- <parity>** - значення біта парності, може приймати значення NONE (жодний), EVEN (парний) або ODD (непарний).
- <stopbits>** - кількість стоп бітів, може приймати значення 1 або 2. При цьому 1 стоп біт має сенс тільки при парності NONE. При інших типах парності даний параметр значення не має.

В стековій нотації синтаксис слова MODBUSPARAM має наступну форму:

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

При виникненні нештатних ситуацій під час роботи зі сторонніми пристроями згідно протоколу MODBUS RTU, може існувати потреба перезапустити ядро протоколу MODBUS RTU. Це можна здійснити словом MODBUSRESET (перезапуск MODBUS RTU).

При роботі з дорученнями, в яких згадується дискретний тип даних, ніякої проблеми з інтерпретацією даних не виникає: 0 дискретного типу відповідає значенню НЕ ІСТИНА однобітової глобальної змінної, а 1 дискретного типу відповідає значенню ІСТИНА однобітової глобальної змінної. При інтерпретації значень регістрового типу існує певна особливість, яка впливає з того, що регістровий тип є беззнаковим. Це означає, що при роботі з дорученнями, в яких згадується читання регістрового типу даних, які підпорядкований пристрій трактує як знакові, необхідно перетворювати отримані беззнакові числа у знакові для розрядної сітки цілих чисел контролера. Проте при роботі з дорученнями, в яких згадується запис регістрового типу даних, необхідність в перетворенні знакових чисел у беззнакові відпадає через те, що розрядна сітка цілих чисел контролера перекриває як знакові так і беззнакові числа регістрового типу даних.

Перетворення знакових чисел регістрового типу даних (представлених як беззнакові) у знакові числа цілого типу (прийнятого в контролері) можна здійснити за допомогою слова US>S (беззнаковий формат у знаковий). Синтаксис даного слова найкраще пояснює стекова нотація:

```
US>S      беззнакове_число ---> число_зі_знаком
```

Наведемо приклад:

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

Число 65535 попадає в розрядну сітку регістрового типу даних але в залежності від інтерпретації може означати або 65535 для беззнакових чисел, або -1 для знакових.

Особливість інтерпретації також виникає при обміні іншими ніж регістровий типами даних, наприклад

при обміні числами в форматі з плаваючою комою одинарної точності (4 байти). Перетворення чисел регістрового типу даних у числа математичного типу (прийнятого в контролері) можна здійснити за допомогою слова US>F (беззнаковий формат у математичний). Зворотнє перетворення можна здійснити за допомогою слова F>US (математичний формат у беззнаковий). Деякі сторонні пристрої підтримують числа математичного типу подвійної точності (8 байтів). Для їх перетворення можна скористатися допомогою слів US>DF (беззнаковий формат у математичний подвійної точності) та DF>US (математичний формат подвійної точності у беззнаковий). Синтаксис даних слів найкраще пояснює стекова нотація:

US>F	число1, число2	---	>	-
F:	-	---	>	F: число
US>DF	число1, число2, число3, число4	---	>	-
F:	-	---	>	F: число
F>US	-	---	>	число1, число2
F:	число	---	>	F: -
DF>US	-	---	>	число1, число2, число3, число4
F:	число	---	>	F: -

Наведемо приклад:

```
> 1.2345 F>US . .
1048 16286 (OK)
> 16286 1048 US>F F.
1.234500 (OK)
> -50.9305 DF>US . . . .
0 40960 30490 49225 (OK)
> 49225 30490 40960 0 US>DF F.
-50.930500 (OK)
>
```

Число 1,2345 в двійковому представленні згідно стандарту IEEE-754 одинарної точності має вигляд 0x3F9E0418, в десятковому еквіваленті регістрового типу даних це відповідає числам 16286 (0x3F9E) та 1048 (0x0418). Аналогічно перетворюються числа подвійної точності.

Деякі прилади потребують передачі по протоколу MODBUS RTU рядкових даних. Зокрема це стосується модулів ES-DU-1M власного виробництва. Для цього призначене слово STR>VAR (рядок у цілочисельні змінні). Слово STR>VAR працює наступним чином: зі стека даних знімається верхнє значення - номер змінної а з вихідного буфера зчитується рядок символів, далі символи починаючи з лівої сторони рядка попарно запам'ятовується у глобальних цілочисельних змінних починаючи з вказаного номера. Приведемо приклад роботи з цим словом:

```
> ." hello " 1 STR>VAR 1 VAR? . 2 VAR? . 3 VAR? .
hello 25960 27756 111 (OK)
>
```

Числа 25960, 27756, 111 в шістнадцятковому представленні мають вигляд 0x6568, 0x6c6c, 0x006f -

вони відповідають кодам символів слова "hello" і розташовані так як необхідно для передачі в модуль ES-DU-1M.

Тепер у нас достатньо засобів щоб запрограмувати роботу з підпорядкованим пристроєм по протоколу MODBUS (в даному випадку з термо-перетворювачем):

```
> : 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)  
>
```

Спочатку ми визначили слово pT, яке перетворює значення температури, яке знаходиться в 1 змінній, з представлення прийнятого в термо-перетворювачі у математичне число та друкує його на дисплеї (в термо-перетворювачі температура представлена у вигляді цілого числа зі знаком в діапазоні від -1900 до +1900, що відповідає -190,0°C... +190,0°C). Далі ми визначили слово AR594, що встановлює точність відображення чисел, запускає друк температури та налаштовує 1 пакет повідомлення на циклічне зчитування що 1,0 сек. даних регістрового типу з одного вхідного регістра розташованого за адресою 0 у підпорядкованому пристрої з логічною адресою 1 та запис цих даних у цілочислову змінну 1.

Наведемо також приклад обміну з трьома однаковими пристроями з використанням одиночного режиму обміну та автоматичних зворотніх викликів:

```
> 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)
>

```

Константи scan1_a, scan2_a, scan3_a та слова (scan1), (scan2), (scan3) слугують для реалізації зворотнього посилання на слова scan1, scan2, scan3. В самих же словах scan1, scan2, scan3 реалізовано друк стану попереднього обміну на дисплеї, налаштування наступного одноразового обміну та налаштування автоматичного зворотнього виклику, який, в свою чергу, має продовжити аналогічний процес для наступного обміну. Далі в прикладі іде зв'язування зворотніх посилань. В слові main відбувається налаштування параметрів обміну та запуск нескінченного ланцюжка обмінів з трьома пристроями. Зупинити обмін можна словом MODBUSSTOP:

```

> 1 MODBUSSTOP
(OK)
>

```

Система

Під терміном система, ми маємо на увазі загальносистемні апаратні та програмні ресурси. Для встановлення параметрів роботи системи звичайно використовується конфігураційне меню (там де воно присутнє) або утиліта конфігурації "FLConfig.exe", проте, практично всі параметри можна встановити програмно. На даний момент реалізовано наступні слова згруповані за певною ознакою, які призначені для отримання статусу або встановлення параметрів системи. Всі налаштування зберігаються у внутрішній енергонезалежній пам'яті. Для встановлення всіх параметрів у початковий стан (фабричні налаштування) необхідно в термінальному режимі роботи вказати з нового рядка без передуючих пробілів команду RESTORE DEFAULTS.

Ідентифікація системи

Для більш чіткої та однозначної ідентифікації системи крім слова VERSION (див. розділ "Введення нових слів"), існує слово SYSINFO. Дане слово друкує у вихідний буфер та на термінал розширену інформацію про версію системи, наявні серійні номери та тип ліцензії. Наявні серійні номери включають в себе або унікальний серійний номер контролера або IMEI для контролерів з вбудованим модулем GSM, а також MAC-адрес для контролерів з вбудованим ETHERNET. Тип ліцензії показує які програмні функції обмежені для даного контролера.

Керування системою

Для встановлення параметрів корекції та опцій системного годинника існує слово SETWATCH, яке вживається разом із системними константами SUMMER_ON (автоматичний перехід увімкнути) та SUMMER_OFF (автоматичний перехід вимкнути). Виклик даного слова може приймати наступні форми (порядок слів має значення!):

<correction> <tzone> SUMMER_ON SETWATCH

- вмикається автоматичний перехід на зимовий/літній час;

Числові параметри в кутових дужках є обов'язковими:

<correction> - ціле число - щомісячна автоматична корекція системного годинника в секундах;
<tzone> - ціле число - часовий пояс в годинах

<correction> <tzone> SUMMER_OFF SETWATCH

- вмикається автоматичний перехід на зимовий/літній час;

Числові параметри в кутових дужках є обов'язковими:

<correction> - ціле число - щомісячна автоматична корекція системного годинника в секундах.
<tzone> - ціле число - часовий пояс в годинах

Для отримання актуальних параметрів корекції та опцій системного годинника існує слово WATCHPARAM (параметри годинника), яке повертає на стек дані параметри та опції. В стековій нотації синтаксис даного слова має наступний вигляд:

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

де summer_flag дорівнює значенню ІСТИНА якщо ввімкнутий автоматичний перехід на зимовий/літній час.

Для встановлення системного годинника існує слово WATCH!, яке знімає зі стеку шість верхніх значень календарного часу та встановлює поточний час системного годинника:

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

- встановлюється поточний час системного годинника;

<sec> - ціле число - секунди в діапазоні 0-59.
<min> - ціле число - хвилини в діапазоні 0-59.
<hour> - ціле число - години в діапазоні 0-23.
<year> - ціле число - рік в діапазоні 1969-2038.
<mon> - ціле число - місяць в діапазоні 1-12.
<day> - ціле число - день місяця в діапазоні 1-31.

Стекова нотація слова WATCH! має наступний вигляд (корисно для автоматизованого встановлення часу з програми):

WATCH!	СЕК, ХВ, ГОД, РОКИ, МІС, ДНІ	---	>	-
--------	------------------------------	-----	---	---

Для встановлення типу комбінованих входів призначено слово DIAI, яке вживається разом із системними константами SET_TO_V (встановити по напрузі), SET_TO_I (встановити по струму) та SET_TO_D (встановити цифровим). Далі представлено порядок слів та констант при застосуванні слова DIAI (порядок слів має значення!):

<type> <input> DIAI

- тип комбінованого входу <input> встановлюється рівним <type>;

Числові параметри в куткових дужках є обов'язковими:

<type> - тип входу - може приймати значення SET_TO_V, SET_TO_I або SET_TO_D;

<input> - ціле число з діапазону від 1 до 4 – номер входу.

Для отримання актуального типу комбінованих входів існує слово DIAIPARAM (параметр входів), яке повертає на стек код типу входу. В стековій нотації синтаксис даного слова має наступний вигляд:

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

Для здійснення процедур калібрування реалізовано наступні слова: CALPOW та CALBAT - калібрування напруги живлення ПЛК; CALV - калібрування комбінованих аналогових входів в режимі роботи по напрузі; CALI - калібрування комбінованих аналогових входів в режимі роботи по струму; CALZ - калібрування нулів аналогових входів. Порядок проведення процедур калібрування описаний в окремій експлуатаційній документації та надається за вимогою.

Дистанційне керування системою за допомогою SMS або голосового меню передбачає здійснення налаштувань системних опцій, які дозволяють певним чином обмежити права користувачів. Для цього реалізовано низку слів та системних констант.

Слово CONTROL (керувати) вживається разом із системними константами LOCAL (локальний), REMOTE (віддалений) та FOR_LOYAL (для обраних), FOR_ALL (для всіх) і призначено для встановлення опції дистанційного керування. Виклик даного слова може приймати наступні форми (порядок слів має значення!):

FOR_ALL REMOTE CONTROL

- дистанційне керування можна здійснювати з любого номера мобільного телефону.

FOR_LOYAL REMOTE CONTROL

- керування можна здійснювати лише з перших шести номерів обраних користувачів, вказаних в конфігураційному меню або за допомогою слова USERPHONE (див. далі).

FOR_ALL LOCAL CONTROL

- дистанційне керування заборонено взагалі.

Для отримання актуальних опцій керування системою існує слово CONTROLPARAM (параметри керування), яке повертає на стек дані опції. В стековій нотації синтаксис даного слова має наступний вигляд:

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

де control_all_flag дорівнює значенню ІСТИНА якщо дистанційне керування можна здійснювати з любого номера; remote_on_flag дорівнює значенню ІСТИНА якщо дистанційне керування взагалі

дозволено.

Слово PASSWORD (пароль) вживається разом із системними константами PROTECT_BY (захистити за допомогою), DISABLE (відмінити) та рядком - паролем, визначеним за допомогою слова ." (крапка-лапки) і призначено для встановлення опції захисту паролем (формально пароль зчитується з вихідного буфера). Виклик даного слова може приймати наступні форми (порядок слів має значення!):

PROTECT_BY <pwd> PASSWORD

- вмикається захист доступу для дистанційного керування, пароль встановлюється рівним рядку <pwd>;

Рядковий параметр в кутових дужках є обов'язковим:

<pwd> рядок пароля у вихідному буфері визначений за допомогою слова ." (крапка-лапки).

PROTECT_BY PASSWORD

- вмикається захист доступу для дистанційного керування, пароль не змінюється.

DISABLE PASSWORD

- вимикається захист доступу для дистанційного керування.

Для отримання актуальних опцій захисту системи існує слово PROTECTPARAM (параметр захисту), яке повертає на стек дану опцію. В стековій нотації синтаксис даного слова має наступний вигляд:

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

де protect_on_flag дорівнює значенню ІСТИНА якщо увімкнено захист доступу для дистанційного керування. Інше слово PASSWORD. (пароль-крапка) друкує у вихідному буфері та на терміналі рядок актуального пароля.

Для автоматичного введення пін-коду задіяної SIM карти (при активованому пін-коді), існує слово PIN, яке працює наступним чином: з вихідного буфера зчитується рядок - пін-код і запам'ятовується у внутрішній енергонезалежній пам'яті. Даний пін-код автоматично вводиться кожного разу при старті системи. Для відображення пін-коду у вихідному буфері та на терміналі існує слово PIN. (пін-крапка).

Наведемо приклад застосування слів PIN та PIN.:

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

Для встановлення головного слова (або слів) керування системою на мові ForthLogic™ використовується слово BOOT (завантажити), яке зчитує з вихідного буфера рядок тексту довжиною не більше ніж 15 символів та запам'ятовує його в енергонезалежній пам'яті. Надалі, кожного разу при старті системи (при включенні живлення або при перезапуску системи) форт-система буде інтерпретувати та виконувати цей рядок (за умови встановлення в конфігураційному меню функції

системи - скрипт). Наведемо приклад:

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

В даному прикладі ми визначили слово `syrena`, яке неперервно перемикає контакти 1 релейного виходу, а потім сформували скрипт керування системою, який складається з наступних дій: переключити 2 релейний вихід, та запустити на виконання слово `syrena`.

Корисним також може бути слово `BOOT`. (завантажити-крапка), яке друкує у вихідному буфері та на терміналі скрипт керування системою. З урахуванням попереднього прикладу:

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

Для налаштування контролерів, які містять вбудований порт `ETHERNET`, існує низка слів для встановлення параметрів доступу до локальної мережі `ETHERNET`.

Слова `ETHIP`, `ETHGATEWAY` та `ETHMASK` встановлюють відповідно власну IP-адресу, IP-адресу основного шлюзу та маску підмережі. Всі ці слова працюють наступним чином: з вихідного буфера зчитується рядок - IP-адреса і запам'ятовується у відповідній комірці внутрішньої енергонезалежної пам'яті. Для відображення власної IP-адреси, IP-адреси основного шлюзу та маски підмережі у вихідному буфері та на терміналі відповідно існують слова "з крапкою" `ETHIP.`, `ETHGATEWAY.`, `ETHMASK.`

Наведемо приклад застосування слів `ETHIP` та `ETHIP.` (застосування решти слів аналогічне):

```
> ." 192.168.2.200 " ETHIP  
192.168.2.200 (OK)  
> ETHIP.  
192.168.2.200 (OK)  
>
```

Слово `MBTCPPOINT!` встановлює порт сервера `MODBUS TCP`. Слово працює наступним чином: зі стеку даних знімається верхнє значення - порт сервера в межах від 1 до 65535 і запам'ятовується у відповідній комірці внутрішньої енергонезалежної пам'яті. Для отримання на стеку даних актуального значення порту сервера `MODBUS TCP` існує слово `MBTCPPOINT?`

Наведемо приклад застосування слів `MBTCPPOINT!` та `MBTCPPOINT?`:

```
> 502 MBTCPPOINT!  
(OK)  
> MBTCPPOINT? .  
502 (OK)  
>
```

Реєстратор

Вбудований в контролери серії ES-ForthLogic™ реєстратор працює незалежно від алгоритму роботи контролера і може бути, наприклад, налаштований за допомогою конфігураційного меню. Проте, для гнучкості процесу реєстрації, реалізовано низку слів та системних констант, які дозволяють повністю здійснити згадані налаштування безпосередньо з програми. В процесі реєстрації застосовується подвійна буферизація на рівні даних та подвійна буферизація на рівні файлів даних (про неї окремо згадано далі по тексту).

Система автоматично і абсолютно прозоро для користувача підтримує механізм двох буферів даних, розташованих в оперативній пам'яті. Незалежно від режиму роботи, дані спочатку потрапляють в один з цих буферів і при його заповненні автоматично переписуються або в енергонезалежну пам'ять або передаються через існуюче GPRS-з'єднання, що визначається режимом реєстрації. Інший буфер, при цьому, дозволяє автоматично продовжувати реєстрацію даних. Така буферизація є практично необхідною при частій реєстрації а також для врахування ресурсу енергонезалежної пам'яті. Розмір буферів даних складає LOGBUF_MAX байт і дозволяє значно скоротити частоту звертання до енергонезалежної пам'яті, яка має обмежений ресурс (~100000 циклів запису). Параметр LOGBUF_MAX залежить від апаратної платформи. Застосування буферизації даних при реєстрації в енергонезалежну пам'ять дозволяє застосовувати інтервали реєстрації від 30 хвилин і більше.

Крім режимів реєстрації в енергонезалежну пам'ять через буфери в оперативній пам'яті, в системі реалізовано режим реєстрації в оперативну пам'ять. Такий режим дозволяє застосовувати інтервали реєстрації від 1 сек до 30 хвилин і не залежить від ресурсу енергонезалежної пам'яті. При цьому розмір сховища в оперативній пам'яті в байтах дорівнює подвійному значенню LOGBUF_MAX.

Слово LOGON (включити реєстрацію) вживається разом із системними константами EVENTS_MODE (режим подій), USER_MODE (режим користувача), INTERVAL_MODE (режим інтервалів), TO_SD (на SD карту), TO_FLASH (на флеш пам'ять), TO_RAM (в оперативну пам'ять), TO_TCP (передача по протоколу TCP) ALL_DATA (всі дані), INPUTS (входи) та OUTPUTS (виходи) і призначено для запуску процесу реєстрації з одночасним встановленням опцій реєстрації. Всі налаштування відразу зберігаються у внутрішній енергонезалежній пам'яті. Виклик даного слова може приймати наступні форми (порядок слів має значення!):

<data> <dest> EVENTS_MODE LOGON

- запускається процес реєстрації в режимі подій;

<dest> USER_MODE LOGON

- запускається процес реєстрації в режимі користувача;

<data> <dest> <interv> INTERVAL_MODE LOGON

- запускається процес реєстрації в режимі інтервалів;

Числові параметри в кутових дужках є обов'язковими:

<data> - набір даних, може приймати значення ALL_DATA, INPUTS або OUTPUTS;

<dest> - місце зберігання даних, може приймати значення TO_SD, TO_FLASH, TO_RAM або TO_TCP;

<interv> - ціле число рівне інтервалу в секундах.

Слово LOGON кладе на стек логічне значення - результат свого виконання. При цьому, логічне значення ІСТИНА означає, що реєстрація успішно розпочалася, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, процес реєстрації вже триває або не вдалося відкрити файл).

Наведемо приклад застосування слова LOGON:

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

В першому випадку ми запустили *"реєстрацію всіх даних на флеш-пам'ять в режимі подій"*. В другому випадку ми запустили *"реєстрацію стану виходів на карту SD/MMC що 25 сек. в режимі інтервалів"*.

Процес реєстрації зупиняється словом LOGOFF (припинити реєстрацію), а слово LOGRUN (запустити реєстрацію) призначено для запуску реєстрації з існуючими опціями - воно корисно коли виникає необхідність для частого запуску та зупинки вже налагодженого процесу реєстрації. Слово LOGRUN кладе на стек логічне значення - результат свого виконання. При цьому, логічне значення ІСТИНА означає, що реєстрація успішно розпочалася, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, процес реєстрації вже триває або не вдалося відкрити файл).

Для переносу даних реєстрації з внутрішньої енергонезалежної пам'яті на карту SD/MMC існує слово LOG>SD. Слово LOG>SD працює наступним чином: з вихідного буфера зчитується рядок - назва файлу і здійснюється спроба або відкрити даний файл на карті SD/MMC, якщо він існує, або створити його, далі відбувається перенесення даних з внутрішньої пам'яті в даний файл а на стек кладеться логічне значення - результат виконання. При цьому, логічне значення ІСТИНА означає, що дані успішно розпочали переноситись, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, триває процес реєстрації або не вдалося відкрити файл). Допускається не вказувати назви файлу, тоді буде використано стандартний файл "datalog.txt".

Для передачі даних реєстрації з внутрішньої енергонезалежної пам'яті по протоколу TCP існує слово LOG>TCP. Слово LOG>TCP здійснює спробу перенести дані з внутрішньої пам'яті через існуюче GPRS-з'єднання згідно протоколу TCP. На стек при цьому кладеться логічне значення - результат виконання даного слова: логічне значення ІСТИНА означає, що дані успішно почали переноситись а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, триває процес реєстрації, не вдалося відкрити файл або відсутнє GPRS-з'єднання).

Передача даних через існуюче GPRS-з'єднання згідно протоколу TCP відбувається без додаткових засобів перевірки цілісності переданих даних - приймається, що протокол TCP/IP є достатньо надійним і автоматично забезпечує цілісність даних. Проте, ситуація раптового розриву GPRS-з'єднання може призвести до втрати даних які передавались в цей момент. Тому потрібні додаткові заходи гарантування успішної доставки даних реєстрації. Серед можливих механізмів реалізації цього завдання було обрано передачу даних реєстрації на сервер баз даних MySQL із застосуванням протоколу HTTP. Для цього існують слова LOG>DB та LOG>SQL. Слово LOG>DB здійснює спробу перенести дані з внутрішньої енергонезалежної пам'яті через існуюче GPRS-з'єднання в режимі CLIENTDB згідно протоколу HTTP. Слово LOG>SQL здійснює спробу перенести дані з внутрішньої оперативної пам'яті через існуюче GPRS-з'єднання в режимі CLIENTSQL згідно протоколу HTTP. На стек при цьому кладеться логічне значення - результат виконання даного слова: логічне значення ІСТИНА означає, що дані успішно почали переноситись а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, триває процес реєстрації, не вдалося відкрити файл або відсутнє GPRS-з'єднання в режимі CLIENTDB або в режимі CLIENTSQL).

Передача даних через існуюче провідне з'єднання ETHERNET згідно протоколу TCP відбувається за допомогою слова LOG>DBTCP. Слово LOG>DBTCP здійснює спробу перенести дані з внутрішньої оперативної пам'яті через існуюче провідне з'єднання ETHERNET згідно протоколу HTTP. На стек при цьому кладеться логічне значення - результат виконання даного слова: логічне значення ІСТИНА означає, що дані успішно почали переноситись а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, триває процес реєстрації, не вдалося відкрити файл або відсутнє провідне з'єднання ETHERNET з віддаленим сервером).

Процес переносу даних реєстрації з внутрішньої енергонезалежної пам'яті може бути досить тривалим (залежно від розміру зареєстрованих даних). Для з'ясування стану процесу переносу даних існує слово LOGSEND?, яке кладе на стек логічне значення - стан процесу переносу даних. При цьому, логічне значення ІСТИНА означає що здійснюється перенос, а НЕ ІСТИНА - що перенос завершено.

Стан процесу реєстрації можна з'ясувати за допомогою слова LOG?, яке кладе на стек логічне значення - стан процесу реєстрації. При цьому, логічне значення ІСТИНА означає, що триває реєстрація, а НЕ ІСТИНА - що реєстрація зупинена.

Для з'ясування вільного місця у внутрішній пам'яті існує слово FREELOG? (вільне місце для реєстрації), яке повертає на стек даних кількість вільного місця у внутрішній енергонезалежній пам'яті в байтах. Для повної чистки внутрішньої пам'яті реєстрації існує слово LOGFORMAT (форматувати пам'ять реєстрації), яке виконує форматування внутрішньої файлової системи з видаленням всіх зареєстрованих даних.

При реєстрування даних у внутрішню енергонезалежну пам'ять застосовується принцип подвійної буферизації на рівні файлів даних. Система автоматично і абсолютно прозоро для користувача підтримує систему з двох файлів реєстру у внутрішній пам'яті даних. Коли користувач розпочинає реєстрацію за допомогою слів LOGON або LOGRUN то дані потрапляють до одного з файлів реєстру (до якого саме - визначає система), який є на даний момент *активним*. Наступні зупинки та відновлення реєстрації відбуваються до того самого файлу. Так відбувається доти, доки користувач не застосує одне зі слів LOG>SD, LOG>DB або LOG>TCP. З цього моменту система автоматично вибирає для реєстрування інший файл реєстру, а переписування даних на карту SD, в базу даних або по протоколу TCP відбувається з попереднього файлу реєстру. Такий механізм дозволяє здійснювати паралельну роботу реєстратора з переписуванням вже зареєстрованих даних на карту SD, в базу даних або по протоколу TCP. Якщо процес переписування вже зареєстрованих даних закінчується успішно, до файл з цими даними видаляється з внутрішньої пам'яті, якщо ж ні - то він залишається і система наступного разу використає його для дописування нових даних. Таким чином в системі одночасно можуть бути присутні до двох файлів реєстру. Це не є проблемою, бо за допомогою двох послідовних процесів переписування їх можна видобути з внутрішньої пам'яті.

За певних обставин буває необхідним вручну змінити активний файл реєстру на інший. Для цього призначено слово LOGFILETOGGLE (переключення файл реєстру). Також можна вручну видалити не активний на даний момент файл реєстру - для цього призначене слово LOGFILEDELETE (видалення файлу реєстру). Якщо обраний режим реєстрації у внутрішню оперативну пам'ять, то дане слово видаляє з цієї пам'яті всі накопичені дані.

При реєстрування даних у внутрішню оперативну пам'ять подвійна буферизація на рівні файлів даних не застосовується - є лише один файл зареєстрованих даних в оперативній пам'яті.

Для з'ясування наявності зареєстрованих даних у внутрішній пам'яті існує слово LOGFILE? (файли реєстрації), яке кладе на стек числове значення – кількість файлів зареєстрованих даних. При цьому, значення 1 або 2 означає, що файли присутні, а значення 0 - що зареєстрованих файлів немає.

Під час автоматичної реєстрації дані записуються порядково в текстовому вигляді у файл "datalog.txt", який розміщується або у вбудованій енергонезалежній пам'яті, або на карті пам'яті SD/MMC. Для різних опцій набору даних реєстрації один запис має різний формат і завжди закінчується символами

кінця рядка та переведення рядка - \n\r (в шістнадцятковому представленні 0x0A та 0x0D):

- ALL_DATA - реєструються час, дата, напруги живлення, входи та виходи у наступній формі:

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

13:04:39	19/03	18.4 13.8	22.23 20.29 0.12 134.34	10000000	0000	000
години, хвилини, секунди	день, місяць	напруга живлення, напруга акумулятора, В	значення аналогових входів AI1...AI4	логічний стан входів DI1...DI8	логічний стан виходів DO1...DO4	логічний стан виходів S1...S3

- INPUTS - реєструються час, дата та входи у наступній формі:

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

13:04:39	19/03	22.23 20.29 0.12 134.34	10000000
години, хвилини, секунди	день, місяць	значення аналогових входів AI1...AI4	логічний стан входів DI1...DI8

- OUTPUTS - реєструються час, дата та виходи у наступній формі:

"13:04:39|19/03|0000|000\n\r", де:

13:04:39	19/03	0000	000
години, хвилини, секунди	день, місяць	логічний стан виходів DO1...DO4	логічний стан виходів S1...S3

Під час автоматичної реєстрації даних в режимі інтервалів відбувається періодичний запис рядків у вказаному вище форматі. Під час автоматичної реєстрації даних в режимі подій запис рядків відбувається лише при виявленні будь-яких змін на унітарних входах або виходах. Додатково в цьому режимі фіксуються всі вхідні/вихідні дзвінки та SMS у наступному форматі рядка:

"13:04:39|19/03|SMS>|+380501234567|Hello world!\n\r", де:

13:04:39	19/03	SMS>	+380501234567	Hello world!
години, хвилини, секунди	день, місяць	тип події	номер телефону	для SMS - текст повідомлення

Тип події може бути наступним:

- ">SMS" - поступило вхідне SMS;
- "SMS>" - відправлено вихідне SMS;
- ">VOICE" - розпочався вхідний голосовий виклик;
- "VOICE>" - розпочався вихідний голосовий виклик;
- "HOLD" - припинився голосовий виклик.

Значення аналогових входів представляють собою цілі числа, які відповідають відлікам 10-розрядного АЦП для відповідного входу. Проте для повної версії контролера, в якій реалізовано конфігураційне меню та фіксований алгоритм роботи, значення відповідають масштабованій фізичній величині представленої у вигляді чисел з фіксованою комою та двома цифрами після коми. Масштабні коефіцієнти приймаються зі значень вказаних у відповідних пунктах конфігураційного меню контролера. Таким чином для повної версії контролера реєструються реальні показники фізичних величин (наприклад температура) від датчиків з уніфікованим аналоговим вихідним сигналом.

Окрім даних реєстрації, які записуються системою автоматично, існує можливість записати в системний реєстр дані безпосередньо з програми. Слово LOG (реєстрація) записує вміст вихідного буфера в системний реєстр. Дані записуються тільки під час роботи реєстратора в режимі користувача і, по замовчуванню, їм передують автоматичні поля з часом, датою та ключовим словом "ForthLogic>":

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

Максимальна довжина рядка тексту, який зчитується з вихідного буфера, дорівнює 120 символів. Згадувані автоматичні поля можна за допомогою слів LOGTITLEON (увімкнути заголовок реєстрації) та LOGTITLEOFF (вимкнути заголовок реєстрації) вмикати та вимикати за бажанням в будь-який момент часу і тим самим звільняти місце у вихідному буфері для корисних даних. Якщо виникає задача реєструвати рядки даних які перевищують 120 символів, то це можна здійснити вимкнувши режим автоматичного додавання до рядків даних символів закінчення рядка та переносу тексту на новий рядок "\n\r". В цьому випадку програма сама визначає коли записати в файл реєстру ці символи за допомогою слова NEWLINE. Згадуване автоматичне додавання символів закінчення рядка та переносу тексту на новий рядок можна за допомогою слів LOGTERMINON (увімкнути закінчення рядка) та LOGTERMINOFF (вимкнути закінчення рядка) вмикати та вимикати за бажанням в будь-який момент часу.

Наведемо приклад застосування слів для реєстрації даних користувача безпосередньо з програми:

```
> : 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 є допоміжним: Print друкує у вихідному буфері словами "ON" та "OFF" логічний стан на вершині стеку. Слова log1 та log2 здійснюють, власне, саму реєстрацію даних користувача через рівні проміжки часу. Весь механізм запускається та зупиняється словами StartLog та StopLog.

Підсистема живлення

Слово POW? кладе на математичний стек напругу основного живлення у вольтах. Слово BAT? кладе на математичний стек напругу живлення акумулятора у вольтах. Наведемо приклад застосування даних слів:

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

Системний час

Слово TIME? кладе на стек три значення в такій послідовності: секунди, хвилини, години системного годинника. Відповідно, при знятті зі стеку, спочатку будуть отримані години, потім хвилини і, нарешті, секунди:

```
TIME?      -      ---> СЕК, ХВ, ГОД
```

Слово DATE? кладе на стек три значення в такій послідовності: рік, місяць, день місяця системного годинника. Відповідно, при знятті зі стеку, спочатку буде отриманий день місяця, потім місяць і, нарешті, рік:

```
DATE?      -      ---> РОКИ, МІС, ДНІ
```

Слово WDAY? кладе на стек значення днів тижня, при цьому, неділя відповідає значенню 0, понеділок - 1, вівторок - 2 і так далі:

```
WDAY?      -      ---> ДНІ ТИЖДНЯ
```

Слово DST? кладе на стек прапорець дії літнього часу. Значення ІСТИНА означає дію літнього часу:

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

Слово UTC? кладе на стек стан системного годинника у вигляді секунд так, як це прийнято в операційній системі UNIX. Моментом початку відліку секунд вважається північ з 31 грудня 1969 року на 1 січня 1970, час з цієї миті називають "ерою UNIX" (англ. Unix Epoch). Час UNIX узгоджується з часом UTC. Спосіб зберігання часу у вигляді кількості секунд дуже зручно використовувати при порівнянні дат (з точністю до секунди), а також для зберігання дат: при необхідності їх можна перетворити в будь-який легкий для читання формат.

Для перетворення загальноприйнятого представлення часу в час UTC існує слово >UTC.

>UTC	СЕК, ХВ, ГОД, РОКИ, МІС, ДНІ	--->	UTC
------	------------------------------	------	-----

Для перетворення часу UTC в загальноприйняте представлення часу існує низка слів: >TIME, >DATE та >WDAY. Ці слова знімають зі стеку значення часу у вигляді секунд UTC та повертають на стек час, дату та день тижня. Найкраще описати роботу даних слів за допомогою стекової нотації:

>TIME	UTC	--->	СЕК, ХВ, ГОД
>DATE	UTC	--->	РОКИ, МІС, ДНІ
>WDAY	UTC	--->	ДНІ ТИЖНЯ

При цьому, дні тижня інтерпретуються наступним чином: неділя відповідає значенню 0, понеділок - 1, вівторок - 2 і так далі.

Для друку у вихідному буфері та на терміналі форматowanego загальноприйнятого календарного часу існує слово **TIMESTAMP** (мітка часу). Календарний час друкується у наступному вигляді (без лапок): "година:хвилини:секунди день/місяць/рік". Наприклад "15:02:45 10/01/11".

Для порівняння між собою загальноприйнятих дат та часів існує слово **ISNOW**, яке попарно порівнює шість верхніх значень на стеку даних (прибираючи їх зі стеку) та повертає на стек логічне значення **ІСТИНА**, тільки в тому випадку, якщо всі порівняння істинні, та **НЕ ІСТИНА** у іншому випадку. Порівняння відбувається в особливому порядку:

ISNOW	A1, B1, C1, C2, B2, A2	--->	$(A1=A2) \cdot (B1=B2) \cdot (C1=C2)$
-------	------------------------	------	---------------------------------------

якщо $A1=A2$, $B1=B2$, $C1=C2$, то на стеку буде -1 (ІСТИНА), інакше 0 (НЕ ІСТИНА). Наведемо приклад застосування всіх вищезгаданих слів:

```
> 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 останньому рядку наведено приклад встановлення системного часу рівним 15:00:00 без зміни дати.

Для задач керування освітленням реалізовано функцію *астрономічного таймера*. Дана функція дозволяє отримати моменти часу сходу та заходу сонця. Функція розрахунку моментів сходу та заходу Сонця реалізується для поточної дати на основі показів системного годинника. В якості параметрів використовуються географічні координати місця встановлення контролера та тип розрахунку. Географічні координати вказуються із врахуванням знаку в межах від -90.0 до +90.0 градусів для широти (додатня для Північної, негативна для Південної широти) та в межах від -180.0 до +180.0 градусів для довготи (додатня для Східної, негативна для Західної довготи). Тип розрахунку вибирається з двох можливих: цивільні сутінки та світанки та офіційний схід та захід Сонця. Цивільні

сутінки - коли верхній край диску Сонця опускається на 6 градусів нижче під горизонтом. У цьому випадку діяльність людини потребує додаткового вуличного освітлення. Цивільний світанок - коли починає розвиднюватися і верхній край диску Сонця ще знаходиться на 6 градусів нижче під горизонтом. Офіційний захід Сонця - момент коли верхній край диску Сонця зникає за горизонтом. Офіційний схід Сонця - момент коли верхній край диску Сонця з'являється над горизонтом.

Функція розрахунку моментів сходу та заходу Сонця представлена двома словами та двома системними константами. Слова SUNRISE? (схід сонця) SUNSET? (захід сонця) працюють наступним чином. З математичного стеку вони знімають два числа - широту та довготу, зі звичайного стеку знімають тип розрахунку, який задається за допомогою системних констант CIVIL (цивільні сутінки та світанки) та OFFICIAL (офіційний схід та захід Сонця) та кладуть на стек три значення в такій послідовності: секунди, хвилини та години відповідного моменту сходу або заходу сонця. Відповідно, при знятті зі стеку, спочатку будуть отримані години, потім хвилини і, нарешті, секунди. В стековій нотації дані слова мають наступний вигляд:

```
SUNRISE?  тип_розрахунку      ---> сек, хв, год
          F: широта, довгота  ---> -

SUNSET?   тип_розрахунку      ---> сек, хв, год
          F: широта, довгота  ---> -
```

Якщо для даної місцевості Сонце не може зійти або сховатись за горизонт, то значення секунд, хвилин та годин відповідного моменту сходу або заходу сонця буде дорівнювати 0. Наведемо приклад програми по застосуванню згаданих слів:

```
( Astronomic timer example )
( )
(DO2 \_____ /_____ \_____ )
(      dawn          dusk          dawn      )
( )
( Used timers: #6 - for 0.75 sec offtask cycle )
(          #7 - for 0.75 sec ontask cycle      )

49.82766 FCONSTANT Latitude
24.00867 FCONSTANT Longitude

: sunrise? Latitude Longitude OFFICIAL SUNRISE? SWAP ROT ;
: sunset? Latitude Longitude OFFICIAL SUNSET? SWAP ROT ;
: dawn? Latitude Longitude CIVIL SUNRISE? SWAP ROT ;
: dusk? Latitude Longitude CIVIL SUNSET? SWAP ROT ;

: offtask dawn? TIME? ISNOW IF 0 2 DO! THEN 0.75 6 TIMER! offtask ;
: ontask dusk? TIME? ISNOW IF 1 2 DO! THEN 0.75 7 TIMER! ontask ;

( main word
: main offtask ontask ;

." main " BOOT
main
```

Інтерфейс користувача

Клавіатура

Для контролерів з вбудованою клавіатурою та дисплеєм, для налаштування функції клавіш **F1**, **F2**, **Δ**, **▽**, **◁**, **▷** та **OK** існує слово **BUTTON**, яке застосовується разом із системні константи імена яких відповідають назві клавіш: **F1**, **F2**, **UP** (**Δ**), **DOWN** (**▽**), **LEFT** (**◁**), **RIGHT** (**▷**), **OK**. Ці константи кладуть на стек номери відповідних клавіш. Слово **BUTTON** працює наступним чином: зі стеку знімається верхнє значення - номер клавіші в діапазоні від 1 до 7, з вхідного тексту вибирається чергове слово (введене після слова **BUTTON**), яке буде виконуватися при натисканні відповідної клавіші, і здійснюється налаштування даної клавіші згідно заданих параметрів. Наведемо приклад застосування слова **BUTTON** для налаштування функції клавіші **F2**:

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

В процесі виконання задачі користувача, функції клавіш можна багатократно змінювати в залежності від контексту задачі. Визначені таким чином функції клавіш **F1** та **F2** будуть активними у *всіх режимах інтерфейсу* крім випадку, коли в режимі меню відбувається введення значення, а функції клавіш **Δ**, **▽**, **◁**, **▷** та **OK** будуть активними лише в *робочому режимі інтерфейсу*.

Слова які призначені клавішам не запам'ятовуються при виключенні живлення контролера. Це слід враховувати при програмуванні задач користувача.

Дисплей

Виведення інформації

Для контролерів з графічним кольоровим дисплеєм, для виведення текстової інформації на дисплей існує слово **PRINT** (друкувати). Виведення можливе лише в *робочому режимі інтерфейсу* - в інших режимах форт-система блокує виведення. Робоче поле для виведення текстової інформації являє собою прямокутну ділянку посередині дисплея. Максимальна кількість рядків та символів в рядку залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Нумерація рядків - зверху вниз від 0 до **ROW_MAX-1**, нумерація символів - зліва направо від 0 до **COL_MAX-1**. Текст відображається за допомогою великого шрифту в одному з 9 кольорів (білий, червоний, помаранчевий, жовтий, зелений, блакитний, синій, фіолетовий, чорний) або на білому фоні або на чорному. По замовчуванню (після подачі живлення на контролер), виведення відбувається чорним шрифтом на білому фоні.

Слово **PRINT** працює наступним чином: зі стеку знімається два верхніх значення - координати позиції виведення тексту, які складаються з номера рядка та номера символу, з вихідного буфера зчитується рядок - текст, який буде виводитися у робоче поле починаючи з даної позиції і відбувається виведення на активний фон дисплея в активному кольорі. При досягненні кінця рядка, символи тексту автоматично переносяться у наступний рядок. В останньому рядку, при досягненні кінця рядка, подальше виведення блокується.

Активний колір шрифту можна встановити за допомогою слів **WHITE** (білий), **RED** (червоний), **ORANGE**

(помаранчевий), YELLOW (жовтий), GREEN (зелений), BLUE (блакитний), DEEPBLUE (синій), VIOLET (фіолетовий) та BLACK (чорний), які працюють однаковим чином: встановлюють відповідний активний колір тексту. Всі подальші виведення відбуваються в активному кольорі аж до його чергової зміни.

Активний фон виведення (а також активний на даний момент колір) за допомогою слова INVERT (інвертувати) можна поміняти на протилежний, здійснюючи таким чином інверсію кольорів, в результаті якої білий фон стає чорним (або навпаки) а колір тексту міняється на доповнюючий ("негативний"). Всі подальші виведення відбуваються в даному кольорі на даний фон аж до їх чергової зміни.

Для чистки робочого поля виведення текстової інформації існує слово CLEAR (очистити), яке витирає всі символи відповідно до активного фону - робоче поле стає або білим або чорним.

Наведемо приклад застосування слів для роботи з дисплеєм:

```
> ." "Червоний" " RED 0 0 PRINT
(OK)
> ." "Зелений" " GREEN 0 1 PRINT
(OK)
> ." "Інверсія зеленого" " INVERT 0 2 PRINT
(OK)
> ." "Синій на чорному" " DEEPBLUE 0 4 PRINT
(OK)
> CLEAR
(OK)
>
```

Слово "Червоний" буде виведено починаючи з символу 0 рядка 0 в червоному кольорі. Слово "Зелений" буде виведено починаючи з символу 0 рядка 1 в зеленому кольорі. Фраза "Інверсія зеленого" буде виведена починаючи з символу 0 рядка 2 в фіолетовому кольорі на чорному фоні. Фраза "Синій на чорному" буде виведена починаючи з символу 0 рядка 4 в синьому кольорі на чорному фоні. Зверніть увагу, що після виконання слова CLEAR, робоче поле (фон) стане чорним.

Введення інформації

Для введення числових параметрів за допомогою клавіш та дисплея існує слово GET, яке працює наступним чином: з математичного стеку знімається верхнє значення - число, яке буде відображено у полі вікна для введення значень, з вихідного буфера зчитується рядок - текст, який буде відображено у шапці вікна для введення значень довжиною не більше 12 літер, з вхідного тексту вибирається два чергових слова зворотнього виклику (введених після слова GET) і на дисплеї відображається стандартне вікно для введення значень.

При редагуванні значень у вікні для введення, переміщення курсора здійснюється клавішами , при цьому курсор підсвічується рожевим кольором. Клавішею **F1** здійснюється вибір набору символів зі списку: [S] - знаки пунктуації; [D] – цифри; [L] - латинські великі літери; [l] - латинські малі літери; [K] - кириличні великі літери; [k] - кириличні малі літери; при цьому позначення активного набору символів здійснюється в лівому верхньому куті вікна для введення (на синьому фоні). Клавішами  здійснюється перебір символів з обраного набору. Видалення символу в позиції курсора здійснюється клавішею **F2**, затвердження обраного значення - клавішею **OK**, а вихід без внесення змін - клавішею **Esc**.

Коли користувач закінчить ввід та натисне клавішу **OK**, форт-система виконає перше з вказаних слів зворотнього виклику, причому, на момент виконання даного слова, на математичному стеку буде знаходитись введене значення. У випадку, коли користувач натисне клавішу **Esc**, форт-система

виконає друге з вказаних після слова GET слів зворотнього виклику. Введення числових параметрів можливе лише в *робочому режимі інтерфейсу* та в межах меню "Функції Користувача" - в інших випадках форт-система блокує роботу слова GET. Наведемо приклад застосування слова GET:

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

Для введення рядкових параметрів за допомогою клавіш та дисплея існує слово GETS, яке працює наступним чином: зі стеку знімається верхнє значення - номер рядкової змінної з текстом, який буде відображено у полі вікна для введення значень та для збереження введеного рядка, з вихідного буфера зчитується рядок - текст, який буде відображено у шапці вікна для введення значень довжиною не більше 12 літер. Далі, з вхідного тексту вибирається два чергових слова зворотнього виклику (введених після слова GETS) і на дисплеї відображається стандартне вікно для введення значень. Правила редагування значень аналогічні до описаних в попередньому параграфі для слова GET.

Коли користувач закінчить ввід та натисне клавішу **OK**, форт-система виконає перше з вказаних слів зворотнього виклику, причому, на момент виконання даного слова, у вказаній рядковій змінній буде знаходитись введений рядок. У випадку, коли користувач натисне клавішу **Esc**, форт-система виконає друге з вказаних після слова GETS слів зворотнього виклику. Введення рядкових параметрів також можливе лише в *робочому режимі інтерфейсу* та в межах меню "Функції Користувача" - в інших випадках форт-система блокує роботу слова GETS. Наведемо приклад застосування слова GETS:

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

Меню користувача

Для контролерів з графічним дисплеєм можна налаштовувати особливе меню користувача, яке функціонує в складі загальносистемного конфігураційного меню. Максимальна кількість пунктів такого меню залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Для налаштування пунктів меню користувача, існують слова MENU та HIDE. Слово MENU працює наступним чином: зі стеку знімається верхнє значення - номер меню в діапазоні від 0 до *MENU_MAX*, з вихідного буфера зчитується рядок - текст, який буде відображатися у відповідному пункті меню, з вхідного тексту вибирається чергове слово (введене після слова MENU), яке буде виконуватися при виборі відповідного пункту меню. Далі здійснюється налаштування відповідного пункту меню згідно заданих параметрів і даний пункт меню стає видимим. Допустимо не вказувати текстовий рядок для відображення, тоді назва пункту меню залишиться без змін. Номери меню від 1 до *MENU_MAX* відповідають пунктам меню користувача, а номер меню 0 відповідає за налаштування функції, яка буде виконуватись кожного разу при входженні до меню "Функції Користувача". Така функція є корисною для початкової ініціалізації структури меню користувача. Треба також пам'ятати про

максимальну довжину рядка, яка буде відображатися в назві пункту меню користувача - 15 символів, причому для пункту меню з номером 0 текст взагалі не має значення. Наведемо простий приклад застосування даного слова:

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

В даному прикладі ми визначили нове слово `inversija` яке виконує наступні дії: на стек кладеться логічний стан релейного виходу S3, здійснюється логічна інверсія вершини стеку, після чого значення зі стеку записується у релейний вихід S3. Далі ми призначили це слово першому пункту меню користувача, встановивши при цьому назву пункту на "Інверсія".

Слова, які призначені пунктам меню а також текст, який відображається в пунктах меню користувача не запам'ятовуються при виключенні живлення контролера. Це слід враховувати при програмуванні задач користувача.

Слово `HIDE` працює наступним чином: зі стеку знімається верхнє значення - номер меню в діапазоні від 1 до `MENU_MAX`, далі здійснюється виключення відповідного пункту меню - воно стає невидимим і недоступним для користувача. Повторне включення такого пункту меню відбувається словом `MENU`. Таким чином можна реалізувати динамічне меню користувача, пункти якого залежать від контексту задачі.

Визначити, який пункт був виконаний останнім можна за допомогою слова `LASTMENU?`, яке кладе на стек номер пункту меню користувача, який був виконаний останнім. Дане слово дозволяє персоналізувати роботу з окремими пунктами меню.

При кожному виконанні слів `MENU` та `HIDE`, *фокус меню* - тобто виділений видимий рядок залишається без змін. Для більшої гнучкості при створенні багаторівневих меню існує слово `FOCUS` яке працює наступним чином: зі стеку знімається верхнє значення - номер меню в діапазоні від 1 до `MENU_MAX` і здійснюється встановлення фокусу на даний пункт меню - тобто він стає виділеним рожевим кольором та відображається першим рядком в існуючій структурі меню.

Додатково, для більшої зручності користуванням меню, існує можливість виведення допоміжної текстової інформації в інформаційному полі меню користувача (ліва верхня частина дисплея). Для цього існує слово `INFO` (інформація), яке з вихідного буфера зчитує рядок тексту довжиною до 15 символів та висвітлює його в лівій частині інформаційного поля. Дане слово працює лише в межах *меню користувача*.

Всі перелічені вище слова для роботи з меню користувача дозволяють створювати багаторівневі динамічні меню користувача, які не поступаються за можливостями вбудованому конфігураційному меню.

Звук

Для формування звукового сигналу за допомогою вбудованого *генератора звукових сигналів*, існує слово `BEEP` (сигнал), яке працює наступним чином: з математичного стеку знімається верхнє значення - тривалість сигналу в діапазоні (0,001 ... 64,0) секунди з кроком 0,001 сек, зі стеку даних знімається верхнє значення - частота звукового сигналу в діапазоні (100 ... 10000) Гц і відбувається генерація одиночного звукового сигналу з заданою частотою та тривалістю.

Слово `BEEP` дозволяє додатково оформити інтерфейс з користувачем звуковими подіями, що, в

цілому, збільшує приязність даного інтерфейсу. Наведемо приклад застосування слова BEEP для генерації сигналу частотою 1000 Гц та тривалістю 2 секунди:

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

Індикація

Для індикації за допомогою вбудованих світлодіодів, існують слова LED? та LED!, які використовуються разом із системними константами F1 та F2. Імена цих констант відповідають назвам світлодіодів і вони кладуть на стек номер відповідних світлодіодів. Слово LED? працює наступним чином. Зі стека даних знімається верхнє значення - номер світлодіода, а на стек даних кладеться стан відповідного світлодіода який приймає значення ІСТИНА або НЕ ІСТИНА. Слово LED! працює іншим чином. Зі стека даних знімається два верхніх значення - номер світлодіода та нове значення стану світлодіода у вигляді ІСТИНА або НЕ ІСТИНА. Стан ІСТИНА означає що світлодіод світиться. Приведемо приклад роботи з цими словами:

```
> TRUE F1 LED!  
(OK)  
> F1 LED? . F2 LED? .  
-1 0 (OK)  
>
```

Акустична система

Деякі моделі контролерів обладнані акустичною системою (далі по тексті - *аудіосистема*), через яку можна відтворювати повідомлення та числа аналогічно до того як це робиться при відтворенні голосового меню через GSM модуль.

Для відтворення звукового повідомлення через аудіосистему в будь-який момент часу, існує слово AUDIOPLAY (відтворити аудіо) яке працює наступним чином: з вихідного буфера зчитується рядок - назва файлу у форматі *.wav, який розташовано на карті пам'яті SD/MMC (процедура створення звукових файлів описана в додатку), з вхідного тексту вибирається чергове слово зворотнього виклику (введене після слова AUDIOPLAY) і запускається процес відтворення звукового повідомлення, а на стек кладеться логічне значення - результат виконання слова AUDIOPLAY. При цьому, логічне значення ІСТИНА означає, що процес відтворення успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, файл або карта пам'яті відсутні або вже здійснюється відтворення через аудіосистему, тощо). Коли звукове повідомлення буде повністю відтворено, форт-система виконає вказане після слова AUDIOPLAY слово зворотнього виклику.

Для відтворення через аудіосистему словами значення числа з вершини математичного стеку існує слово AUDIOSAY (сказати) яке працює наступним чином: з вихідного буфера зчитується рядок - шлях до набору файлів чисел у форматі *.wav, який розташований на карті пам'яті SD/MMC, з математичного стеку знімається верхнє значення, яке має знаходитись в межах від -999,9 до +999,9 включно, з вхідного тексту вибирається чергове слово зворотнього виклику (введене після слова AUDIOSAY) і запускається процес відтворення словами значення числа, а на стек даних кладеться логічне значення - результат виконання слова AUDIOSAY. При цьому, логічне значення ІСТИНА означає, що процес відтворення успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми або число знаходиться поза встановленими межами. Коли значення числа буде повністю відтворено, форт-система виконає вказане після слова AUDIOSAY слово зворотнього виклику. Набір файлів чисел

у форматі *.wav описаний в додатку.

По замовчуванню значення числа з вершини математичного стеку відтворюється з вказанням знаку числа та з точністю 1 цифра після десяткової коми. Відтворення знаку числа можна відключити за допомогою слова NOAUTOSAYPLUS та повторно включити за допомогою слова AUTOSAYPLUS.

Для припинення відтворення звукового повідомлення через аудіосистему або числа з вершини математичного стеку існує слово AUDIOMUTE, яке негайно припиняє відтворення звуку через аудіосистему та блокує виконання заданого після слів AUDIOPLAY або AUDIOSAY слова зворотнього виклику.

Модуль GSM/GPRS

Стан мережі GSM

Для забезпечення гнучкості та зручності використання GSM модуля, в форт-системі доступна інформація, яка стосується стану мережі GSM, стану зв'язку, тощо. На даний момент реалізовано наступні слова для отримання додаткової інформації про стан модуля та мережі.

Слово ROAMING? повертає на стек логічне значення ІСТИНА, якщо GSM модуль зареєструвався в роумінгу або логічне значення НЕ ІСТИНА якщо GSM модуль зареєструвався в домашній мережі. Дане слово може стати в нагоді в прикордонних місцевостях для програмного обмеження трафіку у випадку реєстрації в роумінгу.

Слово OPERATOR друкує у вихідному буфері та на терміналі назву оператора мережі GSM, в якій зареєстровано GSM модуль. Якщо модуль не зареєстровано, то рядок буде пустим.

Слово SIGNAL? повертає на стек числове значення рівня сигналу мережі GSM. Якість сигналу при успішній реєстрації в мережі, може приймати значення від 0 до 4. Якщо виникли проблеми з реєстрацією або самим GSM модулем то слово SIGNAL? повертає на стек число -1. Дане слово може служити індикатором початку успішної роботи в мережі GSM. Інформація про якість сигналу в мережі GSM поновлюється форт-системою що 5 сек.

Слово SIM? повертає на стек логічне значення ІСТИНА, якщо SIM-карта встановлена в зчитувач контролера і успішно активована або логічне значення НЕ ІСТИНА у протилежному випадку.

Для отримання більш докладної інформації про якість сигналу в мережі GSM та стан GSM модуля існує слово MODULE? яке повертає на стек чотири значення:

```
MODULE?      -      --->      rssi,ber,voltage,temp
```

де *rssi* - внутрішнє представлення якості сигналу мережі GSM, *rssi* може приймати значення від -1 до 31, значення -1 свідчить про неможливість виявити сигнал, приблизне співвідношення між значенням *rssi* та потужністю сигналу в dBm:

0	-115 dBm або менше
1	-111 dBm
2...30	-110... -54 dBm
31	-52 dBm або більше;

ber - процент помилок при передачі даних GPRS;

voltage - внутрішня напруга живлення GSM модуля у мВ;

temp - внутрішня температура GSM модуля у градусах.

Наведемо приклад застосування даних слів:

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

Номери телефонів

У наступних прикладах всі номери телефонів можуть бути статично визначені за допомогою рядків. Для того, аби мати змогу міняти номери телефонів без перебудови всієї задачі, в енергонезалежній пам'яті існують спеціальні рядкові змінні довжиною 15 літер для збереження номерів телефонів користувачів. Для роботи з даними змінними введено слово **USER** (користувач), яке дозволяє використовувати при описі задачі телефонні номери користувачів - фактично зчитувати ці спеціальні змінні, та слово **USERPHONE** (телефон користувача) яке дозволяє визначати та змінювати телефонні номери користувачів - тобто записувати нові значення в ці спеціальні змінні. Максимальна кількість спеціальних рядкових змінних для збереження номерів телефонів залежить від апаратної платформи на якій функціонує форт-система і приведено в додатку. Причому перші шість змінних мають статус змінних для збереження телефонних номерів *обраних користувачів*, які можуть встановлюватися за допомогою конфігураційного меню та приймають участь в політиці безпеки всієї системи - при певних умовах лише з цих телефонів можна здійснювати дистанційне керування.

Слово **USER** працює наступним чином: зі стеку знімається верхнє значення, яке може знаходитись в діапазоні від 1 до *PHONE_MAX* - це порядковий номер телефону в списку (або номер користувача), далі у вихідному буфері та на терміналі друкується номер телефону, який зберігається у спеціальній рядковій змінній з цим номером.

Слово **USERPHONE** працює наступним чином: зі стеку знімається верхнє значення, яке може знаходитись в діапазоні від 1 до *PHONE_MAX* - це порядковий номер телефону в списку (або номер користувача), з вихідного буфера зчитується рядок - номер телефону в міжнародному або національному форматі і запам'ятовується у спеціальній рядковій змінній з цим номером у внутрішній енергонезалежній пам'яті. Наведемо приклад застосування слів **USER** та **USERPHONE**:

```
> ." +380507777777 " 1 USERPHONE  
+380507777777 (OK)  
> 1 USER  
+380507777777 (OK)  
>
```

Також існує слово **LAST** (останній), яке дозволяє використовувати при описі задачі номер телефону останнього абонента. Слово **LAST** друкує у вихідному буфері та на терміналі номер телефону абонента, який останнім додзвонився на контролер, або надіслав останнє SMS повідомлення.

Наведемо ще приклади застосування даних слів:

```
> 3 USER LAST  
0501234567 +380507654321 (OK)  
>
```

SMS та USSD

Вхідні SMS представляють один із способів повідомити користувача про ті чи інші події. Проте, вхідні SMS представляють собою не просто можливу відповідь користувача, а набагато потужнішу концепцію реалізовану за допомогою мови ForthLogic™. Справа в тому, що всі вхідні SMS безпосередньо потрапляють (за невеликим виключенням) у вхідний буфер текстового інтерпретатора форт-системи і, відповідно, інтерпретуються та виконуються. Дана концепція є надзвичайно гнучка та потужна, проте, з метою підвищення надійності та безпеки цілої форт-системи, у вхідних SMS заблоковано визначення нових слів через двокрапку.

По аналогії роботи в діалоговому режимі з терміналом - *термінальному режимі*, після отримання вхідного SMS (довжина одного стандартного SMS не може перевищувати 160 символів), автоматично формується вихідний SMS з відповіддю форт-системи - це *дистанційний режим* форт-системи. Відмінність цього режиму від режиму при роботі з терміналом (*термінального режиму*) полягає в тому, що не працює слово WORDS. Крім того, формування вихідного SMS з відповіддю форт-системи можна заблокувати. Дана можливість зменшує кошти експлуатації контролера та корисна для організації взаємодії між контролерами - так званої міжмашинної взаємодії (M2M). Для цього в тексті вхідного SMS спочатку (або після пароля та принаймні одного пробілу - якщо активовано захист паролем) треба вказати службове слово NAK, відділене від решти тексту принаймні одним пробілом.

Крім того, формування вихідного SMS з відповіддю форт-системи на вхідні SMS з *коротких номерів* (менших за 8 цифр) також заблоковано. Проте, це не заважає надсилати керуючі SMS з безплатних сайтів операторів мобільного зв'язку та інших автоматизованих серверів з мережі інтернет - просто такі SMS будуть без відповіді.

Для організації взаємодії за допомогою SMS з автоматизованими системами моніторингу типу АСКОВ існує можливість ідентифікувати кожне передане та прийняте SMS. Для цього в тексті вхідного SMS спочатку (або після пароля та принаймні одного пробілу - якщо активовано захист паролем) треба вказати службове слово IDxxx, відділене від решти тексту принаймні одним пробілом. Параметр xxx це довільне число в діапазоні від 0 до 65535, яке сполучається зі службовим словом без пробілу і є ідентифікатором повідомлення. Форт-система надішле у відповідь SMS, в кінці якого до стандартних слів (OK) або (ERROR - ...) буде додано даний ідентифікатор у вигляді (IDxxx OK) та (IDxxx ERROR - ...). Застосування службового слова IDxxx дозволяє вести облік керуючих та прийнятих повідомлень в умовах, коли оператори зв'язку не гарантують черговість передачі SMS через свої мережі.

Службові слова IDxxx та NAK *не можуть* бути застосовані одночасно - це по-перше не логічно, а по-друге призводить до помилки інтерпретації тексту SMS.

Увага: для даної версії інтерпретатора мови ForthLogic™ існує обмеження щодо використання кирилических літер в текстах SMS - можна застосовувати лише латинські. Тому, при розбудові прикладного словника для конкретної задачі, необхідно враховувати даний факт!

Для передачі текстових повідомлень у вигляді SMS, існує слово SMS яке працює наступним чином: з вихідного буфера зчитується номер телефону у міжнародному або національному форматі та текст самого SMS, між якими має бути принаймні один пробіл, далі здійснюється спроба відправити SMS на вказаний номер телефону, а на стек кладеться логічне значення - результат виконання слова SMS. При цьому, логічне значення ІСТИНА означає, що спроба успішно завершилася, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або він зайнятий). Приведемо приклад відправки SMS безпосередньо з терміналу:

```
> ." +380507654321 Hello World! " SMS .  
+380507654321 Hello World! -1 (OK)  
>
```

Сукупна довжина тексту самого SMS не повинна перевищувати 160 символів, у випадку перевищення цієї довжини система автоматично її обмежить з відкиданням зайвих літер. Текст довжиною більше ніж один рядок (77 символів) можна ввести у вихідний буфер за кілька разів меншими порціями. Приведемо приклад:

```
> : послатиSMS
." +380507654321 "
." Vitaju! Napruga AI2 = " 2 AI? F. ." AI3 = " 3 AI? F.
SMS DROP ;
(OK)
> 1 FPREC! послатиSMS
+380507654321 Vitaju! Napruga AI2 = 9.1 AI3 = 349.0 (OK)
>
```

Ми визначили нове слово послатиSMS, яке друкує у вихідний буфер (та на термінал) номер телефону, пробіл та певним чином відформатований текст самого повідомлення, після цього виконується слово SMS, яке "споживає" всі ці дані та робить спробу послати SMS. Словом DROP ми відкидаємо результат роботи слова SMS, оскільки його не аналізуємо.

Як видно з прикладу, в текст повідомлення можна включати як рядки-константи, так і змінні дані - за допомогою слів . (крапка), F. (FLOAT - крапка) та FE. (FLOAT ENGINEER - крапка). При цьому можуть бути використані всі слова, що кладуть свої дані на стек даних або математичний стек. Для форматування тексту повідомлення можна використовувати слова SPACE (пробіл) та NEWLINE (новий рядок).

Одна з форм коротких текстових повідомлень в мережах стільникового зв'язку GSM має назву USSD (Unstructured supplementary service data) та використовується операторами для виконання різних сервісних завдань (підключення/відключення сервісів, поповнення рахунку, перевірка балансу, тощо). Для роботи з USSD запитами існує слово USSD яке працює наступним чином: з вихідного буфера зчитується номер телефону у міжнародному або національному форматі та рядок USSD запиту, між якими має бути принаймні один пробіл, далі здійснюється спроба відправити USSD запит оператору стільникового зв'язку, а на стек кладеться логічне значення - результат виконання слова USSD. При цьому, логічне значення ІСТИНА означає, що спроба успішно завершилася, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено, здійснюється інший USSD запит). Відповідь оператора на вдалий USSD запит відправляється у вигляді SMS на вказаний номер телефону. Сукупна довжина USSD запиту не повинна перевищувати 80 символів, у випадку перевищення цієї довжини система автоматично її обмежить з відкиданням зайвих літер.

Слово USSD можна використовувати як дистанційно (в дистанційному режимі роботи форт-системи в тексті SMS) так і локально (наприклад, виконуючи по таймеру для регулярної перевірки балансу). Якщо при виконанні слова USSD здійснюється голосовий виклик, то дія слова буде затримана аж до закінчення виклику. Приведемо приклад дистанційного використання USSD запиту для поповнення рахунку та перевірки балансу:

1. Сформувати та послати перше SMS наступного змісту (вказавши при необхідності пароль та вірну послідовність цифр з вчора поповнення):

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

у відповідь прийде SMS з відповіддю оператора на USSD запит (формат представлення залежить від оператора і може бути пустим).

2. Сформувані та послати друге SMS наступного змісту (вказавши при необхідності пароль та вірну послідовність цифр для перевірки балансу):

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

у відповідь прийде SMS з відповіддю оператора на USSD запит, в якому буде присутня інформація про поточний баланс (формат представлення залежить від оператора). В даному прикладі, для встановлення номера телефону, на який буде приходити відповідь на USSD запит, ми використали слово LAST, тим самим вказавши номер телефону з якого посилалися SMS. Телефон також можна вказати безпосередньо або за допомогою слова USER.

З технологічних міркувань, в *дистанційному режимі* роботи форт-системи заблоковано безпосереднє виконання слів SMS та DIAL (про дане слово більш детально в подальших розділах). Проте, це не стосується слова USSD, яке спроектовано відповідним чином. Слова SMS та DIAL можуть бути виконані в складі інших слів як відкладені в часі завдання.

Вхідні SMS повідомлення та вбудований текстовий інтерпретатор форт-системи природнім чином можуть бути використані для організації міжмашинної взаємодії між контролерами або між контролером та іншими інтелектуальними пристроями, які можуть обробляти SMS. Для міжмашинної взаємодії між контролерами, одному контролеру необхідно сформувані та послати іншому SMS, в якому вказати необхідні дії або слова. При цьому, в тексті такого SMS, спочатку (або після пароля) необхідно вказати службове слово NAK - таким чином, інший контролер обробивши в текстовому інтерпретаторі керуючі слова не сформує і не відішле автоматичну відповідь своєї форт-системи, яка може бути абсолютно "незрозуміла" форт-системі контролера, який ініціював обмін. Якщо в тексті вихідного SMS не вказати службову команду NAK, інший контролер сформує і відішле автоматичну відповідь своєї форт-системи, проте, таке SMS взагалі не буде сприйнято форт-системою контролера, який ініціював обмін - це запобіжний захід щоб не створювалася ситуація з ланцюжком нескінченних SMS між двома контролерами.

Голосові дзвінки

Для роботи з голосовим каналом GSM модуля, реалізовано групу слів, які дозволяють здійснити та прийняти голосовий дзвінок, підключити зовнішню гарнітуру та налаштувати реакцію форт-системи на DTMF сигнали.

Відтворення повідомлень

Для відтворення звукового повідомлення в будь-який момент часу існування голосового дзвінка, існує слово PLAY (відтворити) яке працює наступним чином: з вихідного буфера зчитується рядок - назва файлу у форматі *.wav, який розташований на карті пам'яті SD/MMC (процедура створення звукових файлів описана в додатку), з вхідного тексту вибирається чергове слово зворотнього виклику (введене після слова PLAY) і запускається процес відтворення звукового повідомлення, а на стек кладеться логічне значення - результат виконання слова PLAY. При цьому, логічне значення ІСТИНА означає, що процес відтворення успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, файл або карта пам'яті відсутні, вже здійснюється відтворення, тощо). Коли звукове повідомлення буде повністю відтворено, форт-система виконає вказане після слова PLAY слово зворотнього виклику.

Для відтворення словами значення числа з вершини математичного стеку існує слово SAY (сказати) яке працює наступним чином: з вихідного буфера зчитується рядок - шлях до набору файлів чисел у форматі *.wav, який розташований на карті пам'яті SD/MMC, з математичного стеку знімається верхнє значення, яке має знаходитись в межах від -999,9 до +999,9 включно, з вхідного тексту вибирається чергове слово зворотнього виклику (введене після слова SAY) і запускається процес відтворення словами значення числа, а на стек даних кладеться логічне значення - результат виконання слова SAY. При цьому, логічне значення ІСТИНА означає, що процес відтворення успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми або число знаходиться поза встановленими межами. Коли значення числа буде повністю відтворено, форт-система виконає вказане після слова SAY слово зворотнього виклику. Шлях до набору файлів для контролерів з дисплеєм має бути пустим (ці контролери мають вбудований набір файлів). Набір файлів чисел у форматі *.wav описаний в додатку.

По замовчуванню значення числа з вершини математичного стеку відтворюється з вказанням знаку числа та з точністю 1 цифра після десяткової коми. Відтворення знаку числа можна відключити за допомогою слова NOAUTOSAYPLUS та повторно включити за допомогою слова AUTOSAYPLUS.

Для припинення відтворення звукового повідомлення або числа з вершини математичного стеку існує слово MUTE, яке негайно припиняє відтворення звуку та блокує виконання заданого після слів PLAY або SAY слова.

При створенні програми з використанням голосових повідомлень, для їх більш якісного відтворення, необхідно дотримуватися певних засад, а саме: розбивати задачу на дрібніші фрагменти, які мають виконуватись з невеликими часовими проміжками між ними. Також, з цією самою метою, форт-система *автоматично блокує* виведення інформації на дисплей та реакцію на клавіатуру під час відтворення будь-якого голосового фрагменту.

Керування викликом

Для визначення статусу голосового виклику існує слово HOOK? (гак телефонної слухавки), яке кладе на стек логічне значення, що відповідає стану голосового виклику. При цьому, логічне значення ІСТИНА означає, що "слухавка лежить на апараті" - тобто, на даний момент не здійснюється жодного голосового виклику, а НЕ ІСТИНА - що "слухавка знята" - тобто, відбувається голосовий виклик (вхідний або вихідний).

Для припинення голосового виклику, існує слово HOLD (припинити) яке припиняє будь-який голосовий виклик (чи то вхідний, чи то вихідний).

Здійснення виклику

Для здійснення голосового виклику, існує слово DIAL (набрати номер) яке працює наступним чином: з вихідного буфера зчитується рядок - номер телефону у міжнародному або національному форматі, з вхідного тексту вибирається два чергових слова зворотнього виклику (введених після слова DIAL) і запускається процес набору номера телефону, а на стек кладеться логічне значення - результат виконання слова DIAL. При цьому, логічне значення ІСТИНА означає, що процес дозвону успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або вже здійснюється виклик). Коли на другому кінці телефонної лінії буде знята трубка, форт-система виконає перше з вказаних слів зворотнього виклику, а у випадку, коли ніхто не відповість на протязі 60 секунд або виклик буде перервано абонентом, форт-система виконає друге з вказаних слів зворотнього виклику.

Тепер ми можемо привести приклад здійснення голосового виклику:

```
> : повід_темп 1 AI? 2.345 F* SAY HOLD DROP ;
```

```

(OK)
> : привітатися ." hello.wav " PLAY повід_темп DROP ;
(OK)
> : подзвонити ." +380501234567 " DIAL привітатися STOP DROP ;
(OK)
> подзвонити
+380501234567 (OK)
>

```

Спочатку ми визначили нове слово "повід_темп", яке виконує наступні дії: на вершину математичного стеку кладеться значення вимірює на першому аналоговому вході, яке множиться на масштабний коефіцієнт - таким чином на вершині математичного стеку буде знаходитись значення, наприклад, температури, яке потрібне для виконання слова SAY. Далі виконується слово SAY - починається відтворення словами значення числа, в кінці якого буде виконано слово HOLD - покласти трубку, а відразу після слова SAY буде виконано слово DROP - щоб відкинути зі стеку даних результат виконання слова SAY, оскільки ми його не аналізуємо. Далі, ми визначили нове слово "привітатися", яке виконує наступні дії: у вихідний буфер друкується назва файлу, яка потрібна для виконання слова PLAY, далі виконується слово PLAY - починається відтворення, в кінці якого буде виконано слово "повід_темп", а відразу після слова PLAY буде виконано слово DROP - щоб відкинути зі стеку результат виконання слова PLAY, оскільки ми його не аналізуємо. Далі ми визначили нове слово "подзвонити", яке виконує наступні дії: у вихідний буфер друкується номер телефону, який потрібний для виконання слова DIAL, далі виконується слово DIAL - починається процес дозвону, в кінці якого буде виконано або слово "привітатися" - у випадку, якщо трубку знімуть, або слово STOP - у випадку, якщо трубку не знімуть в межах 60 секунд, а відразу після слова DIAL буде виконано слово DROP - щоб відкинути зі стеку результат виконання слова DIAL, оскільки ми його не аналізуємо. Далі ми просто виконуємо слово "подзвонити".

Прийом виклику

Для прийому голосового виклику, існує слово ANSWER (відповісти) яке працює наступним чином: з вихідного буфера зчитується один або кілька рядків розділених пробілом - номери телефонів у міжнародному або національному форматі, з вхідного тексту вибирається чергове слово зворотнього виклику (введене після слова ANSWER) і здійснюється спроба відповісти на виклик з вказаних номерів телефонів, а на стек кладеться логічне значення - результат виконання слова ANSWER. При цьому, логічне значення ІСТИНА означає, що спроба успішно завершилася і встановлено голосове з'єднання, а НЕ ІСТИНА - що на момент виконання слова ANSWER не було вхідного виклику, голосовий виклик вже триває або вхідний виклик здійснено з телефону, якого не було у списку. Коли спроба відповіді успішно завершилася, форт-система виконає вказане після слова ANSWER слово зворотнього виклику. Допустимо також не вказувати жодних номерів телефонів, тоді відповідь буде здійснена на любий вхідний виклик, що існує на момент виконання слова ANSWER. Приведемо приклад:

```

> : привітатися ." hello.wav " PLAY HOLD DROP ;
(OK)
> : чекати ." +380501234567 +380507654321
ANSWER привітатися
NOT IF 3.0 3 TIMER! чекати THEN ;
(OK)
> чекати
+380501234567 +380507654321 (OK)
>

```

Слово "привітатися" ми вже зустрічали. Далі ми визначили нове слово "чекати", яке виконує наступні дії: у вихідний буфер друкуються номери телефонів через пробіл, які потрібні для виконання слова ANSWER, далі виконується слово ANSWER - робиться спроба відповісти на можливий вхідний виклик, в кінці якої, у випадку наявності виклику з вказаних телефонів, буде виконано слово "привітатися", а відразу після слова ANSWER буде здійснено аналіз виконання слова ANSWER - якщо результат виконання буде НЕ ІСТИНА, то через 3 секунди знову буде виконано слово "чекати", а якщо ІСТИНА, то нічого не буде виконано. Далі ми просто виконуємо слово "чекати", запускаючи процес очікування вхідного виклику аж до моменту його успішного прийому.

Іноді виникає необхідність виконати якісь дії у відповідь лише на факт існування вхідного голосового виклику без з'єднання. Такий спосіб керування звичайно дуже обмежений, але є безкоштовним. Для прийому голосового виклику без з'єднання, існує слово CLIP (аббревіатура, яка означає ідентифікацію по вхідному виклику) яке працює наступним чином: з вхідного буфера зчитується один або кілька рядків розділених пробілом - номери телефонів у міжнародному або національному форматі, з вхідного тексту вибирається чергове слово (введене після слова CLIP) і здійснюється аналіз можливого вхідного виклику на предмет відповідності списку номерів телефонів, а на стек кладеться логічне значення - результат виконання слова CLIP. При цьому, логічне значення ІСТИНА означає, що аналіз успішний і голосове з'єднання *розірвано*, а НЕ ІСТИНА - що, або на момент виконання слова CLIP не було вхідного виклику, голосовий виклик вже триває або вхідний виклик здійснено з телефону, якого не було у списку. Коли аналіз успішно завершився, форт-система виконає вказане після слова CLIP слово. Допустимо також не вказувати жодних номерів телефонів, тоді вказане слово буде виконано у відповідь на любий вхідний виклик, що існує на момент виконання слова CLIP. Наведемо приклад:

```
> : пер_бойлер 2 RO? NOT 2 RO! ;  
(OK)  
> : чекати ." +380501234567 +380507654321  
CLIP пер_бойлер  
NOT IF 3.0 3 TIMER! чекати THEN ;  
(OK)  
> чекати  
+380501234567 +380507654321 (OK)  
>
```

Список телефонів, на які відбувається спроба відповісти словом ANSWER або CLIP, обмежується довжиною вихідного буфера. Проте існує інший спосіб, який дозволяє суттєво збільшити кількість номерів телефонів, з яких очікуються вхідні виклики. Наведемо приклад зі словом ANSWER:

```
> : hello ." hello.wav " PLAY HOLD DROP ;  
(OK)  
> : wait1 ." 80501234567 " ANSWER hello DROP 3.0 1 TIMER! wait1 ;  
(OK)  
> : wait2 ." +380687654321 " ANSWER hello DROP 3.0 2 TIMER! wait2 ;  
(OK)  
> wait1 wait2  
80501234567 +380687654321 (OK)  
>
```

Слово hello аналогічне слову привітатися з попереднього прикладу. Далі ми визначили два нових слова wait1 та wait2, які виконують подібні дії: у вихідний буфер друкується номер телефону (свій для кожного слова), далі виконується слово ANSWER - робиться спроба відповісти на можливий вхідний виклик, в кінці якої, у випадку наявності виклику з вказаного телефону, буде виконано слово hello, а відразу після слова ANSWER буде виконано слово DROP - щоб відкинути зі стеку результат виконання слова ANSWER. Через 3 секунди слова wait1 та wait2 виконуються знову. Далі ми просто виконуємо слова wait1 та wait2, запускаючи нескінченний процес очікування вхідного виклику (навіть під час існування голосового з'єднання - засади роботи слова ANSWER передбачають і таку можливість).

DTMF сигнали

При створенні голосових меню, необхідним елементом таких меню є програмування дій на натискання клавіш мобільного телефону - тобто встановлення реакції на DTMF сигнали. Для цього існує три базових слова очікування: WAITKEY (чекати клавішу), яке налаштовує необхідну реакцію форт-системи на поодинокі DTMF сигнали, WAITPW (чекати пароль), яке налаштовує необхідну реакцію форт-системи на введення за допомогою DTMF сигналів загальносистемного пароля доступу та WAITSTR (чекати рядок), яке налаштовує необхідну реакцію форт-системи на введення за допомогою DTMF сигналів рядка цифр, причому рядок цифр може представляти як ціле число так і число з десятковою комою.

В якості керуючих голосовим меню клавіш мобільного телефону, для користувачів доступні клавіші цифр від "1" до "9" а також клавіші "0", "*" та "#". В форт-системі їм відповідають коди від 1 до 12 (тобто, клавішам від "1" до "9" відповідає код від 1 до 9, клавіші "0" - 10, клавіші "*" - 11, а клавіші "#" - 12). По замовчуванню, кожне натискання клавіші супроводжується відповідним сигналом у динаміку мобільного телефону - це означає, що форт-система сприйняла факт натискання. Таку поведінку системи можна вимкнути за допомогою слова NODTMFCONFIRM (вимкнути підтвердження DTMF сигналів) або повторно увімкнути за допомогою слова DTMFCONFIRM (увімкнути підтвердження DTMF сигналів), причому дана опція не зберігається при вимкненому живленні. Необхідність у вимкненні підтвердження DTMF сигналів може виникнути у випадку проблем з передачею сигналів у деяких операторів зв'язку та у випадку підвищеної секретності при наборі команд або кодів.

Також можна у довільний момент часу існування голосового з'єднання згенерувати у динаміку мобільного телефону довільний тон відповідної клавіші за допомогою слова TONE. Слово TONE працює наступним чином: зі стеку знімається верхнє значення - код клавіші в діапазоні від 1 до 12 і у динаміку мобільного телефону відбувається генерація одиночного DTMF сигналу з тоном, який відповідає коду клавіші.

Розглянемо роботу базових слів очікування. Слово WAITKEY (чекати клавішу) працює наступним чином: з вхідного тексту вибирається три чергових слова зворотнього виклику (введених після слова WAITKEY) і налаштовується реакція форт-системи на DTMF сигнали, а на стек кладеться логічне значення - результат виконання слова WAITKEY. При цьому, логічне значення ІСТИНА означає, що процес налаштування пройшов успішно, а НЕ ІСТИНА - що виникли проблеми (наприклад, на даний момент не здійснюється жодний голосовий виклик). Коли на другому кінці телефонної лінії буде натиснена одну з клавіш мобільного телефону, форт-система виконає перше з вказаних слів зворотнього виклику, причому, на момент виконання даного слова на вершині стеку даних буде знаходитися код останнього отриманого DTMF сигналу - тобто код натисненої клавіші; у випадку, коли нічого не буде натиснуто на протязі 60 секунд, форт-система виконає друге з вказаних слів зворотнього виклику; і, нарешті, коли зв'язок буде перервано користувачем, форт-система виконає третє з вказаних слів зворотнього виклику. Наведемо приклад простого голосового меню, яке складається з одного пункту:

```

> : аналіз_кнопки 1 = IF ." true.wav " PLAY HOLD
ELSE ." by.wav " PLAY HOLD THEN DROP ;
(OK)
> : чекати_кнопку WAITKEY аналіз_кнопки HOLD STOP DROP 2 TONE ;
(OK)
> : привітатися ." hello.wav " PLAY чекати_кнопку DROP ;
(OK)
> : голосове_меню ." +380501234567 " DIAL привітатися STOP DROP ;
(OK)
> голосове_меню
+380501234567 (OK)
>

```

Ми визначили нове слово "аналіз_кнопки", яке виконує наступні дії: порівнює код останнього отриманого DTMF сигналу з 1, якщо в результаті порівняння на стеку буде логічне значення ІСТИНА, то відтворюється повідомлення з файлу "true.wav" і зв'язок обривається, інакше відтворюється повідомлення з файлу "by.wav" і зв'язок також обривається, в кожному випадку, буде також виконано слово DROP - щоб відкинути зі стеку результат виконання будь-якого зі слів PLAY, оскільки ми його не аналізуємо. Далі ми визначили нове слово "чекати_кнопку", яке налаштовує необхідну реакцію форт-системи на DTMF сигнали та генерує DTMF сигнал запрошення: при натисканні будь-якої кнопки, буде виконано слово "аналіз_кнопки", у випадку, коли пройде встановлений час реакції 60 секунд, зв'язок буде розірвано форт-системою, а у випадку, коли зв'язок буде розірвано користувачем голосового меню, не буде жодної реакції. Далі ми визначили нове слово "привітатися", яке відтворює повідомлення з файлу "hello.wav", після чого виконує слово "чекати_кнопку". Далі було визначено головне слово - "голосове_меню", яке здійснює голосовий дзвінок на вказаний телефон та, в разі успіху, виконує слово "привітатися". Виконання головного слова в наступному рядку розпочинає процес відтворення голосового меню.

Слово WAITPW (чекати пароль) працює подібним до слова WAITKEY чином: з вхідного тексту вибирається три чергових слова зворотнього виклику (введених після слова WAITPW) і налаштовується реакція форт-системи на введення загальносистемного пароля доступу за допомогою DTMF сигналів, а на стек кладеться логічне значення - результат виконання слова WAITPW. При цьому, логічне значення ІСТИНА означає, що процес налаштування пройшов успішно, а НЕ ІСТИНА - що виникли проблеми (наприклад, на даний момент не здійснюється жодний голосовий виклик). Коли на другому кінці телефонної лінії буде введено вірний пароль (при цьому кожна наступну цифру належить вводити тільки після сигналу запрошення і завершувати введення кнопкою "#"), форт-система виконає перше з вказаних слів зворотнього виклику; у випадку, коли нічого не буде натиснуто на протязі 60 секунд або пароль не вірний, форт-система виконає друге з вказаних слів зворотнього виклику; і, нарешті, коли зв'язок буде перервано користувачем, форт-система виконає третє з вказаних слів зворотнього виклику. Змінімо попередній приклад простого голосового меню, ввівши перевірку пароля (нововведення помічені темно-сірим кольором):

```

> : аналіз_кнопки 1 = IF ." true.wav " PLAY HOLD
ELSE ." by.wav " PLAY HOLD THEN DROP ;
(OK)
> : чекати_кнопку WAITKEY аналіз_кнопки HOLD STOP DROP 2 TONE ;
(OK)
> : привітатися ." hello.wav " PLAY чекати_кнопку DROP ;
(OK)
> : чекати_пароль WAITPW привітатися HOLD STOP DROP 6 TONE ;

```

```

(OK)
> : пароль ." parol.wav " PLAY чекати_пароль DROP ;
(OK)
> : голосове_меню ." +380501234567 " DIAL пароль STOP DROP ;
(OK)
> голосове_меню
+380501234567 (OK)
>

```

Слово WAITSTR (чекати рядок) також працює подібним до слова WAITKEY чином: з вхідного тексту вибирається три чергових слова зворотнього виклику (введених після слова WAITSTR) і налаштовується реакція форт-системи на введення рядка цифр за допомогою DTMF сигналів, а на стек кладеться логічне значення - результат виконання слова WAITSTR. При цьому, логічне значення ІСТИНА означає, що процес налаштування пройшов успішно, а НЕ ІСТИНА - що виникли проблеми (наприклад, на даний момент не здійснюється жодний голосовий виклик). Коли на другому кінці телефонної лінії буде введено рядок цифр, форт-система виконає перше з вказаних слів зворотнього виклику, причому, на момент виконання даного слова, введене число буде знаходитися на вершині математичного стеку; у випадку, коли нічого не буде натиснуто на протязі 60 секунд, форт-система виконає друге з вказаних слів зворотнього виклику; і, нарешті, коли зв'язок буде перервано користувачем, форт-система виконає третє з вказаних слів зворотнього виклику. При введенні рядка цифр, кожен наступну цифру належить вводити тільки після сигналу успішного введення попередньої цифри, для введення десяткової коми використовується кнопка "*", а завершувати введення необхідно кнопкою "#". Наведемо приклад простого голосового меню введення значення температури в математичну змінну:

```

> : встан_темпер FDUP 60.0 F<
IF 1 FVAR! ." pidtvr.wav " PLAY HOLD DROP
ELSE ." nevirno.wav " PLAY HOLD DROP
THEN;
(OK)
> : чекати_темпер WAITSTR встан_темпер HOLD STOP DROP 11 TONE ;
(OK)
> : привітатися ." zaprosh.wav " PLAY чекати_темпер DROP ;
(OK)
> : голосове_меню ." +380501234567 " DIAL привітатися STOP DROP ;
(OK)
> голосове_меню
+380501234567 (OK)
>

```

Ми визначили нове слово "встан_темпер", яке виконує наступні дії: робить копію покладеного на математичний стек числа, порівнює його значення з числом 60.0 (прибираючи при цьому з математичного стеку одну з копій), якщо в результаті порівняння на стеку даних буде покладено логічне значення ІСТИНА, то відбувається присвоєння значення температури математичній змінній номер 1 (при цьому з математичного стеку прибирається остання копія), відтворюється повідомлення з файлу "pidtvr.wav" і зв'язок обривається, інакше відтворюється повідомлення з файлу "nevirno.wav" і зв'язок також обривається. Далі ми визначили нове слово "чекати_темпер", яке налаштовує необхідну реакцію форт-системи на введення рядка цифр: при натисканні кнопки "#", буде виконано слово "встан_темпер", у випадку, коли пройде встановлений час реакції 60 секунд, зв'язок буде розірвано форт-системою, а у випадку, коли зв'язок буде розірвано користувачем голосового меню, не буде жодної

реакції. Далі ми визначили нове слово "привітатися", яке відтворює повідомлення з файлу "zaprosh.wav", після чого виконує слово "чекати_темп". Далі було визначено головне слово - "голосове_меню", яке здійснює голосовий дзвінок на вказаний телефон та, в разі успіху, виконує слово "привітатися". Виконання головного слова в наступному рядку розпочинає процес відтворення голосового меню.

Гарнітура

Для підключення зовнішньої гарнітури (комплекту обладнання для здійснення розмови) під час здійснення голосового виклику (вхідного чи вихідного) існує слово MIC (гарнітура). Виконання даного слова призводить до переключення аудіо каналу GSM модуля з цифрового синтезатора на зовнішню гарнітуру приєднану або до входів SPK, MIC+, MIC-, або до аудіо розніму MIC/EAR. При цьому на стек кладеться логічне значення - результат виконання слова MIC: логічне значення ІСТИНА означає, що процес переключення пройшов успішно, а НЕ ІСТИНА - що виникли проблеми (наприклад, на даний момент не здійснюється жодний голосовий виклик). При успішному виконанні слова MIC, відтворення голосових повідомлень стає неможливим - аудіо канал GSM модуля працюватиме із зовнішнім джерелом звуку а в зовнішньому динаміку буде відтворюватись аудіо з GSM модуля. Після того, як зв'язок буде перервано (користувачем чи форт-системою), аудіо канал GSM модуля автоматично переключається на цифровий синтезатор - тобто в наступному голосовому виклику знову стають доступними відтворення голосових повідомлень а аудіо з GSM модуля не буде чути в зовнішньому динаміку. Також аудіо канал GSM модуля можна переключити в будь-який момент часу за допомогою слова VOICE. Виконання даного слова призводить до переключення аудіо каналу GSM модуля на цифровий синтезатор. При цьому на стек кладеться логічне значення - результат виконання слова VOICE: логічне значення ІСТИНА означає, що процес переключення пройшов успішно, а НЕ ІСТИНА - що виникли проблеми (наприклад, на даний момент не здійснюється жодний голосовий виклик). Прийом DTMF сигналів можливий як при роботі GSM модуля від цифрового синтезатора, так і при роботі від зовнішньої гарнітури.

Для регулювання чутливості зовнішньої гарнітури існує слово MICLEVEL (чутливість гарнітури) яке працює наступним чином: з вершини стеку даних знімається число - нове умовне значення чутливості в діапазоні від 0 до 15 і встановлюється задана чутливість. Збільшення або зменшення умовної чутливості на 1 призводить до збільшення або зменшення реальної чутливості гарнітури в 1,1885 рази. По замовчуванню, умовна чутливість рівна 2 а її нове значення не зберігається при виключенні живлення.

Для регулювання гучності зовнішньої гарнітури існує слово SPKLEVEL (гучність гарнітури) яке працює наступним чином: з вершини стеку даних знімається число - нове умовне значення гучності в діапазоні від 0 до 100 і встановлюється задана гучність. Зменшення умовної гучності призводить до зменшення реальної гучності в динаміку і навпаки. По замовчуванню, умовна гучності рівна 100 а її нове значення не зберігається при виключенні живлення.

Передача даних CSD

Прийом та передача даних за допомогою каналу CSD відбувається по принципу модемного з'єднання точка-точка. На одному кінці каналу, при цьому, знаходиться контролер серії ES-ForthLogic™ з вбудованим модулем GSM. На другому кінці даного каналу може бути або звичайний аналоговий модем, який працює в провідних телефонних мережах, або будь-який GSM модем, який підтримує протокол CSD, або інший контролер серії ES-ForthLogic™ з вбудованим модулем GSM.

Аналогічно до того як із вхідними SMS, вхідний потік даних з каналу CSD безпосередньо потрапляє у вхідний буфер текстового інтерпретатора форт-системи і, відповідно, інтерпретується та виконується. Як уже згадувалося, дана концепція є надзвичайно гнучка, проте в даному випадку вона є потужніша

ніж аналогічна концепція вхідних SMS тому, що у вхідному потоці не заблоковано визначення нових слів через двокрапку. Це зроблено так тому, що користувачу системи доступитися до каналу CSD зі звичайного телефону без комп'ютера та спеціального програмного забезпечення практично неможливо, і відповідно імовірність випадково збурити роботу системи практично низька. Проте для систем типу M2M - міжмашинної взаємодії або систем моніторингу типу ACKOE така опція, як дистанційне визначення нових слів через двокрапку, може стати в нагоді.

Після отримання вхідного потоку даних за допомогою каналу CSD (довжина однієї порції даних не може перевищувати 160 символів), автоматично формується вихідний потік даних з відповіддю форт-системи - це *CSD режим* форт-системи. Відмінність цього режиму від режиму при роботі з терміналом (*термінального режиму*) полягає в тому, що не працює слово WORDS, а у вихідному потоці відсутні всі повідомлення про помилки та інформація в квадратних дужках генерована деякими словами (".S", ".FS" тощо), відсутні всі текстові повідомлення форт-системи в круглих дужках ("OK", "ERROR-...", тощо) а також відсутній символ запрошення форт-системи (">"). Це дозволяє в режимі on-line направляти відповідь однієї форт-системи безпосередньо на вхід іншої, що спрощує побудову систем типу M2M. Проте, в разі виникнення помилок, повідомлення про них висвітлюються в термінальному режимі за умови що існує активне підключення до терміналу.

З програми на мові ForthLogic™ можна ініціювати підключення до каналу CSD, відправити дані у вигляді рядка тексту, ініціювати виконання будь-якого відомого слова в разі отримання даних через канал, розірвати з'єднання та подивитися стан з'єднання. Для цього реалізовано низку слів які представлено далі по тексту.

Для здійснення підключення до каналу CSD існує слово CONNECTCSD (підключитись до каналу CSD) яке працює наступним чином: з вихідного буфера зчитується рядок - номер телефону у міжнародному або національному форматі, з вхідного тексту вибирається два чергових слова зворотнього виклику (введених після слова CONNECTCSD) і запускається процес набору номера телефону, а на стек кладеться логічне значення - результат виконання слова CONNECTCSD. При цьому, логічне значення ІСТИНА означає, що процес підключення успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або вже здійснюється інший вихідний або вхідний виклик). Коли успішно утвориться канал CSD, форт-система виконає перше з вказаних слів зворотнього виклику, в іншому випадку, коли канал не зможе утворитися на протязі 60 секунд або не зможе утворитися в принципі (спроба утворити канал не з модемом), форт-система виконає друге з вказаних слів зворотнього виклику.

Для передачі повідомлення в канал CSD існує слово SENDCSD (відправити дані в канал CSD) яке працює наступним чином: з вихідного буфера зчитується рядок повідомлення, з вхідного тексту вибирається чергове слово зворотнього виклику (введене після слова SENDCSD) і рядок передається через канал CSD, а на стек кладеться логічне значення - результат виконання слова SENDCSD. При цьому, логічне значення ІСТИНА означає, що процес передачі успішно розпочався, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або канал CSD закритий). В разі, коли у відповідь прийде повідомлення з другого кінця каналу CSD, воно потрапить спочатку у інтерпретатор форт-системи (див. вище по тексту), після чого форт-система виконає вказане після слова SENDCSD слово зворотнього виклику.

Для розриву каналу CSD в будь-який момент часу існує слово BREAKCSD (розірвати канал CSD), а для отримання статусу каналу існує слово STATUSCSD (стан каналу CSD), яке повертає на стек логічне значення ІСТИНА якщо канал передачі даних CSD відкритий і логічне значення НЕ ІСТИНА в протилежному випадку.

Наведемо приклад використання вищезгаданих слів для побудови системи типу M2M - міжмашинної взаємодії, в якій утворюється канал CSD між двома контролерами. Далі один з них (ГОЛОВНИЙ) посилає запити і обробляє відповіді таким чином, що стан цифрових входів іншого контролера

(ПІДПОРЯДКОВАНОГО) відображається на цифрових виходах ГОЛОВНОГО:

```
> 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 "
SEND CSD read DROP ;
(OK)
> ." send " FIND TO send_a
(OK)
> : start 1 USER CONNECT CSD send STOP DROP ;
(OK)
> : stop BREAK CSD ;
(OK)
>
```

Дана програма записується в пам'ять тільки одного контролера - ГОЛОВНОГО. Далі ГОЛОВНИЙ контролер ініціює з'єднання з ПІДПОРЯДКОВАНИМ і починає керований обмін станом цифрових входів. Розглянемо даний приклад детальніше.

Канал даних відкривається словом "start", як тільки це відбувається виконується слово "send", яке посилає запит "5 DI? . 6 DI? . 7 DI? . 8 DI? . 0.1 2400 BEEP". Згідно цього запиту, інший контролер - ПІДПОРЯДКОВАНИЙ виводить у свій вихідний буфер стан цифрових входів у вигляді цифр 0 або -1 та відтворює короткий звуковий сигнал. Далі весь вміст його вихідного буфера потрапляє у канал CSD в якості відповіді. Ця відповідь спочатку потрапляє у вхідний буфер ГОЛОВНОГО контролера, який інтерпретує цифри та кладе їх на свій стек. Після цього форт-системою виконується слово "read", яке знімає цифри зі стеку та записує їх у відповідні цифрові виходи. Далі воно відтворює короткий звуковий сигнал і знову виконує слово "send" (через механізм зворотнього посилення, див. розділ "Векторне виконання") - таким чином весь цикл запит-відповідь повторюється безмежну кількість разів.

Канал CSD можна використати для неконтрольованого з програми на мові ForthLogic™ обміну даними з пристроями приєднаними до послідовного інтерфейсу RS485. Фактично утворюється "прозорий" модемний канал між пристроями та програмою для ПЕОМ, яка ініціювала даний канал за допомогою звичайного модема або GSM-модема, і не накладаються жодні обмеження на протокол обміну по даному каналу. Проте, слід взяти до уваги часові затримки, що виникають з природи комутування в часі каналів передачі в мережах GSM. Практично, це полягає в тому, що в неперервному потоці даних можливі паузи до 20 мс. Також слід пам'ятати, що швидкість обміну фіксована та складає 9600 біт/сек.

Для утворення модемного каналу застосовується слово MODEM CSD. Дане слово можна застосовувати лише тоді, коли попередньо утворено канал передачі даних CSD: воно повертає на стек логічне значення ІСТИНА якщо модемний канал успішно утворився і логічне значення НЕ ІСТИНА в протилежному випадку. Практично, дане слово можна застосувати як в термінальному режимі (з програми на ForthLogic™ чи безпосередньо з терміналу) так і в CSD режимі форт-системи (наприклад, з програми для ПЕОМ, яка ініціює модемний канал).

Коли утворюється модемний канал, всі інші способи обміну через GSM (голосові повідомлення, SMS, GPRS) не працюють. Канал можна розірвати шляхом розірвання CSD з'єднання з протилежного кінця лінії. При цьому відновлюється робота через решту каналів GSM (голосові повідомлення, SMS, GPRS та CSD в звичайному режимі).

Передача даних GPRS

GPRS режим форт-системи

Прийом та передача даних за допомогою каналу GPRS відбувається по принципу клієнт-сервер з використанням протоколу TCP. При звичайному клієнт-серверному з'єднанні, над протоколом TCP виконується текстовий протокол обміну з форт-системою подібний до звичайного термінального режиму роботи: після отримання вхідного потоку даних (довжина однієї порції даних не може перевищувати 160 символів), автоматично формується вихідний потік даних з відповіддю форт-системи - це *GPRS режим* форт-системи. При цьому, у вихідному потоці відображаються всі помилки та службова інформація форт-системи що не прийнятно у випадку міжмашинної взаємодії (M2M) між двома контролерами. Для відключення відображення службової інформації та помилок існує слово M2MTCPON (включити режим M2M через протокол TCP). Повторно включити нормальний режим дозволяє слово M2MTCP OFF (виключити режим M2M через протокол TCP). При пропаданні живлення дана опція не зберігається, а при подачі живлення по замовчуванню встановлюється нормальний режим передачі даних через протокол TCP.

Засади клієнт-серверного з'єднання

Необхідно докладно зупинитися на технології клієнт-сервер. На одному кінці каналу завжди знаходиться сервер а на другому клієнт і тому існує два сценарію побудови з'єднання. Перший сценарій полягає в тому, що в якості сервера зручно використовувати серверну програму яка працює на ПЕОМ, яка має фіксовану IP- адресу у всесвітній мережі INTERNET. В цьому випадку контролер серії ES-ForthLogic™ з вбудованим модулем GSM виступає в якості клієнта, який ініціює з'єднання із сервером. Перевага такого сценарію полягає в тому, що до сервера одночасно може бути під'єднано дуже багато клієнтів (сотні та тисячі) і нема необхідності в отримванні спеціальних SIM-карт з доступними назовні IP-адресами.

Другий сценарій полягає в тому, що в якості сервера виступає контролер серії ES-ForthLogic™ з вбудованим модулем GSM який може одночасно прийняти лише *одне* клієнтське з'єднання. При цьому необхідно гарантувати доступність IP-адреси контролера у всесвітній мережі INTERNET - це досягається при застосуванні спеціальних SIM-карт та спеціальних телеметричних тарифів обраного оператора мережі GSM. Перевага даного сценарію полягає в тому, що на другому кінці можна використовувати клієнтську програму яка працює на ПЕОМ і має будь-який доступ до мережі INTERNET. Така програма має одночасно утворювати стільки клієнтських з'єднань з контролерами, скільки необхідно.

Слова для роботи з GPRS

З програми на мові ForthLogic™ можна ініціювати підключення до GPRS, встановити з'єднання в режимі клієнта або сервера, відправити дані у вигляді рядка тексту, розірвати з'єднання та подивитися стан з'єднання. Для цього реалізовано низку слів які представлено далі по тексту. Також реалізована підтримка передачі даних, які отримуються під час роботи реєстратора, через існуюче GPRS-з'єднання або по протоколу TCP/IP, або на сервер баз даних MySQL із застосуванням протоколу HTTP - більш докладно це описано у відповідному розділі. Крім роботи в режимі клієнта або сервера з іншим контролером або програмою яка працює на ПЕОМ, існує можливість підтримувати постійне з'єднання зі спеціалізованим OPC-сервером. При цьому застосовується перший сценарій технології клієнт-сервер (описаний вище).

Початкові налаштування

Для роботи з GPRS, передусім потрібна наявність цієї послуги у оператора мережі GSM на використовуваній SIM-карті. Для з'ясування можливості підключення до сервісу GPRS, існує слово STATUSGPRS (стан сервісу GPRS), яке повертає на стек логічне значення ІСТИНА, якщо сервіс GPRS для даної SIM-карти активовано. Якщо сервіс GPRS не активується по замовчуванню (властивість тарифу) або стався збій мережі, то для активації сервісу існує слово ATTACHGPRS (підключити сервіс GPRS) яке робить спробу підключитися до сервісу GPRS а на стек повертає логічне значення ІСТИНА, якщо спроба розпочалася вдало і НЕ ІСТИНА якщо виникли проблеми (наприклад, GSM модуль вимкнено). Процес підключення сервісу може займати до 10 секунд і перевіряється за допомогою слова STATUSGPRS.

Для зберігання параметрів каналу GPRS таких як APN та IP в енергонезалежній пам'яті існують спеціальні однойменні рядкові змінні. Довжина змінних IP та APN складає 70 символів. Для роботи з цими змінними існують слова IP, APN, SETIP та SETAPN. Слова IP та APN друкують у вихідному буфері та на терміналі вміст однойменних змінних. А слова SETIP та SETAPN запам'ятовують текст з вихідного буфера у відповідних змінних IP та APN. В залежності від режиму роботи каналу GPRS, семантика вмісту цих змінних може змінюватись.

Для зберігання параметрів роботи зі спеціалізованим OPC-сервером або сервером баз даних MySQL, в енергонезалежній пам'яті також існують спеціальні рядкові змінні ID (ідентифікатор) та URL (адреса). Змінна ID призначена для зберігання імені та пароля для доступу до віддаленого сервера. Ім'я та пароль мають бути розділеними пробілом і мати довжину не більше ніж 15 символів. Змінна URL призначена для встановлення адреси всередині хосту, до якої відбувається звертання із застосуванням протоколу HTTP. Довжина змінної ID складає 30 символів а змінної URL - 70 символів. Для роботи з цими змінними існують слова ID, URL, SETID та SETURL. Слова ID та URL друкують у вихідному буфері та на терміналі вміст однойменних змінних. А слова SETID та SETURL запам'ятовують текст з вихідного буфера у відповідних змінних ID та URL.

Для конфігурування протоколу роботи TCP та підключення до GPRS існує слово CONFIGTCP (конфігурування протоколу TCP) яке використовується разом із системними константами CLIENT (клієнт), CLIENTDB (клієнт сервера баз даних MySQL), CLIENTSQL (поточковий клієнт сервера баз даних MySQL), CLIENTOPC (клієнт OPC-сервера) та SERVER (сервер) і працює наступним чином: зі стеку знімається число, яке відповідає одній із вищезгаданих системних констант, з вихідного буфера зчитується рядок який складається з APN, імені та пароля розділених пробілами і відбувається налаштування протоколу роботи TCP та підключення до GPRS. На стек при цьому кладеться логічне значення - результат виконання слова CONFIGTCP: логічне значення ІСТИНА означає, що процес налаштування та підключення здійснився успішно, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або відсутній сервіс GPRS). APN який вказується у вихідному буфері, надається оператором для входження у мережу GPRS. Іноді також надається ім'я та пароль, проте, якщо вони відсутні, то рядок складається тільки з APN.

Для встановлення номеру порту, який буде "прослуховуватись" в режимі роботи типу SERVER, існує слово CONFIGPORT, яке працює наступним чином: зі стеку знімається число в діапазоні від 1 до 65535 і відбувається встановлення номеру порту. На стек при цьому кладеться логічне значення - результат виконання слова CONFIGPORT: логічне значення ІСТИНА означає, що процес встановлення здійснився успішно, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або відсутній сервіс GPRS). Номер порту по замовчуванню завжди дорівнює 2020 і його налаштування має зміст тільки для роботи в режимі SERVER. При пропаданні живлення номер порту не зберігається.

До ресурсів в мережі INTERNET можна звертатися через їхні IP-адреси (наприклад 124.34.1.145) або через призначені для них URL-адреси (наприклад: www.es.ua). По замовчуванню встановлений режим IP-адрес - так зручніше для телеметричних систем. Проте, в таких задачах як робота з сервером баз

даних MySQL виникає необхідність вказувати URL-адреси. Для встановлення режиму адресації за допомогою URL-адрес існує слово DNSMODE (режим імен), яке працює наступним чином: з вихідного буфера зчитується рядок, який складається з двох IP-адрес DNS-серверів розділених пробілами і відбувається встановлення режиму URL-адрес. На стек при цьому повертається результат виконання слова DNSMODE: логічне значення ІСТИНА означає, що процес встановлення здійснився успішно, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або відсутній сервіс GPRS). Допускається не вказувати IP-адрес DNS-серверів, тоді для пошуку URL-адрес система звертається до стандартних відкритих DNS-серверів 208.67.220.220 та 208.67.222.222. Для встановлення режиму адресації за допомогою IP-адрес призначено слово IPMODE (режим IP-адрес) яке повертає на стек логічне значення - результат свого виконання: логічне значення ІСТИНА означає, що процес встановлення здійснився успішно, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або відсутній сервіс GPRS). При пропаданні живлення опція режиму адресації не зберігається.

Встановлення та розрив з'єднання

Для встановлення з'єднання згідно протоколу TCP існує слово OPENTCP (встановити з'єднання по протоколу TCP), яке працює наступним чином: якщо протокол був налаштований на роботу в режимі CLIENT або CLIENTOPC, то з вихідного буфера зчитується рядок який складається з IP-адреси та номеру порту розділених пробілом, якщо протокол був налаштований на роботу в режимі CLIENTSQL, то з вихідного буфера нічого не зчитується, а параметри IP-адреси та номеру порту зберігаються в змінній IP в енергонезалежній пам'яті, з вхідного тексту вибирається два чергових слова (введених після слова OPENTCP) і відбувається встановлення з'єднання. На стек при цьому кладеться логічне значення - результат виконання слова OPENTCP: логічне значення ІСТИНА означає, що процес з'єднання розпочався успішно, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено або відсутній сервіс GPRS). Коли успішно встановиться з'єднання згідно протоколу TCP, форт-система виконає перше з вказаних після слова OPENTCP слів, в іншому випадку, коли з'єднання не зможе утворитися на протязі 120 секунд, форт-система виконає друге з вказаних слів. Адреси у вихідному буфері вказуються тільки для режимів з'єднання типу CLIENT або CLIENTOPC і є адресами сервера з яким відбувається спроба з'єднання. Для режиму з'єднання типу SERVER дане слово лише запускає сервер, який очікує з'єднання з боку клієнта.

Для передачі повідомлення згідно протоколу TCP в будь-якому звичайному режимі з'єднання типу CLIENT або SERVER, існує слово SENDTCP (відправити дані по протоколу TCP) яке працює наступним чином: з вихідного буфера зчитується рядок повідомлення і передається по протоколу TCP, а на стек кладеться логічне значення - результат виконання слова SENDTCP. При цьому, логічне значення ІСТИНА означає, що процес передачі успішно здійснився, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, GSM модуль вимкнено, відсутній сервіс GPRS або канал GPRS закритий). Максимальна довжина повідомлення складає 160 символів.

Передача повідомлень в режимі з'єднання типу CLIENTOPC є неможливою, оскільки обмін даними з OPC-сервером відбувається по спеціалізованому внутрішньому протоколу. OPC-сервер робить запити а система автоматично та прозоро для користувача здійснює обробку та формування відповіді. Передача повідомлень в режимах з'єднання типу CLIENTDB або CLIENTSQL також є неможливою, оскільки дані передаються автоматично з внутрішньої енергонезалежної або оперативної пам'яті на сервер баз даних MySQL із застосуванням протоколу HTTP. Даний процес розпочинається словом LOG>DB або LOG>SQL і повністю контролюється системою.

Для розриву з'єднання типу CLIENT, CLIENTSQL або CLIENTOPC в будь-який момент часу існує слово CLOSETCP (розірвати з'єднання згідно протоколу TCP). Після розриву з'єднання типу CLIENT, CLIENTSQL або CLIENTOPC, можна здійснити спробу з'єднання з іншим сервером.

Для відключення від GPRS в будь-який момент часу існує слово SHUTTCP (розірвати підключення). Після відключення від GPRS, необхідно знову підключатися за допомогою слова CONFIGTCP.

Статус з'єднання

Для отримання статусу з'єднання згідно протоколу TCP існує слово STATUSTCP (стан з'єднання згідно протоколу TCP), яке повертає на стек логічне значення ІСТИНА якщо в режимі з'єднання типу CLIENT, CLIENTDB, CLIENTSQL або CLIENTOPC існує фізичне з'єднання з віддаленим сервером.

В режимі з'єднання типу SERVER слово STATUSTCP повертає на стек логічне значення ІСТИНА якщо сам сервер є активним (при цьому може бути відсутнє саме з'єднання з клієнтом). Для отримання статусу з'єднання з клієнтом в режимі з'єднання типу SERVER існує слово STATUSSERVER (стан серверу), яке повертає на стек логічне значення ІСТИНА якщо в режимі з'єднання типу SERVER існує фізичне з'єднання з віддаленим клієнтом.

Для отримання статусу роботи з OPC-сервером в режимі з'єднання типу CLIENTOPC існує слово STATUSOPC (стан OPC), яке повертає на стек логічне значення ІСТИНА якщо в режимі з'єднання типу CLIENTOPC існує логічне з'єднання з OPC-сервером. Логічне з'єднання з OPC-сервером означає постійний автоматичний обмін даними по спеціалізованому внутрішньому протоколу.

Після підключення до GPRS, можна дізнатися про власну IP-адресу, яку автоматично надає оператор, за допомогою слова LOCALIP (власна IP-адреса). Дане слово друкує у вихідний буфер та на термінал рядок власної IP-адреси.

Якщо в режимі з'єднання типу SERVER існує фізичне з'єднання з віддаленим клієнтом, то можна дізнатися про IP-адресу клієнта за допомогою слова REMOTEIP (IP-адреса клієнта). Дане слово друкує у вихідний буфер та на термінал рядок IP-адреси клієнта. Також в режимі з'єднання типу SERVER можна обмежити коло віддалених клієнтів, якщо записати у енергонезалежну змінну IP IP-адресу клієнта якому дозволено підключатися до сервера. Якщо у цю змінну записати пустий рядок, то до сервера можуть підключатися всі клієнти без обмежень.

Особливості режиму CLIENTOPC

Як уже згадувалось, в режимі CLIENTOPC обмін даними з OPC-сервером відбувається автоматично по спеціалізованому внутрішньому протоколу. OPC-сервер це спеціалізований сервер який з одного боку здійснює обмін даними з контролером через канал GPRS а з другого боку надає доступ до цих даних OPC-клієнтам, в якості яких найчастіше виступають різноманітні системи дистанційного керування та моніторингу технологічних процесів (АСУТП, SCADA). OPC-сервер підтримує специфікації OPC DA v1 та v2 та мусить працювати на ПЕОМ, яка має фіксовану IP-адресу у всесвітній мережі INTERNET. Під час роботи з OPC-сервером контролер виступає в якості клієнта і тому можна застосувати звичайні SIM-карти з підтримкою GPRS-з'єднання. Також можна застосовувати SIM-карти з підтримкою VPN. Ініціатором з'єднання з OPC-сервером виступає контролер. Для успішного з'єднання, контролер має пройти аутентифікацію, яка полягає в співпадині імені та пароля, які задаються за допомогою змінної ID. Одночасно до одного OPC-сервера може бути під'єднано багато контролерів, що дозволяє будувати розподілені системи дистанційного керування та моніторингу технологічних процесів.

Принцип обміну даними з OPC-сервером полягає в тому, що OPC-сервер робить запити а система автоматично та прозоро для користувача здійснює обробку запиту та формування відповіді. При цьому OPC-сервер може вільно встановлювати та читати всі глобальні змінні, глобальні змінні в форматі чисел з плаваючою комою та глобальні бітові змінні. Програма на мові ForthLogic™ не має безпосереднього доступу до обміну даними з OPC-сервером, проте може взаємодіяти з OPC-сервером (а через нього і з системою АСУТП чи SCADA) через глобальні змінні.

Особливості режимів CLIENTDB та CLIENTSQL

В режимі з'єднання типу CLIENTDB або CLIENTSQL дані передаються автоматично з внутрішньої енергонезалежної або оперативної пам'яті на сервер баз даних MySQL із застосуванням протоколу HTTP. Як уже згадувалося, даний процес розпочинається словом LOG>DB або LOG>SQL і повністю контролюється системою.

Принцип передачі даних на сервер баз даних MySQL полягає в тому, що система спочатку передає файл з внутрішньої енергонезалежної або оперативної пам'яті на HTTP-сервер за допомогою методу POST. Цей файл приймається та обробляється спеціалізованим скриптом, ім'я якого необхідно вказувати через змінну URL. Далі цей файл аналізується та дані з нього передаються в таблицю на сервер баз даних MySQL. Відмінність режимів з'єднань типу CLIENTDB від CLIENTSQL полягає в тому, що в режимі типу CLIENTDB HTTP-сервер після отримання файлу закриває фізичне з'єднання з контролером а в режимі CLIENTSQL - ні. По цій причині з'єднання типу CLIENTSQL можна застосовувати для більш інтенсивних режимів реєстрації та передачі даних. Рекомендований інтервал передачі даних на сервер в режимі CLIENTSQL складає від 30 секунд і вище.

В режимі з'єднання типу CLIENTDB або CLIENTSQL контролер виступає в якості клієнта і тому можна застосувати звичайні SIM-карти з підтримкою GPRS-з'єднання. Для успішного з'єднання з HTTP-сервером, контролер має пройти аутентифікацію, яка полягає в співпадині імені та пароля, які задаються за допомогою змінної ID. Крім того, програма на мові ForthLogic™ має забезпечити вірний формат та фіксовану кількість полів зареєстрованих даних для одного запису в файл реєстрації, наприклад:

"1304585524;22.233455;-1;13434", де:

1304585524	22.233455	-1	13434
Поле часу в форматі UTC	Поле числа з плаваючою комою, наприклад температура	Поле прапорця, наприклад стан входу	Поле цілого числа, наприклад кількість імпульсів на вході

Розділювачем між полями даних має бути символ ";" (крапка з комою), в кінці запису розділювач не застосовується. Першим полем має бути час в форматі UTC. Перше поле є обов'язковим. Решта полів та їх типи мають строго відповідати параметрам відповідної таблиці на сервері баз даних MySQL.

В режимі з'єднання типу CLIENTDB або CLIENTSQL контролер також може отримувати зворотню інформацію в кінці кожної сесії передачі файлу на HTTP-сервер. Дана інформація обробляється системою автоматично. При цьому HTTP-сервер може вільно встановлювати та читати всі глобальні змінні, глобальні змінні в форматі чисел з плаваючою комою та глобальні бітові змінні. Програма на мові ForthLogic™ може скористатися даними в цих змінних і в цей спосіб реалізувати "зворотній зв'язок" з HTTP-сервером.

Приклади з'єднання туну SERVER та CLIENT

Наведемо приклад встановлення з'єднання типу SERVER. В даному прикладі використовується SIM-карта з фіксованою IP-адресою та порт сервера по замовчуванню (2020). Клієнт має встановлювати з'єднання з даною адресою та портом. Крім того, клієнт може бути тільки один і також має мати фіксовану IP-адресу.

```
> ." 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)
>

```

Спочатку ми налаштували параметри GPRS-сервісу та дозволили доступ тільки для клієнта з IP-адресою 192.168.0.1. Слово checknet перевіряє стан GSM-модуля, коли він включиться та успішно зареєструється в мережі GSM, буде виконано слово checkservice, яке перевіряє наявність GPRS-сервісу та здійснює спробу підключитися до сервісу в разі його відсутності. У випадку його наявності відбувається спроба підключити та налаштувати GPRS в режимі сервера. При успішній спробі буде виконано слово start. При невдалій спробі буде виконано слово reconfig, яке відключає GPRS та повторює спробу шляхом виконання слова checkservice. Слово start активує сервер та в разі його успішної активації запускає на виконання слово task, інакше все повторюється шляхом виконання слова checkservice. Слово task неперервно перевіряє стан серверу і якщо він відключився (пропала мережа, тощо) то буде виконано слово reconfig, яке спробує відновити роботу сервера.

Тепер наведемо приклад встановлення з'єднання типу 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)
>

```

Спочатку ми налаштували параметри GPRS-сервісу та вказали параметри доступу до сервера. Решта слів практично аналогічні до попереднього прикладу. Лише слово `start` здійснює спробу підключитися до сервера, а слово `checkservice` налаштовує режим клієнта.

Після встановлення з'єднання в будь-якому звичайному режимі `CLIENT` або `SERVER` форт-система працюватиме в GPRS-режимі і можна відсилати з програми рядкові дані або дані реєстрації (дані реєстрації можуть відсилатися або в процесі нагромадження або у вигляді файлу збереженого у внутрішній пам'яті - див. розділ "Реєстратор").

Модуль ETHERNET

Деякі версії контролерів містять вбудований порт ETHERNET і можуть здійснювати обмін даними через локальні мережі ETHERNET або через INTERNET. На даний момент підтримується робота по наступним протоколам: сервер MODBUS TCP в режимі підпорядкованого пристрою (SLAVE), клієнт передачі даних на сервер баз даних MySQL із застосуванням протоколу HTTP.

Засади клієнт-серверного з'єднання *ETHERNET*

Необхідно докладно зупинитися на технології клієнт-сервер при передачі даних через локальні мережі *ETHERNET* або через *INTERNET*. На одному кінці каналу завжди знаходиться сервер а на другому клієнт і тому існує два сценарію побудови з'єднання. Перший сценарій полягає в тому, що в якості сервера зручно використовувати серверну програму яка працює на ПЕОМ або на спеціалізованому серверному обладнанні, яке має фіксовану IP-адресу у всесвітній мережі *INTERNET*. В цьому випадку контролер серії ES-ForthLogic™ з вбудованим портом *ETHERNET* виступає в якості клієнта, який ініціює з'єднання із сервером. Перевага такого сценарію полягає в тому, що до сервера одночасно може бути під'єднано дуже багато клієнтів (сотні та тисячі) і для них нема необхідності в отримванні статичних IP-адрес.

Другий сценарій полягає в тому, що в якості сервера виступає контролер серії ES-ForthLogic™ з вбудованим портом *ETHERNET* який може одночасно обробити лише обмежену кількість клієнтських з'єднань. При цьому необхідно гарантувати доступність IP-адреси контролера у всесвітній мережі *INTERNET* - це досягається при застосуванні статичної IP-адреси. Звичайно, таку послугу замовляють в інтернет-провайдерів. Крім цього, практично завжди виникає необхідність здійснити налаштування локального мережевого обладнання для здійснення перенаправлення портів. Перевага даного сценарію полягає в тому, що на другому кінці можна використовувати клієнтські програми які працюють на ПЕОМ або іншому обладнанні (планшет, смартфон, тощо) і мають будь-який доступ до мережі *INTERNET*.

Слова для роботи з *ETHERNET*

З програми на мові ForthLogic™ можна ініціювати підключення до сервера в якості клієнта, розірвати з'єднання та подивитися стан з'єднання. Робота в якості сервера відбувається здебільшого автоматично на рівні системи і абсолютно прозоро для програми - можна лише перезавантажити відповідний сервер та подивитися стан з'єднання. Також з програми на мові ForthLogic™ можна здійснити загальні налаштування а також налаштування роботи відповідного сервера. Для цього реалізовано низку слів які представлено далі по тексту.

Початкові налаштування

Початкові налаштування контролерів, які містять вбудований порт *ETHERNET*, полягають у встановленні параметрів доступу до локальної мережі *ETHERNET* аналогічно, як це робиться для інших пристроїв в локальній мережі *ETHERNET* (ПЕОМ, мережевого обладнання, принтерів, тощо). Слова, призначені для цього описані в розділі "Керування системою".

Для зберігання параметрів роботи в режимі передачі даних на сервер баз даних MySQL із застосуванням протоколу HTTP, в енергонезалежній пам'яті існують спеціальні рядкові змінні ID (ідентифікатор), IP (IP-адреса) та URL (адреса). Слова для роботи з ними описані в розділі "Слова для роботи з GPRS".

Для з'ясування чи існує фізичне з'єднання з мережею *ETHERNET* існує слово LAN? (стан підключення до локальної мережі) яке повертає на стек логічне значення ІСТИНА, якщо існує фізичне з'єднання з діючою мережею *ETHERNET*, або логічне значення НЕ ІСТИНА у протилежному випадку.

Протокол сервера MODBUS TCP

Робота в якості сервера MODBUS TCP відбувається автоматично на рівні системи і абсолютно прозоро для програми на мові ForthLogic™. Сервер MODBUS TCP працює в режимі підпорядкованого пристрою MODBUS типу SLAVE і акцептує з'єднання з клієнтами, які працюють в режимі головного пристрою MODBUS типу MASTER. Одночасно до сервера MODBUS TCP може бути приєднаний лише

один клієнт.

Принцип обміну даними з сервером MODBUS TCP полягає в тому, що головний пристрій MODBUS типу MASTER (в якості якого може виступати як інший ПЛК так і система АСУТП чи SCADA) робить запити а сервер MODBUS TCP (типу SLAVE) автоматично та прозоро для користувача здійснює обробку запитів та формування відповідей. При цьому головний пристрій MODBUS може вільно встановлювати та читати всі глобальні змінні, глобальні змінні в форматі чисел з плаваючою комою та глобальні бітові змінні.

3 точки зору головного пристрою MODBUS типу MASTER, контролер представляється у вигляді пристрою типу SLAVE з набором змінних розташованих у певних адресах регістрів. Регістр в інтерпретації протоколу MODBUS є або 1-бітова величина або 16-бітова величина:

Адр	Назва	Опис	Тип	Доручення
2000	VAR1	Глобальна змінна 1	довгий	03, 06, 16
2002	VAR2	Глобальна змінна 2	довгий	03, 06, 16
...	...	...	...	...
2508	VAR255	Глобальна змінна 255	довгий	03, 06, 16
2510	VAR256	Глобальна змінна 256	довгий	03, 06, 16
3000	FLOAT1	Глобальна змінна в форматі чисел з плаваючою комою 1	плаваючий	03, 06, 16
3002	FLOAT2	Глобальна змінна в форматі чисел з плаваючою комою 2	плаваючий	03, 06, 16
...	...	...	...	...
3252	FLOAT127	Глобальна змінна в форматі чисел з плаваючою комою 127	плаваючий	03, 06, 16
3254	FLOAT128	Глобальна змінна в форматі чисел з плаваючою комою 128	плаваючий	03, 06, 16
4000	FLAG1	Глобальна бітова змінна 1	бінарний	01, 05, 15
4001	FLAG2	Глобальна бітова змінна 2	бінарний	01, 05, 15
...	...	...	...	...
4254	FLAG255	Глобальна бітова змінна 255	бінарний	01, 05, 15
4255	FLAG256	Глобальна бітова змінна 256	бінарний	01, 05, 15

Всі цифрові значення наведені в таблиці представлені в *десятковій формі*. Адр - це адреса регістра. Доручення - це код доручення (функції), яке може бути виконане пристроєм типу SLAVE. Контролер може обслуговувати коди наступних доручень:

- 01 (0x01) Read Coils (Читання статусу дискретних виходів/регістрів)
- 03 (0x03) Read Holding Registers (Читання статусу регістрів)
- 05 (0x05) Write Single Coil (Запис окремого дискретного виходу/регістру)
- 06 (0x06) Write Single Register (Запис окремого регістра)
- 15 (0x0F) Write Multiple Coils (Запис декількох дискретних виходів/регістрів)
- 16 (0x10) Write Multiple registers (Запис декількох регістрів)

Тип - це вигляд, в якому число записано в пам'яті контролера. Бінарний тип це число довжиною в 1 біт, яке може приймати лише два значення - "1" та "0", які відповідають логічному стану ІСТИНА та НЕ ІСТИНА глобальних бітових змінних контролера. Змінна бінарного типу займає 1 регістр MODBUS. Довгий тип це число довжиною в чотири байти (32-біти) зі знаком. Змінна довгого типу займає 2 сусідні

регістр MODBUS і у таблиці наведено адресу першого регістру, в якому розташовується старше слово (16 біт) змінної. У змінній довгого типу можливо зберігати цілі числа в діапазоні від -2147483648 до +2147483647 і вони повністю відповідають формату глобальних змінних контролера. Плаваючий тип це число довжиною в чотири байти (32-біти) згідно стандарту IEEE-754. Змінна плаваючого типу займає 2 сусідні регістри MODBUS і у таблиці наведено адресу першого регістру, в якому розташовується старше слово (16 біт) змінної. У змінній плаваючого типу можливо зберігати числа з плаваючою комою одинарної точності згідно стандарту IEEE-754 і вони повністю відповідають формату глобальних змінних з плаваючою комою контролера.

Програма на мові ForthLogic™ не має безпосереднього доступу до обміну даними, проте може взаємодіяти з головним пристроєм MODBUS через глобальні змінні. Також можна перезавантажити сервер MODBUS TCP, подивитися стан з'єднання та налаштувати фільтрацію доступу клієнтів на рівні IP-адрес.

У випадку виявлення збою в роботі сервера MODBUS TCP, його можна перезавантажити словом MBTCPRESET. При цьому існуюче з'єднання з клієнтом буде розірвано і через пару секунд той самий клієнт (або інший) може спробувати встановити нове з'єднання.

Стан з'єднання сервера MODBUS TCP з клієнтом можна оцінити за допомогою слова MBTCPREMIP, яке друкує у вихідному буфері та на терміналі IP-адресу активного клієнта. Зокрема, якщо вона пуста - то в даний момент немає жодного клієнта під'єданого до сервера MODBUS TCP.

Для забезпечення безпеки системи існує можливість налаштувати фільтрацію доступу клієнтів до сервера MODBUS TCP на рівні IP-адрес. Тобто, тільки клієнт з дозволеною IP-адресою може встановити з'єднання із сервером MODBUS TCP. Для встановлення дозволеної адреси призначене слово MBTCPSET, яке працює наступним чином: з вихідного буфера зчитується рядок - IP-адреса і запам'ятовується у відповідній комірці внутрішньої енергонезалежної пам'яті. Для відображення дозволеної IP-адреси у вихідному буфері та на терміналі призначене слово MBTCPGET. Якщо встановити дозволена адресу у вигляді пустого рядка, то фільтрація відключиться взагалі і будь-який клієнт може встановити з'єднання із сервером MODBUS TCP.

Протокол передачі даних на сервер баз даних MySQL

При роботі по протоколу передачі даних на сервер баз даних MySQL, дані передаються автоматично з внутрішньої оперативної пам'яті на сервер баз даних MySQL із застосуванням протоколу HTTP. Даний процес розпочинається словом LOG>DBTCP і повністю контролюється системою.

Принцип передачі даних на сервер баз даних MySQL полягає в тому, що система спочатку передає файл з внутрішньої оперативної пам'яті на HTTP-сервер за допомогою методу POST. Параметри IP-адреси та номеру порту HTTP-сервера необхідно вказувати через змінну IP (докладно про цю змінну див. розділ "Слова для роботи з GPRS"). Цей файл приймається та обробляється спеціалізованим скриптом, ім'я якого необхідно вказувати через змінну URL (докладно про цю змінну див. розділ "Слова для роботи з GPRS"). Далі цей файл аналізується та дані з нього передаються в таблицю на сервер баз даних MySQL. HTTP-сервер після отримання файлу не закриває фізичне з'єднання з контролером. По цій причині даний протокол можна застосовувати для досить інтенсивних режимів реєстрації та передачі даних. Рекомендований інтервал передачі даних на сервер складає від 5 секунд і вище.

Для успішного з'єднання з HTTP-сервером, контролер має пройти аутентифікацію, яка полягає в співпадінні імені та пароля, які задаються за допомогою змінної ID (докладно про цю змінну див. розділ "Слова для роботи з GPRS"). Крім того, програма на мові ForthLogic™ має забезпечити вірний формат та фіксовану кількість полів зареєстрованих даних для одного запису в файл реєстрації, наприклад:

"1304585524;22.233455;-1;13434", де:

1304585524	22.233455	-1	13434
Поле часу в форматі UTC	Поле числа з плаваючою комою, наприклад температура	Поле прапорця, наприклад стан входу	Поле цілого числа, наприклад кількість імпульсів на вході

Розділювачем між полями даних має бути символ ";" (крапка з комою), в кінці запису розділювач не застосовується. Першим полем має бути час в форматі UTC. Перше поле є обов'язковим. Решта полів та їх типи мають строго відповідати параметрам відповідної таблиці на сервері баз даних MySQL.

При роботі з даним протоколом контролер також може отримувати зворотню інформацію в кінці кожної сесії передачі файлу на HTTP-сервер. Дана інформація обробляється системою автоматично. При цьому HTTP-сервер може вільно встановлювати та читати всі глобальні змінні, глобальні змінні в форматі чисел з плаваючою комою та глобальні бітові змінні. Програма на мові ForthLogic™ може скористатися даними в цих змінних і в цей спосіб реалізувати "зворотній зв'язок" з HTTP-сервером.

Для встановлення з'єднання при роботі по протоколу передачі даних на сервер баз даних MySQL, існує слово DBTCROPEN, яке працює наступним чином: з вхідного тексту вибирається два чергових слова зворотнього виклику (введених після слова DBTCROPEN) і відбувається встановлення з'єднання. На стек при цьому кладеться логічне значення - результат виконання слова DBTCROPEN: логічне значення ІСТИНА означає, що процес з'єднання розпочався успішно, а НЕ ІСТИНА - що виникли апаратні проблеми (наприклад, вже триває з'єднання з сервером або відсутнє фізичне з'єднання з мережею ETHERNET). Коли успішно встановиться з'єднання, форт-система виконає перше з вказаних після слова DBTCROPEN слів зворотнього виклику, в іншому випадку, коли виникла помилка в спробі з'єднання, форт-система виконає друге з вказаних слів зворотнього виклику.

Для розриву з'єднання при роботі по протоколу передачі даних на сервер баз даних MySQL в будь-який момент часу, існує слово DBTCPCLOSE. Після розриву з'єднання, можна здійснити спробу з'єднання з іншим сервером. Для отримання статусу з'єднання при роботі по протоколу передачі даних на сервер баз даних MySQL існує слово DBTCPSTATUS, яке повертає на стек логічне значення ІСТИНА якщо існує фізичне з'єднання з віддаленим сервером.

Додаток

Налаштування програми-терміналу

В якості терміналу в середовищі ОС Microsoft® Windows®XP, можна використовувати програму емуляції терміналу Microsoft®HyperTerminal, яка входить до складу операційної системи.

Перед використанням програми емуляції терміналу, необхідно встановити драйвер USB. Для цього необхідно або запустити на виконання файл драйвера, який знаходиться в папці "USB" дистрибутива, або вказати шлях до папки "USB" дистрибутива. Після успішної установки, в операційній системі з'явиться новий послідовний порт (його номер можна перевірити та поміняти у програмі "Диспетчер устроїв").

Коли контролер під'єднаний до комп'ютера, можна запустити програму Microsoft®HyperTerminal (Пуск → Программи → Стандартные → Связь → HyperTerminal). При першому запуску, необхідно вказати назву нового підключення. Далі у вікні "Подключение" у списку "Подключаться через" вибрати номер послідовного порту, який було встановлено драйвером USB. Далі у вікні "Свойства: COM" вказати параметри зв'язку: "Скорость" - 19200 для контролерів з дисплеєм та 57600 для контролерів без

дисплея; "Биты данных" - 8; "Четность" - нет; "Стоповые биты" - 1; "Управление потоком" - нет і затвердити все кнопкою "ОК". Далі необхідно зайти в меню "Свойства" вікна програми Microsoft®HyperTerminal (Файл → Свойства), вибрати вкладку "Параметры" та в розділі "Эмуляция терминала" вибрати "TTY", далі натиснути кнопку "Параметры ASCII", зняти всі галочки і затвердити всі налаштування кнопкою "ОК". Після цього, слід зберегти обрані параметри, виконавши команду "Сохранить" у вікні програми Microsoft®HyperTerminal (Файл → Сохранить).

Якщо запускати програму Microsoft®HyperTerminal через ім'я підключення, яке було збереженого при першому запуску, то всі налаштування автоматично вступають в силу і більше нічого налаштовувати не треба.

Якщо програма Microsoft®HyperTerminal недоступна, то можна скористатись програмою емуляції терміналу інтегрованого середовища розробки FLIDE - програмою PuTTY. Параметри програми PuTTY вже налагоджені для роботи з ПЛК серії ES-ForthLogic™, проте їх можна довільно міняти (наприклад шрифт, колір, розмір вікна, тощо). Детальніше про програму PuTTY можна дізнатися із її вбудованої допомоги.

Створення звукових файлів

Для реалізації голосових повідомлень та меню, необхідно створити набір звукових фрагментів та записати їх у відповідному форматі на карту пам'яті SD/MMC. Для цього найкраще підходить програма Microsoft®Звукозапис (Пуск → Программы → Стандартные → Развлечения → Звукозапись) зі складу операційної системи Microsoft® Windows®XP. Кожний записаний звуковий фрагмент необхідно зберегти у файл з розширенням "wav" в форматі PCM з атрибутами "8,000 кГц; 8 бит; Моно 7КБ/с". Для цього у вікні програми Microsoft®Звукозапис вибрати меню "Сохранить как..." (Файл → Сохранить как...), у вікні "Сохранение файла" в полі "Формат:" натиснути кнопку "Изменить...", далі у вікні "Выбор звука" в полі "Формат:" вибрати PCM, а в полі "Атрибуты:" вибрати "8,000 кГц; 8 бит; Моно 7КБ/с" і затвердити всі налаштування кнопкою "ОК". Далі вказати назву файла та зберегти його.

Збережені таким чином файли, необхідно переписати у кореневу директорию на карту пам'яті SD/MMC та вставити її в контролер. В подальшому, при виконанні вбудованого слова PLAY, дані файли можуть будуть використані для відтворення звукових фрагментів.

Набір файлів чисел

Для відтворення словами чисел через аудіосистему контролера, необхідно забезпечити на карті пам'яті SD/MMC набір *.wav файлів які мають мати формат аналогічний до описаного в попередньому розділі та мати фіксовані назви. У наступній таблиці представлено перелік набору файлів.

Назва файлу	Голосовий вміст файлу	Назва файлу	Голосовий вміст файлу
_0.wav	нуль	_40.wav	сорок
_1.wav	один	_400.wav	чотириста
_10.wav	десять	_5.wav	п'ять
_100.wav	сто	_50.wav	п'ятдесят
_11.wav	одинадцять	_500.wav	п'ятсот
_12.wav	дванадцять	_6.wav	шість
_13.wav	тринадцять	_60.wav	шістдесят
_14.wav	чотирнадцять	_600.wav	шістсот

_15.wav	п'ятнадцять	_7.wav	сім
_16.wav	шістнадцять	_70.wav	сімдесять
_17.wav	сімнадцять	_700.wav	сімсот
_18.wav	вісімнадцять	_8.wav	вісім
_19.wav	дев'ятнадцять	_80.wav	вісімдесять
_2.wav	два	_800.wav	вісімсот
_20.wav	двадцять	_9.wav	дев'ять
_200.wav	двісті	_90.wav	дев'яносто
_3.wav	три	_900.wav	дев'ятсот
_30.wav	тридцять	comma.wav	кома
_300.wav	триста	minus.wav	мінус
_4.wav	чотири	plus.wav	плюс

Даний набір файлів має знаходитись в папці з будь-якою назвою на карті пам'яті SD/MMC.

Створення файлів-скриптів на мові ForthLogic™

Безпосередньо вводити з терміналу великі тексти незручно, тому їх зберігають у текстові файли та переносять у форт-систему. Ця можливість також може бути корисною при тиражуванні програм-скриптів та для резервного збереження створеного для конкретної задачі скрипту (це особливо актуально, коли здійснюється поновлення програмної прошивки контролера - зі словника форт-системи видаляються всі введені користувачем слова).

Правила створення

Процедура створення файлів-скриптів надзвичайно проста. В будь-якому текстовому редакторі, наприклад Microsoft® Блокнот (Пуск → Программи → Стандартные → Блокнот) зі складу операційної системи Microsoft® Windows®XP, необхідно набрати текст скрипту та зберегти його у файл з розширенням ".txt" або ".fl" (дане розширення є зручним для подальшої роботи через інтегроване середовище розробки FLIDE і асоціації файлів-скриптів з редактором середовища). При цьому необхідно слідкувати, щоб довжина рядків не перевищувала 77 символів і в кінці файлу був принаймні один пустий рядок. Кодування символів в файлі може бути будь-яким крім UNICODE.

Для більшої зручності створення файлів-скриптів, можна скористатися спеціальним текстовим редактором інтегрованого середовища розробки FLIDE - програмою Programmer's Notepad 2. У текстовому редакторі середовища вже реалізовано підсвічування синтаксису мови ForthLogic™, візуальний контроль за максимальною довжиною рядків у файлі і багато інших зручностей для роботи з текстовими документами. Детальніше про це можна дізнатися із вбудованої в редактор допомоги.

Правила інтерпретації з карти пам'яті

Якщо текст скрипту зберегти у кореневій директорії карти пам'яті SD/MMC у файлі "forthdic.txt", то його можна інтерпретувати у контролерах обладнаних дисплеєм та зчитувачем карт пам'яті SD/MMC. Для цього треба вставити карту в контролер та виконати пункт конфігураційного меню "SD/MMC" → "ВИКОНАТИ:" → "ВИКОН.СКРИПТ!". При цьому даний файл буде порядково інтерпретовано та виконано текстовим інтерпретатором форт-системи. При виявленні інтерпретатором будь-якої

помилки, його робота буде припинена і на вбудованому дисплеї буде відображено повідомлення про код помилки та номер рядка файлу, в якому виникла помилка.

Якщо текст скрипту зберегти у кореневій директорії карти пам'яті SD/MMC у файлі "autorun.txt", то при встановленні карти в будь-який контролер, обладнаний зчитувачем карт пам'яті SD/MMC, даний файл буде автоматично інтерпретовано та виконано текстовим інтерпретатором форт-системи. При виявленні інтерпретатором будь-якої помилки, його робота буде припинена, тому важливо, щоб даний скрипт був попередньо відлагоджений в діалоговому режимі. Після успішної інтерпретації, даний файл буде автоматично *видалено* з карти пам'яті.

Також для всіх типів контролерів, обладнаних зчитувачем карт пам'яті SD/MMC, існує можливість інтерпретації довільного файлу, розташованого на карті пам'яті SD/MMC. Для цього, в термінальному режимі роботи текстового інтерпретатора форт-системи необхідно з нового рядка без передуючих пробілів вказати команду COMPILE FILE <назва файлу>. При виявленні інтерпретатором будь-якої помилки, його робота буде припинена і у вікні терміналу буде відображено повідомлення про помилку та номер рядка файлу, в якому виникла помилка.

Правила інтерпретації через гіпертермінал

Текстовий інтерпретатор форт-системи може інтерпретувати файли не тільки з карти пам'яті SD/MMC. Якщо в термінальному режимі роботи вказати з нового рядка без передуючих пробілів команду RECEIVE FILE, то з'явиться запрошення на пересилку текстового файлу по протоколу CRC Xmodem. При цьому, на протязі 30 сек. необхідно ініціювати відправку файлу. В програмі Microsoft®HyperTerminal для цього необхідно зайти у вікно "Отправка файла" (Передача → Отправит файл...), вибрати за допомогою кнопки "Обзор" ім'я файлу, вказати "Протокол"- Xmodem та натиснути кнопку "Отправит". При виявленні інтерпретатором будь-якої помилки, його робота буде припинена, вікно відправки файлу закриється і у вікні терміналу буде відображено повідомлення про помилку та номер рядка файлу, в якому виникла помилка.

Інтерпретація файлів через гіпертермінал дозволяє скористатись закодованими бінарними файлами скриптів (по замовчуванню, для таких файлів прийняте розширення ".flx"). Такі файли дозволяють зберегти інтелектуальну власність автора програми, обмежити кількість інсталяцій в контролери і створюються спеціальним засобами. Процедура створення таких файлів надається за вимогою. Процедура інтерпретації не відрізняється від звичайної - лише треба вказувати команду RECEIVE FILEX.

Якщо програма Microsoft®HyperTerminal недоступна, то можна скористатись засобами пересилки файлів з інтегрованого середовища FLIDE - програмою FLprog. FLprog це програма для передачі файлів-скриптів та закодованих бінарних файлів по протоколу CRC Xmodem в ПЛК серії ES-ForthLogic™. Програма не має власного графічного інтерфейсу і призначена для запуску з командного рядка у складі різних інтегрованих середовищ розробки програм (IDE), зокрема у складі FLIDE.

Засади ефективного програмування

При створенні файлів-скриптів на мові ForthLogic™, слід чітко розуміти відмінність таких файлів від звичайних файлів програм в інших традиційних мовах програмування. Файл-скрипт може складатися з визначень нових слів та констант а також з інших слів та чисел, які безпосередньо обробляються та виконуються в момент інтерпретації даного файлу. Цей потужний механізм дозволяє автоматизувати практично будь-які операції (особливо за допомогою карт пам'яті) і є головною відмінністю файлів-скриптів.

Створення програм

Написання програм, це здебільшого інтерактивний процес: створення тексту - перевірка в реальних умовах - корекція тексту - перевірка в реальних умовах - корекція тексту - ... Враховуючи згадані особливості файлів-скриптів, можна порекомендувати наступний алгоритм при написанні та відлагоджуванні програм:

1. Створити текстовий файл програми.
2. На початку файлу додати наступні рядки:

```
STOPALL CLEARSYS
2 ERRORLEVEL
FORGET myprogram
0 ERRORLEVEL
: myprogram ." Короткий опис програми та її версія " ;
```

3. Описати всі необхідні слова та константи програми, при необхідності вставити слова, які будуть безпосередньо виконані під час інтерпретації.
4. В кінці файлу описати "головне слово" програми (наприклад *main*) та після опису додати рядки:

```
." main " BOOT
main
```

5. Додати ще один пустий рядок та зберегти файл.
6. Відправити файл на інтерпретацію через гіпертермінал або через FLprog.
7. Якщо в процесі інтерпретації не виявилось синтаксичних або інших помилок, то програма буде запущена на виконання і можна перейти до п.10.
8. При виявленні інтерпретатором будь-якої помилки, його робота буде припинена, вікно відправки файлу через гіпертермінал закриється і у вікні терміналу буде відображено повідомлення про помилку та номер рядка файлу, в якому виникла помилка. Якщо пересилка відбувалася через FLprog, то повідомлення про помилку буде відображено у вікні FLprog.
9. Після виправлення помилки необхідно повторити п.6.
10. Після перевірки роботи програми, внести необхідні корекції в файл і повторити п.6. Якщо вносити корекції вже не потрібно, то програма готова і файл є придатним для передачі замовнику або для довготривалого зберігання.

І нарешті, традиційна при написанні програм рекомендація: максимально використовуйте коментарі в тексті файлів-скриптів (за допомогою дужок), розділяйте частини програми пустими рядками та відділяйте окремі конструкції програми відступами.

Відлагодження програм

Традиційні способи відлагодження програм полягають в можливості зупинки програми, перегляду та зміни контексту під час зупинки, покрокового виконання, тощо. В силу обмежень апаратної платформи та особливостей реалізації багатозадачності на мові ForthLogic™, не всі класичні засоби та способи можливі до реалізації. Проте, інтерпретуючий характер мови дозволяє певну ступінь свободи та надає

можливості нездійсненні для інших програмно-апаратних платформ (головним чином заснованих на компіляторах). Наприклад, відсутність етапу компіляція прискорює процес створення та відлагодження окремих програмних процедур (слів в термінах форт-системи): задавши слово в термінальному режимі, його можна одразу перевірити "підставивши" через стек необхідні для його роботи параметри. Доступ до інтерпретатора в термінальному режимі під час роботи програми також дозволяє певним чином втручатися в її роботу: стеки та глобальні змінні доступні до модифікації в довільний момент часу. Проте, для додаткової зручності, існують слова які дозволяють більш ефективно здійснювати процес відлагодження. Розглянемо їх.

Для динамічного відображення стану виконання програми використовується спеціальний рядковий *буфер повідомлень* довжиною до 78 символів, зміст якого відображається червоним кольором в квадратних дужках під час роботи в режимі відображення статусної інформації про систему у вікні терміналу (див. розд. "Робота в діалоговому режимі"). В будь-якому місці програми можна застосувати слово MESSAGE (повідомлення), яке записує текст з вихідного буфера в буфер повідомлень і він стає видимим в режимі відображення статусної інформації. Таким чином можна організувати передачу даних з програми про ті чи інші процеси, які відбуваються на-живо. Текст у буфері повідомлень можна дописувати. Для цього його треба копіювати у вихідний буфер за допомогою слова MESSAGE. (повідомлення-крапка), додати будь-яким способом новий текст до вмісту вихідного буфера і знову записати результат до буфера повідомлень.

Щоб "побачити" вміст обох стеків не змінюючи їх стану, використовуються слова .S (крапка-стек) та .FS (крапка-математичний стек), які відповідно друкують на терміналі вміст цілочислового та математичного стеку в квадратних дужках в тому порядку, як це прийнято для стекової нотації - верхні значення на стеку (ті які були додані останніми) знаходяться правіше. Значення з математичного стеку друкуються в інженерному/науковому представленні із 6 знаками після десяткової коми. Дані слова можна використовувати лише в режимі інтерпретації (їх не можна використовувати у визначенні інших слів).

Реалізація багатозадачності за допомогою таймерів є простою та потужною концепцією. Іноді буває необхідно зупинити всі задачі, які прив'язані до різних таймерів. Вручну зупиняти кожний таймер довго і не ефективно. Це можна зробити за допомогою слова STOPALL (зупинити всі). При цьому обнуляються проміжки часу та призначені слова всіх таймерів, зупиняються всі налаштовані обміни по протоколу MODBUS RTU, зупиняється будь-яка автоматична активність на лініях входу та виходу, зупиняється реєстрація в режимі користувача, зупиняється відтворення повідомлень та будь-яка активність в каналі GSM/GPRS/CSD, зупиняється робота клієнтів в каналі ETHERNET. Стеки, глобальні змінні та стан виходів залишаються без змін. Для обнулення програмно-доступних стеків, глобальних змінних, буфера повідомлень та вихідних сигналів існує слово CLEARSYS (очистити систему). Більш "глибоку очистку" можна виконати за допомогою перезапуску системи. Слово RESTART (перезапуск) виконує перезавантаження всієї системи та аналогічно холодному старту (вимкнення-увімкнення живлення системи). Дані слова можна використовувати як в режимі інтерпретації так і в складі інших слів.

Для реалізації механізму точок зупинки програми, існують слова BREAKPOINT (точка зупинки) та DEBUGLEVEL (рівень відладки). Слово BREAKPOINT призначено для зупинки форт-системи в процесі відладки і працює наступним чином: зі стека знімається верхнє значення - номер точки зупинки, і якщо цей номер менший або рівний за глобальну системну змінну DEBUGLEVEL, то відбувається зупинка форт-системи. При цьому зупиняються всі таймери, зупиняються всі налаштовані обміни по протоколу MODBUS RTU, зупиняється будь-яка автоматична активність на лініях входу та виходу, зупиняється реєстрація в режимі користувача, зупиняється відтворення повідомлень та будь-яка активність в каналі GSM/GPRS/CSD, зупиняється робота клієнтів в каналі ETHERNET, а на терміналі в фігурних дужках друкуються інформація про номер точки зупинки, вміст всіх стеків (цілочислового, математичного та повернень) та вихідного буфера на момент зупинки. Після цього всі стеки та буфери обнуляються а

глобальні змінні залишаються без змін.

Слово `DEBUGLEVEL` призначено для встановлення однойменної системної змінної і працює наступним чином: зі стека знімається верхнє значення і запам'ятовується в глобальній системній змінній `DEBUGLEVEL`. Початкове значення цієї змінної дорівнює нулю і вона не є енергонезалежною. Зміст даної змінної можна інтерпретувати як *рівень відладки*, який активує різні групи точок зупинки. Наприклад, якщо в програмі існує кілька груп точок зупинки, які мають номери від 100 до 200 і від 200 до 300, то присвоєння системній змінній `DEBUGLEVEL` значення 200 блокує другу групу точок зупинки (форт-система їх ігнорує), а присвоєння системній змінній `DEBUGLEVEL` значення 0 (значення по замовчуванню) блокує всі точки зупинки і програма працює в звичайному режимі. Таким чином, слова `BREAKPOINT` можуть залишатися в штатній програмі постійно при умові, що їх номери більші за нуль. Слово `DEBUGLEVEL` можна використовувати лише в режимі інтерпретації.

Крім точок зупинок можна застосовувати точки спостереження. Для цього в будь-якому фрагменті тексту програми необхідно застосувати слово `WATCHPOINT` (точка спостереження). При досягненні цього слова зі стека знімається верхнє значення - номер точки спостереження і форт-система друкує на терміналі в квадратних дужках поточний час та номер точки а в круглих дужках вміст цілочислового та математичного стеків а також вихідного буфера на даний момент. Після цього форт-система продовжує виконання програми.

В процесі створення програми можливі певні типи помилок які призводять до перезавантаження всієї системи. Якщо в системі налаштовано автозапуск головного слова (за допомогою меню або слова `BOOT`), то при виникненні вищезгаданих помилок система буде неперервно перезавантажуватись. Для уникнення такого явища реалізовано механізм блокування виконання головного слова в термінальному режимі роботи. Тобто, при підключеному кабелі USB та праці через термінал автозапуск не відбувається. В цьому випадку запустити програму можна вручну набравши її головне слово в терміналі.

Помилки мови ForthLogic™

В термінальному та дистанційному режимі роботи форт-системи, всі помилки, які виникають під час діалогу із системою, по замовчуванню відображаються в круглих дужках у вигляді тексту на англійській мові. Під час інтерпретації файлу "forthdic.txt", у разі виникнення помилки, на дисплеї буде відображено повідомлення про код помилки, а не сам текст. Додатково по замовчуванню, в термінальному режимі роботи в квадратних дужках відображаються всі помилки, які виникають під час роботи всієї форт-системи (це стосується і тих частин, які виконують слова призначені таймерам та іншим відкладеним у часі процесам так і частинам, що інтерпретують слова в процесі діалогу). При цьому, в квадратних дужках відображається час виникнення помилки, текст помилки та розшифрований стан стеку повернень на момент виникнення помилки у вигляді ланцюжка слів - дана інформація дозволяє реконструювати місце виникнення помилки. Відображення помилок по замовчуванню можна відключати та включати за допомогою слова `ERRORLEVEL` (рівень відображення помилок), яке знімає зі стеку числовий параметр рівня відображення помилок та налаштовує відповідно поведінку форт-системи. Даний параметр не зберігається під час пропадання живлення і по замовчуванню дорівнює 0 - що означає відображення всіх помилок. Якщо встановити даний параметр рівним 1, то відображаються тільки помилки в квадратних дужках в термінальному режимі роботи. Якщо встановити даний параметр рівним 2, то форт-система взагалі не відображає жодних помилок - такий режим може стати в нагоді в деяких випадках під час відлагодження програм.

Далі представлені всі відомі форт-системі помилки, їх коди та пояснення:

Код	Текст помилки	Пояснення
1	UNKNOWN WORD	Невідоме форт-системі слово
2	ILLEGAL USAGE	Застосування слова у невірному контексті (виконання або компіляції)
3	ILLEGAL PARAMETER	Невірний параметр слова (поза дозволеними межами)
4	INSUFFICIENT PARAMETERS	Недостатня кількість параметрів
5	DATA STACK EMPTY	Стек даних порожній
6	DATA STACK FULL	Переповнення стеку даних
7	RETURN STACK EMPTY	Стек повернень порожній
8	RETURN STACK FULL	Переповнення стеку повернень
9	OUT OF MEMORY	Недостатньо пам'яті в словнику
10	MATHEMATIC STACK EMPTY	Математичний стек порожній
11	MATHEMATIC STACK FULL	Переповнення математичного стеку
12	SD CARD NOT FOUND	Карта пам'яті SD/MMC відсутня
13	FILE NOT FOUND	Файл не знайдено
14	INPUT BUFFER OVERFLOW	Переповнення вхідного буфера
15	FILE SYSTEM BUSY	Файлова система зайнята (наприклад, відтворюється звуковий файл)
16	FILE EMPTY	Файл порожній
17	WRONG CONSTRUCTION	Неповна конструкція керування виконанням (IF-ELSE-THEN)
18	FILE TRANSFER ERROR	Помилка під час прийому файлу по протоколу Xmodem
19	FILE TRANSFER TIMEOUT	Ініційований прийому файлу по протоколу Xmodem не розпочався на протязі 30 сек.
20	COMPILE MODE NOT ALLOWED	Заборона режиму компіляції (стосується текстів SMS)

Апаратна платформа мови ForthLogic™

Апаратна платформа на якій реалізована конкретна форт-система з точки зору програмної моделі характеризується різною кількістю програмно-доступних ресурсів та різним набором базових слів.

Для всіх апаратних платформ спільні наступні ресурси та параметри. Всі цілочислові арифметичні та логічні операції здійснюються над стеком даних глибиною 16 елементів. Діапазон представлення цілих чисел від -2147483648 до 2147483647. Всі математичні операції з числами з плаваючою комою здійснюються над математичним стеком глибиною 16 елементів. Діапазон представлення чисел з плаваючою комою відповідає одинарній точності згідно стандарту IEEE-754: $\pm(1.4 \times 10^{-45} \dots 3.4 \times 10^{38})$. В логічних операціях ІСТИНА означає довільне число не рівне 0 (звичайно -1), НЕ ІСТИНА означає 0. Операції, які повертають логічне значення ІСТИНА, повертають його у вигляді -1. Інформація у вихідному буфері дублюється або відображенням у вікні терміналу в термінальному режимі форт-системи або відображенням в тексті SMS у дистанційному режимі форт-системи для апаратних платформ, які мають модуль GSM. Глибина програмно недосяжного стеку повернень дорівнює 64 рівня. Стек повернень використовується системою для рекурсивного виклику вбудованих слів з інших вбудованих слів а також для зберігання контексту конструкції IF-ELSE-THEN.

В наступній таблиці версій представлені ресурси форт-системи, які залежать від апаратної платформи та версії прошивок, для яких також актуальна дана настанова з програмування:

Ресурс	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
FIRMWARE	Версія прошивки (мікропрограми)	-	1.16(B)	1.16(D)	1.16(GII)	4.50	2.06(SG) 1.14(SG)	3.06(SB)	2.06(SB)	2.06(ST)	1.02(M)
D_MAX	Кількість глобальних змінних для зберігання	13	64	64	64	128	256	256	256	128	64

Ресурс	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
	цілих чисел										
F_MAX	Кількість глобальних математичних змінних для зберігання чисел у форматі з плаваючою комою	14	32	32	32	64	128	128	128	64	16
B_MAX	Кількість глобальних однокітових змінних	14	128	128	128	128	256	256	256	128	64
S_MAX	Кількість рядкових змінних	15	-	4	4	8	16	4	4	8	-
T_MAX	Кількість незалежних таймерів	20	32	32	32	64	128	128	128	64	32
DI_MAX	Кількість цифрових входів	24	8	8	8	8	6*	6*	6*	6*	6*
AI_MAX	Кількість аналогових входів	24	4	4	4	4	2	2	2	2	-
DO_MAX	Кількість цифрових виходів	28	4	4	4	4	6*	6*	6*	6*	6*
RO_MAX	Кількість релейних виходів	28	3	3	3	3	-	-	-	-	-
MENU_MAX	Кількість пунктів меню користувача	52	-	4	-	8	-	-	-	-	-
ROW_MAX	Кількість рядків тексту дисплея	50	-	7	-	6	-	-	-	-	-
COL_MAX	Кількість стовпчиків тексту дисплея	50	-	15	-	15	-	-	-	-	-
PHONE_MAX	Кількість рядкових змінних для збереження номерів телефонів	56	-	-	40	40	50	-	-	50	-
OUTBUF_MAX	Розмір вихідного буфера, байт	14	120	120	180	180	180	120	120	180	80
LOGBUF_MAX	Розмір буфера реєстрації, байт	42	-	-	512	256	1024	2048	2048	-	-

* зі складу універсальних цифрових входів/виходів. Наприклад якщо задіяно 4 входи то залишається 2 виходи, якщо задіяно 1 вхід, то залишається 5 виходів, тощо.

Вбудовані слова мови ForthLogic™

Далі представлені всі відомі форт-системі вбудовані слова з коротким описом та посиланням на відповідну сторінку документу з детальним описом:

№	Група	Слово	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
1	vx/вих	AI?	кладе на мат. стек стан аналогового входу	24	+	+	+	+	+	+	+	+	
2	vx/вих	AIS?	кладе на мат. стек масштабований стан аналогового входу	25				+					
3	vx/вих	BAT?	кладе на мат. стек напругу живлення акумулятора	47	+	+	+	+					
4	vx/вих	POW?	кладе на мат. стек напругу основного живлення	47	+	+	+	+					
5	vx/вих	DI?	кладе на стек стан цифрового входу	24	+	+	+	+	+	+	+	+	+
6	vx/вих	DO!	встановлює стан цифрового виходу рівним значенню зі стеку	28	+	+	+	+	+	+	+	+	+
7	vx/вих	DO?	кладе на стек стан цифрового виходу	28	+	+	+	+	+	+	+	+	+
8	vx/вих	RO!	встановлює стан релейного виходу рівним значенню зі стеку	28	+	+	+	+					
9	vx/вих	RO?	кладе на стек стан релейного виходу	28	+	+	+	+					
10	vx/вих	WIEGANDSTART	налаштовується та запускається ядро протоколу WIEGAND-26	26					+	+	+		
11	vx/вих	WIEGANDSTOP	зупиняється ядро протоколу WIEGAND-26	26					+	+	+		
12	vx/вих	PWMSTART	налаштовується та запускається генерація сигналу ШИМ	29					+	+	+		
13	vx/вих	PWMSTOP	зупиняється генерація сигналу ШИМ	29					+	+	+		
14	vx/вих	COUNTERSTART	налаштовується та запускається лічильник імпульсів	25					+	+	+		
15	vx/вих	COUNTERSTOP	зупиняється лічильник імпульсів	25					+	+	+		
16	vx/вих	COUNTERGET	на стек повертається вміст лічильника імпульсів	25					+	+	+		
17	GSM	OPERATOR	друкує у вих. буфері оператора мережі	55			+	+	+				+
18	GSM	ROAMING?	кладе на стек ознаку роботи в роумінгу	55		+	+	+	+				+
19	GSM	SIGNAL?	кладе на стек рівень сигналу мережі	55			+	+	+				+
20	GSM	MODULE?	кладе на стек розширені параметри мережі та модуля GSM	55					+				+
21	GSM	SIM?	кладе на стек стан активації карти SIM	55				+	+				+
22	GSM	LAST	друкує у вих. буфері номера телефону останнього абонента	56			+	+	+				+
23	GSM	USER	друкує у вих. буфері номер телефону користувача	56			+	+	+				+
24	GSM	USERPHONE	встановлює номер телефону користувача	56			+	+	+				+
25	GSM/текст	SMS	відправляє SMS	57			+	+	+				+
26	GSM/текст	USSD	відправка USSD-запиту	58			+	+	+				+
27	GSM/голос	ANSWER	відповідь на голосовий дзвінок	61				+	+				
28	GSM/голос	CLIP	виконання дії у відповідь на голосовий дзвінок	62			+	+	+				+
29	GSM/голос	DIAL	здійснення голосового дзвінка	60				+	+				
30	GSM/голос	HOLD	припинення голосового дзвінка	60				+	+				
31	GSM/голос	HOOK?	кладе на стек статус голосового дзвінка	60				+	+				
32	GSM/голос	MUTE	припинення відтворення звуку	60				+	+				
33	GSM/голос	PLAY	відтворення звукового файлу з карти пам'яті SD/MMC	59				+	+				
34	GSM/голос	SAY	відтворення значення з вершини мат. стеку	60				+	+				
35	GSM/голос	MIC	підключення до каналу GSM зовнішньої аудіо системи та динаміка	66				+	+				
36	GSM/голос	VOICE	підключення до каналу GSM внутрішнього синтезатора	66				+	+				
37	GSM/голос	MICLEVEL	встановлення чутливості зовнішньої аудіо системи	66				+	+				
38	GSM/голос	SPKLEVEL	встановлення підсилення зовнішнього динаміка	66				+	+				
39	GSM/голос	AUTOSAYPLUS	вимикає автоматичне відтворення знаку плюс	60				+	+				
40	GSM/голос	NOAUTOSAYPLUS	вимикає автоматичне відтворення знаку плюс	60				+	+				
41	GSM/CSD	CONNECTCSD	встановлення каналу даних CSD	67			+		+				
42	GSM/CSD	SENDSCD	відправка рядкових даних через канал CSD	67			+		+				
43	GSM/CSD	BREAKCSD	розрив каналу даних CSD	67			+		+				
44	GSM/CSD	STATUSCSD	кладе на стек стан каналу даних CSD	67			+		+				

№	Група	Слово	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
45	GSM/CSD	MODEMCSD	активує прозорий модемний режим	68			+		+				
46	GSM/DTMF	TONE	генерація сигналів DTMF	63				+	+				
47	GSM/DTMF	WAITKEY	введення поодиноких сигналів DTMF	63				+	+				
48	GSM/DTMF	WAITPW	введення пароля за допомогою сигналів DTMF	64				+	+				
49	GSM/DTMF	WAITSTR	введення рядка цифр за допомогою сигналів DTMF	65				+	+				
50	GSM/DTMF	DTMFCONFIRM	вмикає автоматичне підтвердження прийнятого сигналу DTMF	63				+	+				
51	GSM/DTMF	NODTMFCONFIRM	вимикає автоматичне підтвердження прийнятого сигналу DTMF	63				+	+				
52	GSM/GPRS	APN	друкує у вих. буфері вміст енергонезалежної змінної APN	70			+		+				
53	GSM/GPRS	SETAPN	встановлює вміст енергонезалежної змінної APN	70			+		+				
54	GSM/GPRS	IP	друкує у вих. буфері вміст енергонезалежної змінної IP	70			+		+				
55	GSM/GPRS	SETIP	встановлює вміст енергонезалежної змінної IP	70			+		+				
56	GSM/GPRS	ID	друкує у вих. буфері вміст енергонезалежної змінної ID	70			+		+				
57	GSM/GPRS	SETID	встановлює вміст енергонезалежної змінної ID	70			+		+				
58	GSM/GPRS	URL	друкує у вих. буфері вміст енергонезалежної змінної URL	70					+				
59	GSM/GPRS	SETURL	встановлює вміст енергонезалежної змінної URL	70					+				
60	GSM/GPRS	ATTACHGPRS	підключення до сервісу GPRS	70			+		+				
61	GSM/GPRS	CONFIGTCP	налаштування GPRS з'єднання	70			+		+				
62	GSM/GPRS	CLIENT	системна константа для слова CONFIGTCP	70			+		+				
63	GSM/GPRS	CLIENTDB	системна константа для слова CONFIGTCP	70					+				
64	GSM/GPRS	CLIENTSQL	системна константа для слова CONFIGTCP	70					+				
65	GSM/GPRS	CLIENTOPC	системна константа для слова CONFIGTCP	70			+		+				
66	GSM/GPRS	SERVER	системна константа для слова CONFIGTCP	70			+		+				
67	GSM/GPRS	CONFIGPORT	встановлення порту в режимі сервера	70			+		+				
68	GSM/GPRS	DNSMODE	встановлення режиму URL-адрес	71			+		+				
69	GSM/GPRS	IPMODE	встановлення режиму IP-адрес	71			+		+				
70	GSM/GPRS	OPENTCP	встановлення з'єднання згідно протоколу TCP	71			+		+				
71	GSM/GPRS	CLOSETCP	розрив з'єднання згідно протоколу TCP	71			+		+				
72	GSM/GPRS	SHUTTCP	відключення від GPRS	72			+		+				
73	GSM/GPRS	SENDTCP	передача рядкових даних згідно протоколу TCP	71			+		+				
74	GSM/GPRS	STATUSGPRS	повертає на стек стан сервісу GPRS	70			+		+				
75	GSM/GPRS	STATUSSERVER	повертає на стек стан роботи в режимі сервера	72			+		+				
76	GSM/GPRS	STATUSTCP	повертає на стек стан з'єднання згідно протоколу TCP	72			+		+				
77	GSM/GPRS	STATUSOPC	повертає на стек стан з'єднання з OPC-сервером	72			+		+				
78	GSM/GPRS	REMOTEIP	друкує у вих. буфері IP-адресу клієнта в режимі сервера	72			+		+				
79	GSM/GPRS	LOCALIP	друкує у вих. буфері власну IP-адресу	72			+		+				
80	GSM/GPRS	M2MTCAPON	включає режим без відповіді форт-системи	69			+		+				
81	GSM/GPRS	M2MTCPOFF	включає нормальний режим відповіді форт-системи	69			+		+				
82	ETHERNET	LAN?	кладає на стек стан з'єднання з мережею ETHERNET	76						+			
83	ETHERNET	MBTCPRESET	перезавантаження сервера MODBUS TCP	78						+			
84	ETHERNET	MBTCPREMIP	друкує у вих. буфері адресу активного клієнта MODBUS TCP	78						+			
85	ETHERNET	MBTCPPIPSET	встановлює адресу дозволеного клієнта MODBUS TCP	78						+			
86	ETHERNET	MBTCPPIPGET	друкує у вих. буфері адресу дозволеного клієнта MODBUS TCP	78						+			
87	ETHERNET	DBTCPOPEN	встановлення з'єднання із сервером передачі даних	79						+			
88	ETHERNET	DBTCPCLOSE	розрив з'єднання із сервером передачі даних	79						+			
89	ETHERNET	DBTCPSTATUS	повертає на стек стан з'єднання із сервером передачі даних	79						+			
90	меню	FOCUS	встановлює фокус на пункт меню користувача	53		+		+					
91	меню	HIDE	робить невидимим пункт меню користувача	53		+		+					
92	меню	INFO	друкує текст в інформаційному полі меню користувача	53									
93	меню	LASTMENU?	кладає на стек номер останнього задіяного пункту меню корист.	53		+		+					
94	меню	MENU	налаштовує та робить видимим пункт меню користувача	52		+		+					
95	дисплей	BLACK	зміна кольору шрифту на чорний	51		+		+					
96	дисплей	BLUE	зміна кольору шрифту на блакитний	51		+		+					
97	дисплей	DEEPBLUE	зміна кольору шрифту на синій	51		+		+					
98	дисплей	GREEN	зміна кольору шрифту на зелений	51		+		+					
99	дисплей	RED	зміна кольору шрифту на червоний	50		+		+					
100	дисплей	ORANGE	зміна кольору шрифту на помаранчевий	50		+		+					
101	дисплей	VIOLET	зміна кольору шрифту на фіолетовий	51		+		+					
102	дисплей	WHITE	зміна кольору шрифту на білий	50		+		+					
103	дисплей	YELLOW	зміна кольору шрифту на жовтий	51		+		+					
104	дисплей	INVERT	інверсія кольорів шрифту та фону	51		+		+					
105	дисплей	CLEAR	очистка дисплея	51		+		+					
106	дисплей	PRINT	друк рядка з вихідного буфера на дисплеї	50		+		+					
107	дисплей	GET	відображення на дисплеї вікна для введення числових значень	51		+		+					
108	дисплей	GETS	відображення на дисплеї вікна для введення рядкових значень	52		+		+					
109	клавіші	BUTTON	налаштовує функцію клавіші	50			+		+				
110	клавіші	F1	системна константа для слів BUTTON, LED! та LED?	50		+		+		+	+		+
111	клавіші	F2	системна константа для слів BUTTON, LED! та LED?	50		+		+		+	+		
112	клавіші	DOWN	системна константа для слова BUTTON	50		+		+					
113	клавіші	LEFT	системна константа для слова BUTTON	50		+		+					
114	клавіші	OK	системна константа для слова BUTTON	50		+		+					
115	клавіші	RIGHT	системна константа для слова BUTTON	50		+		+					
116	клавіші	UP	системна константа для слова BUTTON	50		+		+					
117	сигнал	BEEP	генерація звукового сигналу	53	+	+	+	+	+	+	+	+	
118	сигнал	LED!	встановлює стан вбудованого світлодіода рівним значенню зі стеку	54						+	+		+
119	сигнал	LED?	кладає на стек стан вбудованого світлодіода	54						+	+		+
120	аудіо	AUDIOMUTE	припинення відтворення звуку через аудіосистему	55					+				
121	аудіо	AUDIOPLAY	відтворення звук. файлу з карти SD/MMC через аудіосистему	54					+				
122	аудіо	AUDIOSAY	відтворення значення з вершини мат. стеку через аудіосистему	54					+				
123	сист. час	>DATE	перетворює час UTC в загальноприйнятну дату	48		+	+	+	+	+	+	+	
124	сист. час	>TIME	перетворює час UTC в загальноприйнятий час	48		+	+	+	+	+	+	+	
125	сист. час	>UTC	перетворює загальноприйнятий час в час UTC	47	+	+	+	+	+	+	+	+	
126	сист. час	>WDAY	перетворює час UTC в загальноприйнятий день тижня	48		+	+	+	+	+	+	+	
127	сист. час	DATE?	кладає на стек значення років, місяців, днів	47	+	+	+	+	+	+	+	+	

№	Група	Слово	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
128	сист. час	TIME?	кладе на стек значення секунд, хвилин, годин	47	+	+	+	+	+	+	+	+	
129	сист. час	UTC?	кладе на стек секунди UTC так, як це прийнято в ОС UNIX	47	+	+	+	+	+	+	+	+	
130	сист. час	WDAY?	кладе на стек значення днів тижня	47	+	+	+	+	+	+	+	+	
131	сист. час	DST?	кладе на стек признак дії літнього часу	47			+		+	+	+	+	
132	сист. час	TIMESTAMP	друкує у вих. буфері календарне представлення часу	48			+		+	+	+	+	
133	сист. час	SUNRISE?	кладе на стек значення секунд, хвилин, годин сходу сонця	49					+	+	+	+	
134	сист. час	SUNSET?	кладе на стек значення секунд, хвилин, годин заходу сонця	49					+	+	+	+	
135	сист. час	CIVIL	системна константа для слів SUNRISE? та SUNSET?	49					+	+	+	+	
136	сист. час	OFFICIAL	системна константа для слів SUNRISE? та SUNSET?	49					+	+	+	+	
137	MODBUS	MODBUSSTART	формується та активується пакет протоколу MODBUS	30	+	+	+	+	+	+	+	+	
138	MODBUS	MODBUSSTOP	припиняється формування пакету протоколу MODBUS	32	+	+	+	+	+	+	+	+	
139	MODBUS	CYCLIC_ACCESS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
140	MODBUS	SINGLE_ACCESS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
141	MODBUS	READ_COILS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
142	MODBUS	READ_HOLDREGS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
143	MODBUS	READ_INPUTREGS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
144	MODBUS	READ_INPUTS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
145	MODBUS	WRITE_COIL	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
146	MODBUS	WRITE_COILS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
147	MODBUS	WRITE_REG	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
148	MODBUS	WRITE_REGS	системна константа для слова MODBUSSTART	30	+	+	+	+	+	+	+	+	
149	MODBUS	MODBUSCALLBACK	налаштовує автоматичний зворотний виклик протоколу MODBUS	32	+	+	+	+	+	+	+	+	
150	MODBUS	MODBUSPARAM	встановлює параметри послідовного каналу протоколу MODBUS	33	+	+	+	+	+	+	+	+	
151	MODBUS	EVEN	системна константа для слова MODBUSPARAM	33	+	+	+	+	+	+	+	+	
152	MODBUS	NONE	системна константа для слова MODBUSPARAM	33	+	+	+	+	+	+	+	+	
153	MODBUS	ODD	системна константа для слова MODBUSPARAM	33	+	+	+	+	+	+	+	+	
154	MODBUS	MODBUSSTATUS?	стан обміну по протоколу MODBUS	32	+	+	+	+	+	+	+	+	
155	MODBUS	MODBUSTIMEOUT!	встановлює час TIMEOUT протоколу MODBUS	33	+	+	+	+	+	+	+	+	
156	MODBUS	MODBUSRESET	перезапуск ядра протоколу MODBUS RTU	34					+	+	+	+	
157	реєстр.	LOGON	запускає та налаштовує процес реєстрації	42	+	+	+	+	+	+	+	+	
158	реєстр.	LOGRUN	запускає процес реєстрації з існуючими налаштуваннями	43	+	+	+	+	+	+	+	+	
159	реєстр.	LOGOFF	зупиняє процес реєстрації	43	+	+	+	+	+	+	+	+	
160	реєстр.	ALL_DATA	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
161	реєстр.	EVENTS_MODE	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
162	реєстр.	USER_MODE	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
163	реєстр.	INPUTS	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
164	реєстр.	INTERVAL_MODE	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
165	реєстр.	OUTPUTS	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
166	реєстр.	TO_FLASH	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
167	реєстр.	TO_RAM	системна константа для слова LOGON	42					+	+	+	+	
168	реєстр.	TO_SD	системна константа для слова LOGON	42	+	+	+	+	+	+	+	+	
169	реєстр.	TO_TCP	системна константа для слова LOGON	42			+		+				
170	реєстр.	LOG	перепише вихідний буфер у системний реєстр	46	+	+	+	+	+	+	+	+	
171	реєстр.	LOG?	кладе на стек стан процесу реєстрації	44	+	+	+	+	+	+	+	+	
172	реєстр.	LOGSEND?	кладе на стек стан процесу переносу даних	44	+	+	+	+	+	+	+	+	
173	реєстр.	LOGFILE?	кладе на стек статус зареєстрованих даних	44	+	+	+	+	+	+	+	+	
174	реєстр.	LOG>SD	переносить дані реєстрації на карту пам'яті SD/MMC	43	+	+	+	+	+	+	+	+	
175	реєстр.	LOG>TCP	переносить дані реєстрації через канал GPRS	43			+		+				
176	реєстр.	LOG>DB	переносить дані реєстрації через канал GPRS в базу MySQL	43					+				
177	реєстр.	LOG>SQL	переносить дані реєстрації через канал GPRS в базу MySQL	43									
178	реєстр.	LOG>DBTCP	переносить дані реєстрації через канал ETHERNET в базу MySQL	44						+			
179	реєстр.	FREELOG?	кладе на стек кількість вільного місця у внутр. пам'яті реєстрації	44	+	+	+	+	+	+	+	+	
180	реєстр.	LOGFORMAT	форматує внутрішню пам'ять реєстрації	44	+	+	+	+	+	+	+	+	
181	реєстр.	LOGFILETOGGLE	переключення активного файлу реєстру	44					+	+	+	+	
182	реєстр.	LOGFILEDELETE	видалення не активного файлу реєстру	44					+	+	+	+	
183	реєстр.	LOGTITLEOFF	вимикає автоматичний заголовок під час реєстрації	46	+	+	+	+	+	+	+	+	
184	реєстр.	LOGTITLEON	вмикає автоматичний заголовок під час реєстрації	46	+	+	+	+	+	+	+	+	
185	реєстр.	LOGTERMINON	вимикає автоматичне закінчення рядків під час реєстрації	46			+		+	+	+	+	
186	реєстр.	LOGTERMINOFF	вмикає автоматичне закінчення рядків під час реєстрації	46			+		+	+	+	+	
187	службов.	:	початок визначення нового слова	9	+	+	+	+	+	+	+	+	+
188	службов.	;	закінчення визначення нового слова	9	+	+	+	+	+	+	+	+	+
189	службов.	CONSTANT	визначає числову константу	12	+	+	+	+	+	+	+	+	+
190	службов.	TO	записує вершину стеку в числову константу	12	+	+	+	+	+	+	+	+	+
191	службов.	FCONSTANT	визначає числову математичну константу	12	+	+	+	+	+	+	+	+	+
192	службов.	TOF	записує вершину мат. стеку в числову математичну константу	12	+	+	+	+	+	+	+	+	+
193	службов.	ARRAY	визначає масив числових констант	13					+	+	+	+	
194	службов.	SET	записує вершину стеку в елемент масиву	13					+	+	+	+	
195	службов.	FINDINDEX	пошуку елемента масиву рівного певному значенню	13					+	+	+	+	
196	службов.	IF	слово умовного оператора керування виконанням алгоритму	19	+	+	+	+	+	+	+	+	+
197	службов.	ELSE	слово умовного оператора керування виконанням алгоритму	19	+	+	+	+	+	+	+	+	+
198	службов.	THEN	слово умовного оператора керування виконанням алгоритму	19	+	+	+	+	+	+	+	+	+
199	службов.	EXECUTE	виконання адреси з вершини стеку	23	+	+	+	+	+	+	+	+	+
200	службов.	FIND	кладе на стек виконавчу адресу слова з вихідного буфера	23	+	+	+	+	+	+	+	+	+
201	службов.	NAME	відображення у вих. буфері слова, заданого адресою	21	+	+	+	+	+	+	+	+	+
202	службов.	WORDS	відображення на екрані в термінальному режимі словника	11	+	+	+	+	+	+	+	+	+
203	службов.	FORGET	видалення слова зі словника	10	+	+	+	+	+	+	+	+	+
204	службов.	ERASE	видалення всіх слів зі словника	11				+	+	+	+	+	
205	службов.	UNUSED	кладе на стек вільне місце в словнику в байтах	11	+	+	+	+	+	+	+	+	+
206	службов.	RESTART	перезапуск системи	84	+	+	+	+	+	+	+	+	+
207	службов.	CLEARSYS	очистка системи	84	+	+	+	+	+	+	+	+	+
208	службов.	STOP	слово, яке нічого не виконує (ну дуууже корисне!)	22	+	+	+	+	+	+	+	+	+
209	службов.	STOPALL	зупиняє всі таймери	84	+	+	+	+	+	+	+	+	+
210	службов.	TIMER!	запуск таймера, який після заданого часу виконає задане слово	20	+	+	+	+	+	+	+	+	+

№	Група	Слово	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
211	службов.	TIMER?	кладе на стек адресу слова, яке буде виконане таймером	21	+	+	+	+	+	+	+	+	+
212	службов.	AUTOSPACE	режим відображення у вих. буфері з автоматичним пробілом	7	+	+	+	+	+	+	+	+	+
213	службов.	NOAUTOSPACE	режим відображення у вих. буфері без автоматичного пробілу	7	+	+	+	+	+	+	+	+	+
214	службов.	FPREC!	встановлення точності відображення мат. стеку	6	+	+	+	+	+	+	+	+	+
215	службов.	VERSION	відображення у вих. буфері версії прошивки	11	+	+	+	+	+	+	+	+	+
216	службов.	SYSINFO	відображення у вих. буфері розширеної версії прошивки	37					+	+	+	+	+
217	службов.	(	початок коментаря кінець або літера) або кінець рядка вводу	9	+	+	+	+	+	+	+		+
218	службов.	.FS	неруйнівне відображення на терміналі всього мат. стеку	84	+	+	+	+	+	+	+	+	+
219	службов.	.S	неруйнівне відображення на терміналі всього стеку	84	+	+	+	+	+	+	+	+	+
220	службов.	BREAKPOINT	точка зупинки форт-системи	84	+	+	+	+	+	+	+	+	+
221	службов.	DEBUGLEVEL	встановлює рівень відладки для точок зупинки	85	+	+	+	+	+	+	+	+	+
222	службов.	ERRORLEVEL	встановлює рівень відображення помилок	85	+	+	+	+	+	+	+	+	+
223	службов.	WATCHPOINT	точка спостереження форт-системи	85					+	+	+	+	+
224	службов.	MESSAGE	встановлює зміст буфера повідомлень	84					+	+	+	+	+
225	службов.	MESSAGE.	друкує у вих. буфері зміст буфера повідомлень	84					+	+	+	+	+
226	конфігур.	BOOT	встановлює скрипт автозапуску системи	40	+	+	+	+	+	+	+	+	+
227	конфігур.	BOOT.	друкує у вих. буфері скрипт автозапуску системи	41	+	+	+	+	+	+	+	+	+
228	конфігур.	CALBAT	калібровка акумуляторної батареї	39	+		+						
229	конфігур.	CALI	калібровка комбінованого входу по струму	39	+		+		+	+	+	+	
230	конфігур.	CALPOW	калібровка напруги живлення	39	+								
231	конфігур.	CALV	калібровка комбінованого входу по напрузі	39	+		+						
232	конфігур.	CALZ	калібровка нуля комбінованого входу	39					+	+	+	+	
233	конфігур.	PASSWORD	встановлює системний пароль	40	+		+		+	+	+	+	+
234	конфігур.	PROTECTPARAM	повертає на стек параметри захисту системи	40			+		+	+	+	+	+
235	конфігур.	PASSWORD.	друкує у вих. буфері системний пароль	40			+		+	+	+	+	+
236	конфігур.	DISABLE	системна константа для слова PASSWORD	40	+		+		+	+	+	+	+
237	конфігур.	PROTECT_BY	системна константа для слова PASSWORD	40	+		+		+	+	+	+	+
238	конфігур.	CONTROL	встановлює права доступу для дистанційного режиму	39			+		+			+	
239	конфігур.	CONTROLPARAM	повертає на стек параметри доступу до дистанційного режиму	39			+		+			+	
240	конфігур.	FOR_ALL	системна константа для слова CONTROL	39			+		+			+	
241	конфігур.	FOR_LOYAL	системна константа для слова CONTROL	39			+		+			+	
242	конфігур.	LOCAL	системна константа для слова CONTROL	39			+		+			+	
243	конфігур.	REMOTE	системна константа для слова CONTROL	39			+		+			+	
244	конфігур.	PIN	встановлює пін-код	40			+		+			+	
245	конфігур.	PIN.	друкує у вихідний буфер пін-код	40			+		+			+	
246	конфігур.	DIAI	встановлює параметри комбінованого входу	38	+		+						
247	конфігур.	SET_TO_D	системна константа для слова DIAI	38	+		+						
248	конфігур.	SET_TO_I	системна константа для слова DIAI	38	+		+						
249	конфігур.	SET_TO_V	системна константа для слова DIAI	38	+		+						
250	конфігур.	DIAIPARAM	повертає на стек параметри комбінованого входу	39	+		+						
251	конфігур.	SETWATCH	встановлює параметри системного годинника	37	+		+		+	+	+	+	+
252	конфігур.	SUMMER_OFF	системна константа для слова SETWATCH	37	+		+		+	+	+	+	+
253	конфігур.	SUMMER_ON	системна константа для слова SETWATCH	37	+		+		+	+	+	+	+
254	конфігур.	WATCH!	встановлює системний годинник	38	+	+	+	+	+	+	+	+	+
255	конфігур.	WATCHPARAM	повертає на стек параметри системного годинника	38			+		+	+	+	+	+
256	конфігур.	ETHIP	встановлює власну IP-адресу	41						+			
257	конфігур.	ETHIP.	друкує у вихідний буфер власну IP-адресу	41						+			
258	конфігур.	ETHGATEWAY	встановлює IP-адресу основного шлюзу	41						+			
259	конфігур.	ETHGATEWAY.	друкує у вихідний буфер IP-адресу основного шлюзу	41						+			
260	конфігур.	ETHMASK	встановлює маску підмережі	41						+			
261	конфігур.	ETHMASK.	друкує у вихідний буфер маску підмережі	41						+			
262	конфігур.	MBTCPPTORT!	встановлює з вершини стеку порт сервера MODBUS TCP	41						+			
263	конфігур.	MBTCPPTORT?	кладе на стек порт сервера MODBUS TCP	41						+			
264	рядки.	."	відображення у вих. буфері рядка	15	+	+	+	+	+	+	+	+	+
265	рядки	FLUSH	очищення вихідного буфера	15	+	+	+	+	+	+	+	+	+
266	рядки	LENGTH	кладе на стек довжину рядка тексту у вих. буфері	15		+	+	+	+	+	+	+	+
267	рядки	NEWLINE	перенесення тексту у вих. буфері на новий рядок	8		+	+	+	+	+	+	+	+
268	рядки	SPACE	відображення у вих. буфері пробілу	7	+	+	+	+	+	+	+	+	+
269	рядки	QUOTE	відображення у вих. буфері лапок	15	+	+	+	+	+	+	+	+	+
270	рядки	STRING!	записує у рядкову змінну текст з вихідного буфера	15		+	+	+	+	+	+	+	+
271	рядки	STRING?	кладе у вихідний буфер рядок тексту з рядкової змінної	15		+	+	+	+	+	+	+	+
272	рядки	STR>VAR	переносить символи з вихідного буфера у цілочисельні змінні	35					+	+	+		
273	мат.опер.	F-	віднімання на математичному стеку	8	+	+	+	+	+	+	+	+	+
274	мат.опер.	F*	множення на математичному стеку	8	+	+	+	+	+	+	+	+	+
275	мат.опер.	F**	функція піднесення до степені на математичному стеку	8	+	+	+	+	+	+	+	+	+
276	мат.опер.	F.	відобр. у вих. буфері вершини мат. стеку в форматі з фікс. комою	6	+	+	+	+	+	+	+	+	+
277	мат.опер.	F/	ділення на математичному стеку	8	+	+	+	+	+	+	+	+	+
278	мат.опер.	F+	додавання на математичному стеку	8	+	+	+	+	+	+	+	+	+
279	мат.опер.	F<	логічна операція МЕНШЕ на математичному стеку	17	+	+	+	+	+	+	+	+	+
280	мат.опер.	F=	логічна операція РІВНО на математичному стеку	17					+	+	+	+	
281	мат.опер.	F>	логічна операція БІЛЬШЕ на математичному стеку	17	+	+	+	+	+	+	+	+	+
282	мат.опер.	F<=	логічна операція МЕНШЕ РІВНО на математичному стеку	17					+	+	+	+	
283	мат.опер.	F<=	логічна операція НЕ РІВНО на математичному стеку	17					+	+	+	+	
284	мат.опер.	F>=	логічна операція БІЛЬШЕ РІВНО на математичному стеку	17					+	+	+	+	
285	мат.опер.	FABS	отримання абсолютного значення на математичному стеку	8	+	+	+	+	+	+	+	+	+
286	мат.опер.	FACOS	функція арккосинус на математичному стеку	8	+	+	+	+	+	+	+	+	+
287	мат.опер.	FASIN	функція арксинус на математичному стеку	8	+	+	+	+	+	+	+	+	+
288	мат.опер.	FATAN	функція арктангенс на математичному стеку	8	+	+	+	+	+	+	+	+	+
289	мат.опер.	FCOS	функція косинус кута на математичному стеку	8	+	+	+	+	+	+	+	+	+
290	мат.опер.	FCOSH	функція косинус гіперболічний на математичному стеку	8	+	+	+	+	+	+	+	+	+
291	мат.опер.	FDEPTH	кладе на стек кількість елементів мат. стеку	6	+	+	+	+	+	+	+	+	+
292	мат.опер.	FDROP	відкидання вершини мат. стеку	5	+	+	+	+	+	+	+	+	+
293	мат.опер.	FDUP	дублювання вершини мат. стеку	5	+	+	+	+	+	+	+	+	+

№	Група	Слово	Короткий опис	Стор.	B	D	G	повна	SG	SE	SB	ST	M
294	мат. опер.	FE	відображення у вих. буфері вершини мат. стеку в наук. форматі	6	+	+	+	+	+	+	+	+	+
295	мат. опер.	FEXP	функція експонента на математичному стеку	8	+	+	+	+	+	+	+	+	
296	мат. опер.	FLN	функція логарифм натуральний на математичному стеку	8	+	+	+	+	+	+	+	+	
297	мат. опер.	FLOG	функція логарифм десятковий на математичному стеку	8	+	+	+	+	+	+	+	+	
298	мат. опер.	FNEGATE	зміна знаку числа на математичному стеку	8	+	+	+	+	+	+	+	+	
299	мат. опер.	FOVER	дублюв. значення мат. стеку під вершиною на вершину мат. стеку	5	+	+	+	+	+	+	+	+	+
300	мат. опер.	FPICK	дублювання довільного значення мат. стеку на вершину мат. стеку	6	+	+	+	+	+	+	+	+	+
301	мат. опер.	FROLL	циклічна ротація довільної кількості верхніх значень матем. стеку	6	+	+	+	+	+	+	+	+	+
302	мат. опер.	FROT	циклічна ротація трьох верхніх значень мат. стеку	5	+	+	+	+	+	+	+	+	+
303	мат. опер.	FSIN	функція синус кута на математичному стеку	8	+	+	+	+	+	+	+	+	
304	мат. опер.	FSINH	функція синус гіперболічний на математичному стеку	8	+	+	+	+	+	+	+	+	
305	мат. опер.	FSQRT	функція корінь квадратний на математичному стеку	8	+	+	+	+	+	+	+	+	
306	мат. опер.	FSWAP	перестановка двох верхніх значень мат. стеку	5	+	+	+	+	+	+	+	+	+
307	мат. опер.	FTAN	функція тангенс кута на математичному стеку	8	+	+	+	+	+	+	+	+	
308	мат. опер.	FTANH	функція тангенс гіперболічний на математичному стеку	8	+	+	+	+	+	+	+	+	
309	мат. опер.	FVAR!	записує в математичну змінну значення з мат. стеку	14	+	+	+	+	+	+	+	+	+
310	мат. опер.	FVAR?	кладе на мат. стек значення математичної змінної	14	+	+	+	+	+	+	+	+	+
311	цілі опер.	-	операція віднімання	8	+	+	+	+	+	+	+	+	+
312	цілі опер.	*	операція множення	8	+	+	+	+	+	+	+	+	+
313	цілі опер.	.	відображення у вих. буфері вершини стеку	5	+	+	+	+	+	+	+	+	+
314	цілі опер.	/	операція ділення з відкиданням залишку	8	+	+	+	+	+	+	+	+	+
315	цілі опер.	+	операція додавання	8	+	+	+	+	+	+	+	+	+
316	цілі опер.	<	логічна операція МЕНШЕ	17	+	+	+	+	+	+	+	+	+
317	цілі опер.	<=	логічна операція МЕНШЕ РІВНО	17	+	+	+	+	+	+	+	+	+
318	цілі опер.	<>	логічна операція НЕ ДОРІВНЮЄ	17	+	+	+	+	+	+	+	+	+
319	цілі опер.	=	логічна операція ДОРІВНЮЄ	17	+	+	+	+	+	+	+	+	+
320	цілі опер.	>	логічна операція БІЛЬШЕ	17	+	+	+	+	+	+	+	+	+
321	цілі опер.	>=	логічна операція БІЛЬШЕ РІВНО	17	+	+	+	+	+	+	+	+	+
322	цілі опер.	ABS	операція отримання абсолютного значення	8	+	+	+	+	+	+	+	+	+
323	цілі опер.	AND	операція логічного І	16	+	+	+	+	+	+	+	+	+
324	цілі опер.	DEPTH	кладе на стек кількість елементів стеку даних	6	+	+	+	+	+	+	+	+	+
325	цілі опер.	DROP	відкидання вершини стеку	5	+	+	+	+	+	+	+	+	+
326	цілі опер.	DUP	дублювання вершини стеку	5	+	+	+	+	+	+	+	+	+
327	цілі опер.	FALSE	кладе на стек логічне значення НЕ ІСТИНА	16	+	+	+	+	+	+	+	+	+
328	цілі опер.	FLAG!	записує в бітову змінну значення зі стеку	14	+	+	+	+	+	+	+	+	+
329	цілі опер.	FLAG?	кладе на стек значення бітової змінної	14	+	+	+	+	+	+	+	+	+
330	цілі опер.	ISNOW	попарно порівнює шість значень на вершині стеку	48	+	+	+	+	+	+	+	+	+
331	цілі опер.	LSHIFT	операція логічного зсуву ліворуч	17	+	+	+	+	+	+	+	+	+
332	цілі опер.	MOD	операція ділення з отриманням залишку	8	+	+	+	+	+	+	+	+	+
333	цілі опер.	NEGATE	операція зміни знаку числа	8	+	+	+	+	+	+	+	+	+
334	цілі опер.	NOT	операція логічного ЗАПЕРЕЧЕННЯ	16	+	+	+	+	+	+	+	+	+
335	цілі опер.	OR	операція логічного АБО	16	+	+	+	+	+	+	+	+	+
336	цілі опер.	OVER	дублюв. значення під вершиною стеку на вершину стеку	5	+	+	+	+	+	+	+	+	+
337	цілі опер.	PICK	дублювання довільного значення стеку на вершину стеку	5	+	+	+	+	+	+	+	+	+
338	цілі опер.	ROLL	циклічна ротація довільної кількості верхніх значень на стеку	5	+	+	+	+	+	+	+	+	+
339	цілі опер.	ROT	циклічна ротація трьох верхніх значень на стеку	5	+	+	+	+	+	+	+	+	+
340	цілі опер.	RSHIFT	операція логічного зсуву праворуч	17	+	+	+	+	+	+	+	+	+
341	цілі опер.	SWAP	перестановка двох верхніх значень стеку	5	+	+	+	+	+	+	+	+	+
342	цілі опер.	TRUE	кладе на стек логічне значення ІСТИНА	16	+	+	+	+	+	+	+	+	+
343	цілі опер.	VAR!	записує в змінну значення зі стеку	13	+	+	+	+	+	+	+	+	+
344	цілі опер.	VAR?	кладе на стек значення змінної	13	+	+	+	+	+	+	+	+	+
345	цілі опер.	XOR	операція логічного ВИКЛЮЧНОГО АБО	16	+	+	+	+	+	+	+	+	+
346	пер. типів	D>F	перенесення вершини стеку даних на вершину мат. стеку	6	+	+	+	+	+	+	+	+	+
347	пер. типів	F>D	перенесення вершини мат. стеку на вершину стеку даних	6	+	+	+	+	+	+	+	+	+
348	пер. типів	F>US	перетворення "математичний формат у беззнаковий"	35	+	+	+	+	+	+	+		
349	пер. типів	US>F	перетворення "беззнаковий формат у математичний"	35	+	+	+	+	+	+	+		
350	пер. типів	US>DF	перетворення "беззнаковий формат у подвійний математичний"	35					+	+	+		
351	пер. типів	DF>US	перетворення "подвійний математичний формат у беззнаковий"	35					+	+	+		
352	пер. типів	US>S	перетворення "беззнаковий формат у знаковий"	34	+	+	+	+	+	+	+		

Внесені зміни та доповнення

Версія документу 2.0

- початок реєстрації змін та доповнень для єдиного документу "Настанова з програмування на мові ForthLogic™"

Версія документу 2.1

- додано розділ "Передача даних CSD";
- в розділ "DTMF сигнали" додано опис слів DTMFCONFIRM, NODTMFCONFIRM;
- в параграф "Керування системою" розділу "Система" додано опис слова WATCH!;
- з параграфу "Системний час" розділу "Система" вилучено опис слова UTC!

УВАГА! ДАНЕ СЛОВО ВИЛУЧЕНО З ПОДАЛЬШИХ ВЕРСІЙ ФОРТ-СИСТЕМИ!

Версія документу 2.3

- додано розділ "Передача даних GPRS";
- додано розділ "Відтворення повідомлень через акустичну систему";
- в додаток додано розділ "Набір файлів чисел";
- в розділі "Послідовний порт RS485" доповнено опис слова MODBUSPARAM;
- в розділ "Керування системою" додано опис слів WATCHPARAM, CALZ, CONTROLPARAM, PROTECTPARAM, PASSWORD. (пароль-крапка), PIN. (пін-крапка);
- в розділ "Системний час" додано опис слів DST?, TIMESTAMP;
- в розділі "Відтворення повідомлень" доповнено опис слова SAY;
- в розділ "Відтворення повідомлень" додано опис слів AUTOSAYPLUS, NOAUTOSAYPLUS;
- в розділ "Передача даних CSD" додано опис слова MODEMCSD;
- в розділ "Реєстратор" додано опис слів TO_TCP та LOG>TCP;
- в таблиці версій додано стовпчик версії контролера ES-ForthLogic SG

Версія документу 2.4

- в розділі "Послідовний порт RS485" доповнено опис слова MODBUSTIMEOUT!;
- в розділі "Передача даних GPRS" додано опис слів M2MTCPON, M2MTCPOFF, ATTACHGPRS та доповнено опис слів SETIP, SETAPN;
- в розділі "Реєстратор" додано опис слів USER_MODE, LOGSEND?, LOGFILE? Та доповнено опис слів LOG?, LOG;
- в розділі "Створення програм" змінено опис вірця файлу-скрипта;
- в розділі "Відлагодження програм" додано опис слова CLEARSYS;
- в розділі "Помилки мови ForthLogic™" додано опис слова ERRORLEVEL;

Версія документу 2.5

- в розділі "Система" додано опис команди RESTORE DEFAULTS;
- в розділі "Константи, рядки та змінні" додано опис слова QUOTE;
- в розділі "Реєстратор" доповнено опис слова LOGFILE?;
- в розділі "Керування системою" доповнено опис слів SETWATCH та PASSWORD;
- в розділі "Керування системою" додано опис слова DIAIPARAM;

Версія документу 2.6

- в розділі "Реєстратор" додано опис слова LOG>DB та доповнено опис процесу реєстрації;

- розділ "Акустична система" перейменовано на "Гарнітура" та доповнено;
- в розділі "Передача даних GPRS" додано опис слів ID, URL, SETID, SETURL, DNSMODE, IPMODE, CLIENTDB, CLIENTOPC, STATUSOPC

Версія документу 2.7

- додано новий розділ "Індикація"
- додано новий розділ "Ідентифікація системи"
- додано новий розділ "Модуль ETHERNET"
- доповнено розділ "Створення файлів-скриптів на мові ForthLogic™"
- в розділі "Константи, рядки та змінні" додано опис слів ARRAY, SET, FINDINDEX
- в розділі "Відлагодження програм" додано опис слів MESSAGE, MESSAGE, WATHCPOINT та змінено синтаксис слова BREAK на BREAKPOINT.
- в розділі "Системний час" додано опис слів SUNRISE?, SUNSET?, CIVIL, OFFICIAL
- в розділі "Послідовний порт RS485" додано опис слів MODBUSRESET, US>DF, DF>US, STR>VAR
- в розділі "Слова для роботи з GPRS" додано опис слів CLIENTSQL
- в розділі "Реєстратор" додано опис слів TO_RAM, LOG>SQL, LOG>DBTCP, LOGTERMINON, LOGTERMINOFF, LOGFILETOGGLE, LOGFILEDELETE
- в розділі "Введення нових слів" додано опис слів ERASE
- в розділі "Логічні операції" додано опис слів F=, F<=, F<>, F>=
- в розділі "Стан мережі GSM" додано опис слів MODULE?, SIM?
- в розділі "Входи" додано опис слів WIEGANDSTART, WIEGANDSTOP, COUNTERSTART, COUNTERSTOP, COUNTERGET
- в розділі "Виходи" додано опис слів PWMSTART, PWMSTOP
- в розділі "Керування системою" додано опис слів ETHIP, ETHGATEWAY, ETHMASK, ETHIP., ETHGATEWAY., ETHMASK., MBTCPPOINT!, MBTCPPOINT?
- в додаток "Апаратна платформа мови ForthLogic" додано опис нових контролерів;
- в додаток "Вбудовані слова мови ForthLogic" додано опис нових контролерів;