**Системные функции. Вводная часть.**

Ниже приведено описание всех системных функций. С точки зрения Lua, это внешние глобальные функции. Поэтому синтаксические ошибки в именах функций не проверяются. В случае если имени функции написано неверно, при выполнении LUA приложения возникнет ошибка исполнения. Ошибка и имя функции, на которую не была найдена ссылка, отобразиться в телеметрических данных системы.

При использовании системных функций, стоит следить за типом данных передаваемых как параметры. Система контролирует только выход параметров за допустимые диапазоны в рамках типа данных. Однако для повышения производительности системы, проверка типа данных не производится. Поэтому, в случае если в качестве параметра необходимо передать целое беззнаковое, а передастся число с плавающей точкой или отрицательное число, произойдёт автоматическое преобразование типа данных по правилам ядра LUA. Результат этого преобразования может быть не предсказуем. Речь идет прежде всего о параметрах системных функций, когда данные передаются за пределы Lua программы. Внутри самой Lua программы преобразования работают корректно.

Все системные функции, если это не указано отдельно, являются функциями немедленного действия. Они приостанавливают выполнения прикладной программы на время их выполнения ядром системы.

**Функции общего назначения.**

**SYSTEM\_RESTASRT()**

Функция перезапуска системы. Перезапуск будет выполнен после завершения текущего цикла прикладной программы. Т.е. сначала будет выполнен весь код от SYSTEM\_RESTASRT() до Yield. При этом буду применены новые значения выходных сигналов и только после этого произойдет перезапуск. При перезапуске все выходные каналы будут выключены, после этого произойдет полная реинициализация системы, включая выгрузку прикладной программы из памяти.

**возвр1 SetEEPROM( пра1, пар2)**

Функция для записи данных в EEPROM. Система поддерживает адресацию до 409 4-х байтных объектов. Сохранять можно следующие Lua типы данных: float, integer, boolean. При записи переменной в память сохраняется ее тип. При чтении из памяти, возвращается переменная того же типа, который был при записи.

Скорость записи данных значительно ниже общего быстродействие системы, поэтому для того, что не снижать быстродействие алгоритма, не рекомендуется производить запись без необходимости. EEPROM в первую очередь предназначен для хранения уставок и/или записи ошибок, возникающие в процессе эксплуатации.

Если необходимо организовать байтовый доступ к EEPROM, то запаковку данных необходимо делать не прикладном уровне. Байтовый доступ не был реализован принципиально. Это сокращает риск потери данных не квалифицированным прикладным программистом, поскольку разбивка float LUA переменной на байты и сборка ее обратно, требует большой аккуратности. А если уж такая задача решена, то проблем организовать байтовый доступ к 4-х байтовым виртуальным секторам EEPROM не составит большой сложности.

***пар1. -обязательный, при отсутствии функция игнорируется.***

*Значение параметра* : адрес в памяти

*Допустимые значения:* от 0 до 408.

*Тип данных*: беззнаковое целое

***пар2 – обязательный, при отсутствии функция игнорируется.***

*Значение параметра* : данные для записи

*Допустимые значения:* любая переменная или константа типа float, integer, boolean.

*Тип данных*: float, integer, Boolean

***Возвр1– возвращаемое значение.***

1 – если запись прошла успешно

0 – ошибка записи

**возвр1 GetEEPROM( пар1)**

Функция чтения данных из EEPROM с кэшированием. При вызове функция проверяет системный кэш, и если в нем данные отличны от нуля, то возвращает их обратно, без непосредственного чтения EEPROM. После старта системы, весь кэш нулевой. При любой записи и чтении данные (если это не 0), дублируются по соответствующему адресу в кэш.

*пар1. -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : адрес в памяти

*Допустимые значения:* от 0 до 408.

*Тип данных*: беззнаковое целое

*Возвр1– возвращаемое значение.*

*Значение параметра* : данные по соответствующему адресу.

*Тип данных*: float, integer, Boolean, в зависимости от хранимого типа данных.

**Функции работы с выходными каналами.**

**setOutConfig( пар1, пар2, <пар3>,<пар4>,<пар5>)**

Функция конфигурация канала. Если не вызвать setOutConfig, то канал будет выключен на уровне ядра и не будет реагировать ни на какие команды

*пар1. -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : номер канала

*Допустимые значения:* от 1 до 20.

*Тип данных*: беззнаковое целое

*пар2 – обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : номинальный ток.

*Допустимые значения:* Для каналов 1-8 до 30.0А, для каналов 9-20 до 15.0А.

*Тип данных*: положительное c плавающей точкой. Кол-во знаков после запятой ограничено представление вещественных числе в LUA. Точность срабатывания уставки зависит от стабильности питания и потребителя. В общем случае рекомендуется исходить из возможных колебаний тока потребления нагрузки в пределах +-0,5А.

*пар3. – необязательный, значение по умолчанию - 1*

*Значение параметра* : сброс перегрузки выключение канала. При 0, ошибка на канале сбрасывается только рестартом системе. При 1, выключение канала из прикладной программы.

*Допустимые значения:* 0 или 1.

*Тип данных*: беззнаковое целое

.

*Пар4. – необязательный, значение по умолчанию - 0*

*Значение параметра* : время работы в перегрузке в мс

*Допустимые значения:* от 0 до 32 000

*Тип данных*: беззнаковое целое

*Пар5. – необязательный, значение по умолчанию – номинальный ток*

*Значение параметра* : максимальный ток перегрузки

*Допустимые значения:* Для каналов 1-8 до 120.0А, для каналов 9-20 до 40.0А.

*Тип данных*: положительное c плавающей точкой. Кол-во знаков после запятой ограничено представлением чисел c плавающей точкой в LUA. Точность срабатывания уставки зависит от стабильности питания и потребителя. В общем случае рекомендуется исходить из возможных колебаний тока потребления нагрузки в пределах +-0,5А.

**OutResetConfig(пар1, пар2, пар3)**

Конфигурация поведения канала при перегрузке. Если функция не вызвана и канал сконфигурирован, то при перегрузке канал уходит в ошибку.

***пар1. -обязательный, при отсутствии функция игнорируется.***

*Значение параметра*: номер канала

*Допустимые значения:* от 1 до 20.

*Тип данных*: беззнаковое целое

***пар2 – обязательный, при отсутствии функция игнорируется***

*Значение параметра*: кол-во попыток запуска

*Допустимые значения:* 0 – бесконечное кол-во пере запусков, 1, выключается при перегрузке, 2- две попытки включения, 3- три попытки включения. Максимальное кол-во 255

Имеется ввиду, что 2 попытки включения, это включение и один перезапуск в случае ошибки.

*Тип данных*: беззнаковое целое

***пар3. – обязательный, при отсутствии функция игнорируется.***

*Значение параметра* : таймаут перед очередным перезапуском в мс.

*Допустимые значения:* от 0 до 65 535

*Тип данных*: беззнаковое целое

**setPWMGroupeFreq(пар1, пар2)**

Установка частоты ШИП для группы каналов. Менять частоту возможно только для группы каналов, после вызова функции для всех каналов группы будет доступен ШИМ на указанной частоте. *Работа ШИМ на высокой частоте может быть не стабильна из-за помех от нагрузки и/или линиях подключения. Частоты выше 1кГц не рекомендуется использовать длительное время. Они подходят только для механизмов плавного пуска. Длительная работа выходных каналов на высоких частотах приведет к перегреву выходных ключей и срабатыванию аппаратной защиты.*

*пар1. -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : номер группы каналов

*Допустимые значения:*

0– каналы 5,6,9,10

1 – каналы 11,12,16

2 – каналы 4,15

3 -каналы 1,2,8,20

4 – каналы 13,14,17,18

5 -каналы 7 и 19

*Тип данных*: беззнаковое целое

*пар2 – обязательный, при отсутствии функция игнорируется*

*Значение параметра* : частота ШИМ в ГЦ

*Допустимые значения:* от 1 до 6000 Гц

*Тип данных*: беззнаковое целое

**OutSetPWM( пар1, пар2)**

Установка скважность ШИМ сигнала для выбранного канала. Для того, что бы функция работала, необходимо что бы бал установлена частота ШИМ для соответствующей группы каналов и канал бы включен. Если канал выключен, то ШИМ на новой скважности запустился только после включения. При не установленной частое группы каналов, функция игнорируется.

Одновременная работа setOutSoftStart и OutSetPWM на текущий момент не предусмотрена. OutSetPWM будет мешать механизму плавного пуска.

*пар1. -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : номер канала

*Допустимые значения:* от 1 до 20.

*Тип данных*: беззнаковое целое

*пар2 – обязательный, при отсутствии функция игнорируется*

*Значение параметра* : скважность ШИМ сигнала в %

*Допустимые значения:* 0 – 100. Во избежание ложных срабатывай защиты, рекомендуется менять параметр при включенном канале до 99%. Токовая защита при ШИМ сигнале работает по своему алгоритму. Переход на включенном канале с 50% на 100% фактический меняет алгоритм обработки на лету, что при определённых типах нагрузок может привести к ложному срабатыванию токовой защиты.

*Тип данных*: беззнаковое целое

**setOutSoftStart( пар1, пар2, пар3)**

Функция конфигурирует режим плавного пуска. В течении времени пуска, пар2, система линейно поднимает скважность ШИМ от значения, заданного пар3, до 100%. Контролируемое значение тока равоно пар5 функции setOutConfig. Для того, что бы функция работала, необходимо что бы до ее вызова, бала установлена частота ШИМ для соответствующей группы каналов.

Одновременная работа setOutSoftStart и OutSetPWM на текущий момент не предусмотрена. OutSetPWM будет мешать механизму плавного пуска.

*пар1. -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : номер канала

*Допустимые значения:* от 1 до 20.

*Тип данных*: беззнаковое целое

*пар2 – обязательный, при отсутствии функция игнорируется*

*Значение параметра* : длительность плавного пуска в мс

*Допустимые значения:* 0 – 65535.

*Тип данных*: беззнаковое целое

*Пар3. -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : скважность ШИМ сигнала в % при старте.

*Допустимые значения:* 0 – 99.

*Тип данных*: беззнаковое целое

**Функции CAN**

Все функции CAN, как и остальные системные функции, выполняются при вызове. Однако они передают/получают данные не непосредственно из CAN шин. Работает многоступенчатая система, позволяющая максимально распараллелить обмен по CAN шине и работу прикладного алгоритма.

При отправке данных из Lua в CAN, фрейм записываются в системную очередь. Дальше системный драйвер по мере физической отправки пакетов раскидывает очередь в аппаратные FIFO буфера микропроцессора, который пытается выпихнуть их на шину. Такая многоступенчатая архитектура позволяет максимально снизить нагрузку на процессор и увеличить скорость выполнения прикладной программы. Таким образом если в одном цикле программы отправить несколько десятков фреймов, физический последний из пактов может уйти на шину уже на следующем цикле программы. Для передачи доступны пакеты как обычным, так и с расширенным CAN-ID. Для отправки расширенного ID необходимо установить старший бит адреса в 1. Проще всего битовым ИЛИ с системной константной EXT\_CAN\_ID ( например так - CanSend(EXT\_CAN\_ID | 0x0F000000 ,1,5)

Прием пакетов организован по принципу почтовых ящиков. Почтовый ящик при конфигурации привязывается к своему СAN-ID, после чего происходит настройка аппаратной фильтрации. Система принимает только пакеты с ID, из списка получено при конфигурации почтовых ящиков. Если фрейм проходит ID фильтр, то его данные записываются в соответствующий почтовый ящик. На почтовом ящике устанавливается флаг о новых данных. Если данные не прочитать до прихода нового пакета, предыдущие данные будут утеряны. Пользователю доступны до 55 уникальных ящиков. Если на 2 ящика будет установлен один и тот же ID, данные будут записаны в ящик, созданный раньше. Для приема не доступны расширенные CAN-ID.

Ящики должны быть настроены один раз, при старте системы. Если будут попытки конфигурации более 55 почтовых ящиков, такие попытки будут игнорироваться.

Помимо 55 почтовых ящиков для входящих CAN сообщений, реализован механизм запрос-ответ. В системе есть 56-й почтовый ящик, который конфигурируется отдельно. Он предназначен для ожидания ответа на запрос. Т.е. при вызове соответствующей функции, в сеть отправляется указный при вызове пакет и 56-й почтовый ящик конфигурируется на пример ответа с указанным при вызове ID. Как только ответ пришел, устанавливается флаг. Приход ответа из LUA можно проверять асинхронно, в любой момент времени. Таких вызовов “запрос-ответ” может быть бесконечное кол-во. При каждом вызове 56-й почтовый ящик настраивается на прием ответа с новым ID.

**CanSend( адр, д1, <д2>,<д3>,<д4>,<д5>,<д6>,<д7>,<д8>)**

Функция отбавляет 8 байтовый с ID = **адр**. Обязательно передавать хотя бы один байт данных – д1. д2-д8 по умолчанию заполняются 0.

*адр -обязательный, при отсутствии функция игнорируется.*

*Значение параметра* : CAN-ID

*Допустимые значения:* любое целое. Для отправки расширенного ID необходимо установить в 1 самый старший бит адреса, например так:  *CanSend(EXT\_CAN\_ID | 0x0F000000 ,1,5)*

*Тип данных*: беззнаковое целое

*д1 – обязательный, при отсутствии функция игнорируется*

*Значение параметра* : первый байт фрейма

*Допустимые значения:* проверка не проводится, просидит автоматическое преобразование в Lua integer, а потом в байт на уровне ядра. При передаче значения выходящего за пределы одного байта, скорее всего старшие 3 байта будут проигнорированы.

*Тип данных*: беззнаковое целое

*д2-д8 - необязательный, значение по умолчанию - 0*

*Аналогично д1.*

**СanTable ( адр, размер, lua таблица )**

Функция отправляет по адресу адр фрейм из размер байтов, данные содержаться в таблице. Пример использования:

*Data = { [1] = 0x23,[2]=0x34, [3]=0x45}*

*CanTable(0x40,3,Data)*

При исполнении функции, производится извлечение данных из таблицы с индексами от 1 до размер. Если размер будет больше чем, размер таблицы или таблица будет объявлена не как одномерный массив, то это вызовет ошибку исполнения Lua, как если бы подобное было в коде lua программы.

***адр -обязательный, при отсутствии функция игнорируется.***

*Aналогично функции CanSend*

***размер – обязательный, при отсутствии функция игнорируется***

*Значение параметра*: кол-во передаваемых байт в фрейме.

*Допустимые значения:* 0 – 8. Если больше 8, функция не выполняется.

*Тип данных*: беззнаковое целое

**Lua таблица *- обязательный, при отсутствии функция игнорируется.***

*Переменная –* одномерная lua таблица с числовыми ключами от 1 до 8, сдержавшая integer значения не более 255. Поведение при некоренных значениях аналогично функции CanSend.

**возвр1 setCanFilter( адр )**

Функция создаёт новый почтовый ящик в системе и настраивает фильтр на прием в него данных с CAN-ID =адр.

***адр -обязательный, при отсутствии функция игнорируется.***

*Значение параметра* : CAN-ID

*Допустимые значения:* любое целое. Расширенный CAN-ID не поддерживается.

*Тип данных*: беззнаковое целое

***Возвр1*** *– 0 если ящик успешно создан, 1 – если уже создано 55 почтовых ящиков.*

**возвр1 CheckCanID( aдр)**

*Функция проверяет если новые сообщения по адресу адр*

***адр -обязательный, при отсутствии функция игнорируется.***

*Аналогично setCanFilter*

***Возвр1*** *– 1 если есть новое сообщение, 0 – новых сообщений нет.*

*<***д1>,<д2>,<д3>,<д4>,<д5>,<д6>,<д7>,<д8> getCanMessage( aдр )**

*Если в системе есть новое сообщение с ID =* ***адр****, функция вернёт кол-во байт, соответствующие размеру сообщения. Например если пришло 3 байта во фрейме с ID = 0x15*

*B1,B2,B3,B4,B5,B6,B7,B8 = getCanMessage (0x15)*

*То результат будет следующий:*

*B1 – первый байт*

*B2 – второй байт*

*B3 – третий байт*

*B4– nil*

*B5– nil*

*B6– nil*

*B7– nil*

*B8– nil*

*Данную функцию лучше использовать в паре с CheckCanID*

***адр -обязательный, при отсутствии функция игнорируется.***

*Аналогично setCanFilter*

***Возвращаемые значения***

*Байт данных типа integer*

**Возвр1 GetCanToTable( aдр , lua таблица)**

*Если в системе есть новое сообщение с ID =* ***адр****, функция запишет новые данные в таблицу и вернет 1. Если новых данных нет, вернет 0, значения в таблице не изменяться. Если размер таблицы больше чем кол-во байт во фрейме, дополнительные элементы таблицы сохранят текущие значения.*

***адр -обязательный, при отсутствии функция игнорируется.***

*Аналогично setCanFilter*

**Lua таблица *- обязательный, при отсутствии функция игнорируется.***

Одномерная lua таблица с числовыми ключами от 1 до 8. Поведение при некоренных значениях аналогично функции CanSend.

***Возвр1*** *– 1 если есть новое сообщение, 0 – новых сообщений нет.*