Работа с интерпретатором Fr-Python.

Фирма Фрактал
Москва Зеленоград
www.fractal.com.ru
fractal@aha.ru
(495) 978-12-86
(499) 710-12-60

Последние изменения.

30.09.2013 v0.70

Первая доступная для тестирования версия Fr-Python.
Версия совместима со всеми модулями ООО "Фрактал" в составе которых есть микроконтроллер STM32F103.
Процедура программирования интерпретатора в модуль полностью аналогична программированию обновлений Fractal-BASIC. Необходимо до подачи питания на модуль , надеть джампер отвечающий за переход в режим загрузчика Boot-loader Fractal . После запуска утилиты AppSend.exe, она, взаимодействуя с Boot-loader Fractal прошитый в модуле, загружает очередную версию интерпретатора. При таком способе обновлений, пользователь, при желании, всегда может вернуться к предыдущей версии, использовав соответствующий архив. В том числе, можно вернуться и к интерпретатору Fractal-BASIC.

Для работы с Fr-Python необходим Fr-Terminal версии от 1.12. Терминал универсален и предназначен для работы со всем спектром модулей Фрактал. Поддерживается работа в посимвольном режиме с Fractal-BASIC, работа в пакетном режиме с модулями по протоколу MODBUS и работа с Fr-Python.
Для корректной работы с Fr-Python необходимо до запуска терминала подключить модуль к USB и после запуска выбрать номер COM-порта соответствующего модулю, а так же переключить режим терминала в "Python".
При последующих запусках терминала он будет самостоятельно применять выбранные ранее параметры.

Терминал имеет два основных состояния - собственно терминал и редактор программного кода.
При использовании терминала, ввод команд должен производится через командную строку. Под правую кнопку доступно выпадающее контекстно-зависимое меню с набором часто используемых команд.

Если все правильно подключено и в модуле отсутствует ранее записанная Python-программа, то при нажатии "Enter" в командной строке терминала модуль ответит ">>>". Это означает что модуль готов работать в режиме командной строки и выполнять команды.
Например:

```
>>> 2 * 2
4
>>> a = 'qwerty'
>>> b = 3
>>> print a * b
qwertyqwertyqwerty
>>> for i in a:
...     print i
...
q
w
e
r
t
y
>>>
```

Ну и т.д.
Не забываем про правила форматирования Python (отступ = 4 пробела).

Для написания программы (модуля "main") нужно переключиться в редактор. Здесь можно набрать текст, сохранить его и загрузить в модуль. Либо выбрать уже готовый файл для редактирования и/или загрузки в модуль.
Команды доступны в выпадающем под правую кнопку мыши меню.
После загрузки программы в модуль состояние оболочки автоматически переходит в терминал.
Запустить выполнение программы можно выбрав в меню пункт " Reset", или набрав в командной строке:

```
>>> import main
```

Если программа не предусматривает зацикливание , то после завершения она вернется к приглашению ">>>".
Если программа зациклена или выполняется долго, то при необходимости ее можно остановить "Ctrl-C" или из меню.

Алгоритм работы модуля после подачи питания или сброса:
- если есть ранее загруженная программа в модуль, то она запускается;
- если программы нет, запускается режим командной строки;
- если программа выполнена, то запускается командная строка;
- если программа была остановлена пользователем, запускается командная строка.

В интерпретаторе предустановлен библиотечный модуль "fr". Модуль содержит набор функций для удобной работы с многими узлами микроконтроллера и внешними модулями, подключаемыми по последовательным интерфейсам и линиям ввода/вывода. Этот программный модуль постоянно обновляется и дополняется.

Примеры работы с модулем "fr" из командной строки:

```
>>> from fr import *
```

Измерим напряжение на линии 3(измерение будет в вольтах, первое измерение на линии даст результат 0.0)

```
>>> adc(3)
1.617773
```

Выдадим логический ноль на линию 17

```
>>> pin(17, 0)
```

Считаем состояние линии 25

```
>>> pin(25)
1
```

Инициализируем 3 таймер для работы с ШИМ с периодом 65535мкс

```
>>> pwm(3, 65535)
```

Выдадим на 4-й канал таймера 3 ШИМ сигнал со скважностью 33%

```
>>> pwm(34, 0.33)
```

Выдадим на 2-й ЦАП напряжение 2.785 В

```
>>> dac(2, 2.785)
```

Задержка 1.23 секунды

```
>>> delay(1.23)
```

Инициализация таймера 2 со входом с канала 3 в режим подсчета импульсов с обнулением при прочтении с предустановкой значения 12345

```
>>> count(223, 12345)
```

Чтение состояния счетчика 2

```
>>> count(2)
12345
```

Инициализация таймера 3 со входами с группы 1 в режим энкодера с обнулением при прочтении с предустановкой значения 0

```
>>> count(341, 0)
```

Чтение состояния счетчика 3

```
>>> count(3)
0
```

Запись в устройство I2C со slave-адресом 0x10 в ячейки начиная с 0-й байтов 1, 2, 3

```
>>> i2cw(0x10, 0, 1, 2, 3)
0
```

Запись в устройство I2C со slave-адресом 0x54 в ячейки начиная с 6-й байта 0x80 и строки "Hello!"

```
>>> i2cw(0x54, 6, 0x80, 'Hello')
0
```

Чтение из устройства I2C со slave-адресом 0x10 из ячейки 0 однобайтового числа

```
>>> i2cr(0x10, 0, 1)
123
```

Чтение из устройства I2C со slave-адресом 0x10 из очередной ячейки(без установки адреса) двухбайтового числа

```
>>> i2cr(0x10, 2)
65535
```

Обмен с устройством SPI , посылка 5 байт 0x80, 0x30, 0x31, 0x32, 0x0A и прием 5 байт из канала(все 0xFF)

```
>>> spi(0x80, '123\n')
'\xff\xff\xff\xff\xff'
```

Циклическое измерение температуры 1раз в секунду на термодатчике DS18B20, подключенном к линии 7, и вывод на ЖКИ в позицию 5 верхней строки в формате 3 знака слева, 4 справа

```
>>> while 1:
...     lcd(5, 0, ds18b20(7), 3.4)
...     delay(1)
```

Список функций библиотечного модуля "fr"

adc(n) Измерение напряжения на линии.
count(n, value) Работа со встроенными аппаратными счетчиками.
dac(n, value) Работа со встроенным ЦАП.
delay(t) Задержка.
ds18b20(n) Работа с термодатчиком DS18B20.
i2cr(sl_adr, wd_adr, n) Чтение из I2C устройства с адресом "sl_adr" (+"wd_adr") n байт.
i2cw(sl_adr, wd_adr, data,) Запись в канал I2C по адресу "sl_adr" (+"wd_adr") data.
lcd(x, y, mode, mes, f) Работа с ЖКИ.
pin(n, mode) Дискретный ввод/вывод, управление состоянием линии.
pwm(n, value) Работа со встроенными аппаратными ШИМами.
spi(data,) Работа с последовательным каналом SPI.

pin(n, mode) ***Список функций***

Дискретный ввод/вывод, управление состоянием линии.
Число параметров: 1, 2**.**
n : номер линии. Тип-> *int*. Допустимые значения: 0 ... 34, 0xA0...0xD2.
mode : режим линии. Тип-> *int*. Допустимые значения: 0...0xFF.
Возвращает : состояние линии. Тип-> *int*. Значения : 0, 1.

Функция обеспечивает считывание и управление состоянием дискретной линии ввода-вывода.
Обязательный параметр ***n*** - номер линии конкретного модуля 0...34(короткая - косвенная адресация),
или номер порта-линии микроконтроллера 0xA0...0xD2(абсолютная адресация).
Например, линия порта A №4 может быть адресована как 0A4h, D №15 -> 0DFh , а линия модуля MCU32-1.x IO3 (контакт №3 X2) как 3, линия KR19(контакт №19 X1) как 1Fh.
Если параметр ***mode*** не задан, то линия переходит в состояние входа, при этом, если прошлое состояние было выходом, то новое состояние будет «Режим дискретного входа с подтяжкой +3.3В / ~ 40 К» (mode=0x81).
Если прошлое состояние было входом, то режим будет сохранен.(аналоговый/дискретный, с подтяжкой/без и т.д.)
Варианты значений ***mode***:

Номер режима	**Описание**
0	**Дискретный выход 0В**
1	**Дискретный выход +3.3В**
0x10	Дискретный выход 0В с ограничительным резистором 1К
0x11	Дискретный выход +3.3В с ограничительным резистором 1К
0x40	**Дискретный выход альтернативной функции push-pull**
0x41	**Дискретный выход альтернативной функции open-drain**
0x80	**Режим дискретного входа с подтяжкой 0В / ~ 40 К**
0x81	**Режим дискретного входа с подтяжкой +3.3В / ~ 40 К**
0x82	**Режим плавающего дискретного входа 0 / +3.3В**
0x90	Режим дискретного входа с подтяжкой 0В / 1 К (делитель 0 / +24В)
0x91	Режим дискретного входа с подтяжкой +3.3В / 1 К («сухой контакт»)
0xC0	**Аналоговый режим потенциального входа 0…+3.3В**
0xD0	Аналоговый режим потенциального входа 0…+10В(+24В)
0xD1	Режим работы с термодатчиками PT1000
0xE0	Аналоговый режим токового входа 0…20мА

Применение других значений ***mode*** , не перечисленных в этом перечне, приведет к настройке линии в режим 0x82.
Доступность того или иного режима см. в описании на соответствующий модуль. В некоторых случаях для реализации конкретного режима требуется установка соответствующих джамперов. Жирным шрифтом выделены режимы доступные во всех модификациях.

Примеры:

```
pin(31, 1) # выдача 1 на линию 31(для модуля MCU32-1.x это KR19)
pin(0xA7, 0) # выдача 0 на линию 7 порта A
pin(3, 0x40) # настройка линии 3 "Дискретный выход альтернативной функции push-pull"
x = pin(5, 0x80) #ввод линии 5 и настройка в "Реж.дискр.входа с подт.  0В / ~ 40К"
x = pin(0xB7) # ввод линии 7 порта B с сохранением режима входа установленного ранее
```

adc(n) ***Список функций***

Измерение напряжения на линии.
Число параметров: 1.
n *:* номер линии / задание режима. Тип-> *int*. Допустимые значения: 0 ... 15 / 100...102, 200...201.
Возвращает : измеренное напряжение. Тип-> *float*. Значения : 0...3.3 .

Функция предназначена для получения измерения выбранного канала АЦП.
Параметром функции является номер аналогового канала (0…15, 200...201) или номер устанавливаемого режима.
Линии имеющие функции аналогового входа перечислены в описании соответствующего модуля.
При первом вызове функции производится инициализация узла АЦП, при этом возвращается неправильное значение в независимости от канала измерения.
При обращении к линии, которая до этого была настроена на дискретный режим, она автоматически настраивается на режим аналогового входа.
Для модулей с расширенными возможностями конфигурации входов (MCX53-32.x), возможна предварительная конфигурация входных цепей, которая осуществляется функцией pin(). При этом формат результата будет соответствовать выбранному режиму.
Функция возвращает измерение в одном из трех форматов заданных ранее. Формат один для всех каналов АЦП.
Возможны 3 варианта:
- формат 0 (по умолчанию) -> величина в вольтах 0…3,3 В ;
- формат 1 -> величина соответствующая коду АЦП 0…4095.0 ;
- формат 2 -> величина в долях от целого 0…1.0 .

Для выбора формата необходимо обратиться к несуществующим каналам 100, 101, 102 соответственно.

Примеры: :

```
adc(101)    # установка формата «код АЦП»
x = adc(4)  # измерение канала 4
```

Также, функция позволяет получить сведения о температуре кристалла и напряжении встроенного REF.
Функция adc(200) возвращает температуру в цельсиях.
Функция adc(201) возвращает значение REF в вольтах. В качестве опорного напряжения для встроенного АЦП используется основное напряжение питания микроконтроллера 3.3В, оно задается отдельным внешним для микроконтроллера стабилизатором. Встроенный REF имеет типичное значение = 1.2В. При необходимости, пользователь может использовать его измерение для дополнительной калибровки точности своих измерений.

dac(n, value) ***Список функций***

Работа со встроенным ЦАП.
Число параметров: 1, 2**.**
n *:* номер ЦАП. Тип-> *int*. Допустимые значения: 1, 2, 100...102.
value : записываемое значение. Тип-> *int / float*. Допустимые значения: 0...3.3 / 0...4095 / 0...1.0 .
Ничего не возвращает .

Функция предназначена для выдачи аналогового сигнала на выбранный канал ЦАП.
Параметр ***n*** - номер ЦАП (1, 2) или команда к смене формата интерпретации значения ***value*** .
Линии имеющие функции аналогового выхода перечислены в описании соответствующего модуля.
При вызове функции происходит автоматическая инициализация соответствующей линии в режим аналогового выхода.
Функция заносит значение ***value*** в указанный ЦАП (0…3,3В) в соответствии с выбранным форматом данных.
Формат один для всех каналов ЦАП. Возможны 3 варианта:
- формат 0 (по умолчанию) -> величина в вольтах 0…3,3 В;
- формат 1 -> величина соответствующая коду записываемому в ЦАП 0…4095;
- формат 2 -> величина в долях от целого 0…1.

Для выбора формата данных необходимо обратиться к несуществующим каналам 100, 101, 102 соответственно, без указания параметра ***value***.

Примеры:

```
dac(2, 1.45)  # выдать на DAC2 1.45В
dac(102)  # установка формата «доля от целого»
```

lcd(x, y, mode, mes, f) ***Список функций***

Работа с ЖКИ.

Число параметров: 3...5**.**

x : номер позиции в строке. Тип-> *int / float*. Допустимые значения для вывода по знакоместам: 0...39.

Допустимые значения для графического вывода : 0.0 … 127.0 . Привязка начала знакоместа первого символа.

y : номер строки. Тип-> *int / float*. Допустимые значения для вывода по знакоместам: 0...3 . Строка 0 - верхняя.

Допустимые значения для графического вывода : 0.0 … 31.0 . Строка 0 — нижняя. Привязка низа знакоместа первого символа.

mode : Тип-> *string*. Режим вывода. Допустимые значения:

отсутствие параметра => режим вывода не меняется,
's' => режим вывода с забоем поверх,
'o' => режим вывода ИЛИ (сложением),
'x' => режим вывода исключающий или,
'n' => режим вывода инверсный (перо белое),
'p' => режим вывода прямой-позитивный (перо черное),
'r' => возврат к режиму вывода обычных символов по знакоместам,
'p' => режим вывода прямой-позитивный (перо черное),
'l' => маленькие(нормальные) символы,
'w' => широкие,
'h' => высокие,
'b' => большие(двойной размер).

mes : сообщение. Тип-> *int / float / string*. Допустимая длинна *string*: 0...21 .

f : формат при выводе числа . Тип-> *int / float*. Для целых 1...10. Для *float* целая часть 0...8, дробная 0...6

Ничего не возвращает .

Функция обеспечивает форматированный вывод на ЖКИ. Для случая графического ЖКИ , поддерживается вывод символов разного размера и заданную точку экрана в разных режимах вывода.
Если координаты ***x*** и ***y*** целочисленные, то они интерпретируются как координаты знакоместа с началом координат в левом углу верхней строки.
Если они плавающие, то воспринимаются как графическая координата точки левого нижнего угла первого знакоместа выводимого сообщения. Графическое начало координат - левый нижний угол.
В обоих случаях обе координаты начинаются с 0.
Необязательный параметр ***mode*** задает режим вывода. При его отсутствии, сохраняется предыдущий режим.
Если выводится число, то должен присутствовать параметр ***f*** , задающий формат вывода.
При выводе целого, указывается число выводимых символов, при этом производится выравнивание по правому краю.
При выводе плавающего, целая часть ***f*** указывает на число символов до запятой, дробная - после. При этом если дробная часть ***f*** равна 0, то отобразится только целая часть числа без точки.
Если выводимое число невозможно отобразить в указанном формате, будут, в соответствии с форматом, выведены знаки '#'. Общее число значащих цифр не должно превышать для целого 10 знаков, для плавающего 8, с учетом знака.

Примеры:

```
lcd(0, 1, 'qwerty') # вывод 'qwerty' с 0-й позиции строки №1(вторая строка сверху)
lcd(45., 7., 'h', '123') # вывод '123' двойной высоты в координату 45,7
lcd(7, 0, x, 3.2) #вывод x в формате '###.##' в позицию 7(8-е знакоместо)верхней стоки
```

count(n, value) ***Список функций***

Работа со встроенными аппаратными счетчиками.
Число параметров: 1, 2**.**
n : номер таймера / канала / режима. Тип-> *int*. Допустимые значения: 2, 3, 8, x11-x14, x21-x24, x31-x32, x41-x42.
value : предустанавливаемое значение. Тип-> *int / float*. Допустимые значения: 0...65535 .
Возвращает : значение счетчика. Тип-> *int*. Значения : 0... 65535 .

Функция обеспечивает взаимодействие с внутренними аппаратными счетчиками, обеспечивающими подсчет импульсов от внешних источников и работу с энкодерами. Подсчет происходит аппаратно, независимо от работы интерпретатора.
Одновременно доступно до трех(в зависимости от типа модуля) независимых 16 битных счетчиков. Для запуска счета необходимо вызвать функцию ***count(n, value)*** с параметрами инициализации. В параметрах указывается:
- номер аппаратного таймера (2, 3, 8) -> третья тетрада считая от младшей,
- режим счета(счет с/без обнулением при прочтении, энкодер, энкодер с обнулением) -> вторая тетрада от младшей,
- комбинация входа(ов)(для счетчика до 4 линий, для энкодера до 2 пар линий) -> младшая тетрада,
- начальное состояние счетчика (0…0xFFFF ⇔ 0…65535).

При инициализации выбранная(ые) линии автоматически инициализируются как входы с подтяжкой.
В режиме энкодера подсчет ведется при каждом изменении любого из сигналов, т.е. на период получается 4 .
Т.к. используются аппаратные таймеры, то для подачи входного сигнала доступны не все линии, а только конкретные комбинации для конкретного типа модуля. Доступные комбинации приведены в описаниях на модули.
Функция ***count(n)*** (без параметра ***value)*** возвращает число накопленных импульсов(шагов для энкодера) для указанного счетчика. . Есть 2 группы режимов с обнулением и без обнуления при прочтении.

Примеры:

```
count(0x213, 7) #инициализация TIM 2 в счетчик без обн. на 3 вход с предустановкой 7
count(0x821, 5) #инициализация TIM 8 в счетчик с обн. на 1 вход с предустановкой 5
count(0x831, 4) #инициализация TIM 8 в энкодер без обн. на 1 комб. с предустановкой 4
count(0x342, 0) #инициализация TIM 3 в энкодер с обн. на 2 комб. с предустановкой 0
count(3, 5678) #занесение в TIM 3 значения 5678
x = count(8)  #чтение значения TIM 8
```

pwm(n, value) ***Список функций***

Работа со встроенными аппаратными ШИМами.
Число параметров: 2**.**
n *:* номер таймера / канала . Тип-> *int*. Допустимые значения: 2, 3, 8, 21-28, 31-38, 81-88.
value : период / значение ШИМ. Тип-> *int / float*. Допустимые значения: 0...65535 мкс / 0.0 … 1.0 .
Ничего не возвращает .

Функция позволяет использовать аппаратные ШИМы микроконтроллера.
Одновременно доступно до 12 линий ШИМ и до 9 линий с фазовым регулированием. Точное число доступных линий определяется конкретным типом модуля(см. описания модулей).
Для работы в режиме ШИМ в интерпретаторе доступны до трех таймеров(2, 3, 8) по четыре канала(1-4) в каждом.
Поддержаны 2 режима работы оператора: обычный и с фазовым регулированием. Для обычного режима доступны 4 выхода, для фазового регулирования 3 выхода и одна линия является входом фазового детектора. Входом фазового детектора всегда является первый канал таймера.

Время периода(для фазового режима это глубина регулирования) задается в микросекундах индивидуально для каждого из таймеров. Для выбора режима фазового регулирования эта величина задается со знаком "-".
Диапазон задания периода/глубины регулирования от 1 до 65535мкс.
Величина ШИМ задается индивидуально для каждого выбранного канала в долях от целого: 0 ... 1.0. Минимальный квант выбран равным 1 мкс. Соответственно чем меньше период, тем разрешающая способность ШИМ меньше. При максимальном периоде она равна 1/65536 (16 бит).
Для инициализации необходимо сначала настроить режим таймера выбранного канала. Для этого в параметре ***n*** указывается номер таймера, параметр ***value*** задает период ШИМ в микросекундах.

Примеры:

```
pwm(2, 10000) # инициализация TIM 2 в ШИМ с периодом 10000 мкс = 100 Гц
pwm(8, 500) # инициализация TIM 8 в ШИМ с периодом 500 мкс = 2000 Гц
pwm(3, -7000) # инициализация таймера 3 с глубиной регулирования 7мс
```

При необходимости можно многократно изменять период таймера.
Инициализация не переводит линии микроконтроллера в режим ШИМ. Для начала работы выбранной линии необходимо обратиться к ней, указав значение ШИМ, при этом линия автоматически настраивается на выход (режим push-pull) и начинает выдавать ШИМ сигнал с заданными параметрами. Линия указывается в параметре вместе с номером таймера.
При этом десятки это номер таймера, единицы это номер выбранного канала этого таймера.
Если вместо номеров 1...4 указать соответственно 5...8, то указанная линия(больше номера на 4) то режим выхода этой линии будет open-drain(открытый сток).
Соответствия ножкам разъемов и количество доступных линий см. в описании на конкретный модуль.

Примеры:

```
pwm(21, 0.5) # перевод соотв.линии канала 1 TIM 2 на выдачу ШИМ скважностью 50%
pwm(34, 0.123) # перевод соотв.линии канала 4 TIM 3 на выдачу ШИМ скважностью 12.3%
pwm(82, 0.0001) # перевод соотв.линии канала 2 TIM 8 на выдачу ШИМ скважностью 0.01%
```

Замечания по режиму фазового регулирования:
На линию приходящую на первый канал каждого из таймеров подается сигнал с фазового детектора. В простейшем случае вполне достаточно всего 2 компонентов:

- резистора сопротивлением порядка 220 кОм / не менее 0.5 Вт и допускающим падение на нем не менее 400В;
- транзисторной оптопары с биполярным входом, например PC814.

На вход оптопары через резистор подается сетевое переменное напряжение, а выходной транзистор подключается непосредственно между землей модуля и входом фазового детектора без дополнительных компонентов. При этом не должно быть никаких соединений между высоковольтной частью и модулями, сигнал передается от сети только через оптопару. Подтягивающие резисторы к выходу оптопары не нужны.
Второй, третий и четвертый каналы таймеров могут быть проинициализированы как выходы на фазовое управление. К ним через ограничивающий резистор можно подключать оптотриаки *без "детектора нуля"*. Для оптотриаков с входным управляющим током 10 мА это резистор порядка 200 Ом.
Данные рекомендации даны из расчета, что пользователь имеет необходимые знания нужные для работы с высоким напряжением и мерах безопасности при этом.
!Если у Вас нет нужной квалификации для работы с сетевым напряжением - не используйте наши модули для этого!
Управление нагрузками осуществляется занесением необходимых значений в ШИМ нужного канала.
Пользователь имеет возможность самостоятельно задавать глубину регулирования в зависимости от конкретной реализации своей схемы и типа нагрузки.
Поскольку основой для реализации этой функции служат аппаратные таймеры, микроконтроллер не тратит программного времени на фазовое регулирование, оно работает полностью прозрачно и независимо от программы.

delay(t)

Задержка.
Число параметров: 1.
t *:* величина задержки. Тип-> *int / float*. Допустимые значения: 0 ... 1000000 секунд, дискрет 0.01с.
Ничего не возвращает .

Функция задержки.
Пример:

```
delay(1.23) # задержка на 1.23с
```

spi(data,) ***Список функций***

Работа с последовательным каналом SPI.

Число параметров: 1 … (24)**.** Общее число байт в посылке не должно превышать 24 байта.

data *:* передаваемые данные. Тип-> *int / string*. Допустимые значения *int* : 0 … 0xFF.

Возвращает : принятые данные. Тип-> *string*. Число принятых байт равно числу переданных.

Функция обеспечивает передачу с одновременным приемом данных по каналу SPI.
В качестве параметра(ов) можно указывать последовательность из целых(один байт) или строк, в том числе и бинарных. Можно чередовать разные типы данных. Важно учитывать, что в рамках одного вызова функции , нужно чтобы общее количество получившейся байтовой последовательности, не превышало 24 байта. Если требуется только чтение канала(передаваемые данные не важны), то в параметре ***data*** нужно задать строку с длинной соответствующей количеству необходимых к принятию байт.
Функция возвращает строку длинной равной числу переданных байт.

Примеры:

```
x = spi(1, 2, 3) # передача байтов 0x01, 0x02, 0x03, в x будет строка длинной 3 байта
x = spi(0x80, '123\n') # передача  0x80, 0x31, 0x32, 0x33, 0x0A, прием 5 байт
x = spi(' ' * 20) # передача 20 байт 0x20, прием 20 байт
```

Пример программы для работы с модулями ADC4-1.x

```
#Модули подключаются в без адресном режиме(без CS)
#сброс канала передачи
spi(0, 255, 255, 255, 255)
while 1:
    spi(0x38, 0x40) #запуск преобразования Single Conversion Mode 16bit
    delay(0.1) #пауза
    x = spi(0x48, 0, 0) #команда "чтение результата" и последующее чтение 2х байт
    u = (ord(x[1]) * 256 + ord(x[2])) / 65535 * 2.5 - 1.25 # приведем к вольтам
    print u # выводим
```

i2cw(sl_adr, wd_adr, data,)

Список функций

Запись в канал I2C по адресу "sl_adr" (+"wd_adr") data.
Число параметров: 2 … (25)**.** Общее число байт в посылке, включая *wd_adr* не должно превышать 24 байта.
sl_adr *:* Slave-адрес. Тип-> *int* . Допустимые значения *int* : 0 … 0xFF(нечетные игнорируются).
wd_adr *:* Word-адрес. Тип-> *int*. Допустимые значения *int* : 0 … 0xFF.
data *:* передаваемые данные. Тип-> *int / string*. Допустимые значения *int* : 0 … 0xFF.
Возвращает : диагностику обмена *int* : ACK 0->OK / -1->NOack.

Функция обеспечивает формирование пакета записи в канал I2C.
Обязательный параметр ***sl_adr*** это адрес устройства которому адресуется пакет. Используются только четные адреса, при задании нечетного адреса будет произведено округление вниз до четного.
Обязательный параметр ***wd_adr*** это адрес ресурса внутри устройства. Фактически это первый байт записываемых данных. Если устройство не имеет адресации внутренних ресурсов или имеет адресацию бОльшим числом байт чем один, ***wd_adr*** можно интерпретировать либо как первый передаваемый байт данных, либо часть адреса, дополнив оставшуюся часть адреса последующими байтами данных.
Параметр(ы) ***data*** это собственно передаваемые данные. Можно указывать последовательность из целых(один байт) или строк, в том числе и бинарных. Можно чередовать разные типы данных. Важно учитывать, что в рамках одного вызова функции , нужно чтобы общее количество получившейся байтовой последовательности с учетом ***wd_adr***, не превышало 24 байта.
Подводя итог, синтаксис функции можно описать так:
первый параметр всегда расценивается как ***sl_adr,*** остальные (обязательно должен присутствовать хотя бы один байт), передаются слитным пакетом данных
Функция возвращает диагностику обмена - подтверждение приема от устройства.

Примеры:

```
x = i2cw(16, 0, 3) #запись в устр. 16 по внутр.адр. 0, байта 0x03,в x подтверждение
i2cw(34, '123\n', 0x55) #запись в устр. 34,байтов 0x31, 0x32, 0x33, 0x0A, 0x55
i2cw(56, 78) #запись в устр. 56, байта 78
```

i2cr(sl_adr, wd_adr, n)

Чтение из I2C устройства с адресом "sl_adr" (+"wd_adr") n байт.

Число параметров: 2 … 3 **.**

sl_adr : Slave-адрес. Тип-> *int* . Допустимые значения *int* : 0 … 0xFF(нечетные игнорируются).

wd_adr : Word-адрес. Тип-> *int*. Допустимые значения *int* : 0 … 0xFF.

n : число запрашиваемых байт данныч. Тип-> *int* . Допустимые значения *int* : 1 … 2.

Возвращает : принятые данные / диагностику. Тип-> *int* . если ***n*** =2, то первый пришедший байт рассматривается как младший. Если нет ответа от устройства, будет возвращено значение -1.

Функция обеспечивает формирование пакета чтения в канале I2C.
Обязательный параметр ***sl_adr*** это адрес устройства которому адресуется пакет. Используются только четные адреса, при задании нечетного адреса будет произведено округление вниз до четного.
Необязательный параметр ***wd_adr*** это адрес ресурса внутри устройства. Если ***wd_adr*** присутствует, то будет передан составной пакет «запись - рестарт- чтение». Сначала в устройство будет записан ***wd_adr*** , потом будет передана комбинация «рестарт» и далее последует чтение.
Если ***wd_adr*** отсутствует, то будет сформирован просто пакет чтения, без предварительной записи ***wd_adr.***
Обязательный параметр ***n*** -это число запрашиваемых байт. Если ***n*** =2, то первый пришедший байт рассматривается как младший.
Функция возвращает целое число в соответствии с количеством запрошенных байт в случае если устройство ответило. Если нет ответа от устройства, будет возвращено значение -1.

Примеры:

```
x = i2cr(16, 0, 2) #чтение из устр. 16 по внутр.адр. 0, 2х байт
x = i2cr(34, 1) #чтение из устр. 34 одного байта
```

ds18b20(n) ***Список функций***

Работа с термодатчиком DS18B20.
Число параметров: 1**.**
n *:* номер линии. Тип-> *int*. Допустимые значения: 0 ... 34, 0xA0...0xD2.
Возвращает : температуру в цельсиях и диагностику обмена. Тип-> *float*. Значения : -300, -200, -100,.-55.0...125.0 .

Функция предназначена для работы с термодатчиками DS18B20.
Обязательный параметр ***n*** - номер линии конкретного модуля 0...34(короткая - косвенная адресация),
или номер порта-линии микроконтроллера 0xA0...0xD2(абсолютная адресация).
Например, линия порта A №4 может быть адресована как 0A4h, D №15 -> 0DFh , а линия модуля MCU32-1.x IO3 (контакт №3 X2) как 3, линия KR19(контакт №19 X1) как 1Fh.
При первом обращении к конкретной линии, функция настраивает указанную линию для работы с цифровым термодатчиком DS18B20 и запускает процесс периодического обмена с ним. Первое обращение это инициализация и оно возвращает неправильный результат измерения(измерение еще не проведено, процесс только запущен).
При каждом обмене производится полная диагностика - КЗ, нет ответа(обрыв), не совпадение CRC8.
При возникновении одной из этих ситуаций, диагностика возвращается в самом результате:

-при несовпадении CRC8 возвращается значение "-100.0";
-при отсутствии ответа(обрыве) "-200.0";
-при КЗ на линии "-300.0".

При нормальном ответе термодатчика - возвращается температура в градусах Цельсия с максимальной точностью, которую дает датчик -> 0.0625 градуса. Автоматически, примерно раз в 750мс, производится чтение результата и запуск следующего преобразования. Запрашиваемый результат возвращается немедленно, без ожидания - берется последнее измерение из буфера.
Интерпретатор позволяет параллельно работать с датчиками на 31(48) линии, при этом это почти не отражается на быстродействии программы, обработка всех разрешенных датчиков идет фоном.
При работе с датчиками поддерживается безадресный режим с / без паразитным питанием.
Необходимым условием правильной работы является наличие подтягивающего резистора 1...3 кОм выбранной линии к +5В или +3.3В. На некоторых линиях он может быть уже установлен - см. описание модуля. Допускается запараллеливание подтяжек, но результирующее сопротивление не должно быть меньше 1 кОм.
Для всех сигнальных входов модулей MCX53-32 подтяжки не требуются, они включаются автоматически, при этом допустима только короткая - косвенная адресация этих линий.
Для случая паразитного питания надо не забывать объединять на датчике ножки "питание" и "общий".
При необходимости, можно работать и с другими типами датчиков: DS1820 и DS18S20. В этом случае потребуется результат делить на 8(у них разрешающая способность меньше в 8 раз), все остальное так же.

Примеры:

```
x = ds18b20(24) # присваивание x результата измерения DS18B20 на линии 24
print ds18b20(0xA3) # печать температуры датчика подключенного к линии PA3print
lcd(6, 0, ds18b20(0), 3.2) #вывод на ЖКИ в позицию 6 верхней строки измерения линии 0
```