# PICµBASIC V3.4.x

# PICµBASIC V3.4.x
# Описание языка программирования

### Требования к оборудованию.

**Платформа с ETHERNET**

Минимально: Контроллер семейства PIC18F67j60, одна микросхема внешней памяти 24LC512.
Максимально: Контроллер семейства PIC18F67j60, четыре микросхемы внешней памяти 24LC512, часы реального времени DS1307, микросхемы интерфейсов ST232, ST485, символьный ЖКИ, графический ЖКИ UC1601, обвязка для шины 1-Wire, для чтения EM-marine карт, выходные ключи… И многое другое. Неподключенные микросхемы/модули не влияют на общую работоспособность. Не будут работать только команды с ними связанные.

**Платформа с USB**

Минимально: Контроллер PIC18F2550/4550, одна микросхема внешней памяти 24LC512.
Максимально: Контроллер PIC18F2550/4550, четыре микросхемы внешней памяти 24LC512, часы реального времени DS1307, микросхемы интерфейсов ST232, ST485, символьный ЖКИ, графический ЖКИ UC1601, обвязка для шины 1-Wire, для чтения EM-marine карт, выходные ключи… И многое другое. Неподключенные микросхемы/модули не влияют на общую работоспособность. Не будут работать только команды с ними связанные.

**Платформа с USB MID edition**

Контроллер PIC18F4550, две/четыре микросхемы внешней памяти 24LC512, часы реального времени DS1307, микросхемы интерфейсов ST232, ST485, символьный ЖКИ, обвязка для шины 1-Wire, для чтения EM-marine карт, выходные ключи… И многое другое. Неподключенные микросхемы/модули не влияют на общую работоспособность. Не будут работать только команды с ними связанные.

**Платформа Device**

Минимально: Контроллер семейства PIC1826K22.
Максимально: Контроллер семейства PIC1826K22/18F46K22, микросхемы интерфейсов ST232, ST485, символьный ЖКИ, обвязка для шины 1-Wire, для чтения EM-marine карт, выходные ключи… И многое другое. Неподключенные микросхемы/модули не влияют на общую работоспособность. Не будут работать только команды с ними связанные.

### Распределение памяти и типы переменных

**Память переменных:**

Всего 1024 8-битных ячеек с адресами от 0 до 1023. На нее отображаются все переменные. Разбита на 4 страницы по 256 байт. При работе с индексными переменными, как массивом или строками запись не должна пересекать границы блоков по 256 байт.

**Память программ**

**WEB**

Микросхемы 24LC512. Вмещает в себя до 1008 строк программы, нумерованных от 0 до 1007
Возможна установка 2-х штук, в этом случае память программ увеличивается до 2032 строк.

**USB**

Микросхема 24LC512. Вмещает в себя до 1023 строк программы, нумерованных от 0 до 1023
Возможна установка 2-х штук, в этом случае память программ увеличивается до 2047 строк.

**DEVICE**

Используется внутренняя память. Вмещает в себя до 511 строк программы, нумерованных от 0 до 511

**Файловая система**

Требуется установка 3 и 4 микросхем 24LC512.Общий размер диска 128 кб. Максимальное количество файлов – 64. Максимальный размер файла 32кб. Поддерживаются все основные функции стандартной файловой системы. Также существует встроенная в прошивку файловая система для нужд WEB сервера.

**Энергонезависимая память данных**

Общий размер - 256(usb)/960(web)/1024(device) байт. Предназначена для хранения пользовательских данных, настроек и др. Поддерживаются функции побайтного доступа и блочного доступа. То есть можно сохранять/читать как одну переменную, так и массив.

**Переменные:**

Для всех переменных выделена область оперативной размером 1024 байта памяти, разделенная на 4 блока по 256 байт. Существует 52 обычных переменных, обозначаемых буквами от **A** до **Z** и **a-z.** Внутреннее представление 16-битное со знаком. Отрицательные числа записываются в дополнительном коде. Отображаются на память с адреса 128 по 231. Занимают два байта, старший байт числа в младшем адресе. При печати оператором **PRINT** отображаются знаковыми целыми в диапазоне от -32768 до +32767

Пример записи выражения с переменными: **10 C=A+b:#(100)=55:$(A)=C +(10/2+1)*2**

Существует три индексных переменных $() , #() , @(). Индексом может быть число от 0 до 1023 или простая переменная, при этом ее значение должно быть в этом же диапазоне. Индекс соответствует адресу ячейки.
Так например индекс =129 соответствует младшему байту обычной переменной **A.** Пустой индекс не допускается.
Внутреннее представление 8-битное . Значение всех переменных с одним индексом –одинаковое. Основное применение – косвенная адресация, работа с текстами и массивами.

Их отличие - при выводе.

Отображение при печати оператором **PRINT** значения ячейки 0x4A по адресу 20
$(20) - представление как символ с кодом 0x4A – **J** по таблице ASCII

#(20) - представление десятичного числа - **74**
@(20) - представление как HEX запись – **4A**

При работе с арифметическими операторами значение представляется числом от 0 до 255. При математических операциях с разными типами в индексные переменные записывается и читается младший байт от обычных переменных. Вложенные индексы не поддерживаются.

Отдельной записи переменных типа «текст» не предусмотрено, тем не менее, поддерживается работа со строками текста. Для этого предусмотрен формат хранения строк в индексных переменных. Любая строка текста может быть записана в массив индексной переменной $(x). В первой ячейке хранится длинна текста байт от 0x00 до 0x7F, в остальных сам текст в ASCII кодах символов. Следует следить за размером текстовых массивов и их возможным наложением друг на друга.

**Числа:**
Все числа являются знаковыми целыми в диапазоне -32767 до +32767 либо целыми в диапазоне 0-255

**Текст**
Текст всегда заключается в кавычки **"text"**

## ***Новое в версии 3.4x*** **- Стек переменных, Относительный переход**

### Стек переменных

Стек переменных организован в области индексных переменных, начиная с адреса 256 до 511, и имеет размер 256 байт (128 знаковых или незнаковых переменных). Стек движется вверх при увеличении указателя. Не зависимо от исходного типа, всегда занимает два байта, старший байт числа в младшем адресе. В случае байтовой исходной переменной, она помещается в младший байт стека.

Основное применение – локальные переменные в программах/подпрограммах и передача параметров к ним/от них.

**Операции со стеком.**

Символ подстановки – знак параграфа **§** , набирается на клавиатуре как alt+21 (удерживая ALT на цифровой клавиатуре набрать 2 и 1 ) или alt+0167

Символ подстановки используется вместо имени переменной.
Он может быть использован как простая переменная §=55 ; A=§ ; PRINT §
Как переменная в индексе #(§)=55 ; PRINT #(§); a=#(§)

Если переменной с именем **§** присваивается какое либо значение, то оно будет помещено в стек и указатель стека увеличится. Первая запись займет адреса (256:257), вторая (258:259) и так далее до исчерпания стека. Переполнение вызовет сдвиг стека с потерей данных в верхушке (256:257) стека(первая запись). Их заменят данные из 258:259(вторая запись). А последняя запись займет последнее место (510:511). Таким образом, в стеке можно организовать циклический буфер.

Если переменная **§** читается, то значение будет выбрано из стека, и указатель стека уменьшится.

Если стек пуст, то всегда будет читаться значение из первой записи (256:257)

**Пример**

```
§=5+3          256:257= 0x00:0X08
§=55           258:259= 0x00:0X37     256:257= 0x00:0X08
A=§
B=§
C=§
После выполнения
A=55
B=8
C=8
```

Указатель стека не изменяется, в случае использования операторов работающих с массивами данных или текстовыми строками, например: WRPE, WRPE, TIME, DATE, SDATE, STIME, SDT, SCOMP, INSTR, MID, VPRINT, SPUT. И при этом символ подстановки используется в качестве индекса для индексных переменных. Так же он не изменяется в операторе NEXT при ветвлении на начало цикла и понижается при выходе. FOR §=0 TO x …. NEXT § работает корректно со стеком, занимая всего одну позицию.

**Операции с указателем стека – операторы.**

**§+** увеличение указателя стека – аналогично записи в стек числа, без изменения самих значений в памяти
**§-** уменьшение указателя стека – аналогично чтения из стека числа, без изменения самих значений в памяти
**§^** Обнуление указателя стека
**§?** загрузка в стек указателя стека, позволяет узнать адрес в стеке текущей записи

Также стек обнуляется при выполнении оператора **RUN, CLR**

**Виртуальные переменные, выделение памяти для задач.**

Часто необходимо предать подпрограмме параметры – это решается заполнением стека нужными параметрами и вызовом подпрограммы.

Но подпрограмме нужен произвольный доступ к данным в стеке без использования переменных, для этого можно использовать виртуальные переменные. Они именованы русскими буквами от **А** до **Я** и от **а** до **я**. Всего их 64. Нумерация согласно алфавита. Буква Ё отсутствует. Эти виртуальные переменные отображаются **А**- последний элемент стека, **Б** – предпоследний, и так до конца. Если стек заполнен менее чем 64 значения, все остальные ссылаются на первый элемент стека.

Также если подпрограмме надо работать с несколькими локальными переменными то выделяем в стеке нужное количество памяти оператором **§+** и работаем с виртуальными переменными, при выходе понижаем стек **§-.** Это позволяет разделить глобальные переменные и локальные в подпрограммах. Во всем остальном виртуальные переменные равнозначны обычным 16-битным переменным.

### Относительный переход

Для операторов
**GOTO, GOSUB, CASE**
можно указывать номер строки, куда выполняется переход – прямая адресация
также теперь можно указывать смещение со знаком – относительный переход от номера текущей строки.

**Пример**

**33 GOTO 100** выполняет переход на строку с номером 100
**33 GOTO +10** выполняет переход на строку с номером 43 (33+10)
**33 GOTO -5** выполняет переход на строку с номером 28 (33-5)

# Операторы

**Арифметические:**
+ сложение
- вычитание
* умножение
/ целочисленное деление (нужно иметь в виду, что, например, 14/5=2)
% - взятие остатка от деления (14%5=4)

& побитное И
| побитное или
^ побитное исключающее или
! побитная инверсия

**Сравнения** (используются только оператором IF):
= равно
<> (или ><) не равно
> больше
< меньше
>= (или =>) больше или равно
<= (или =<) меньше или равно

**Выражения**
Выражения составляются из чисел, переменных, арифметических логических и битовых операторов. Операторы сравнения не могут быть использованы в выражениях. Функции не могут быть использованы в выражениях. Вычисления производятся слева направо. Применением скобок можно изменить порядок вычислений.
Существует 4 уровня.
- сначала вычисляются выражение с унарными операторами **+** и **–**
например: A=-B*2, при этом первым будет выполнено взятие B с противоположным знаком;
- затем вычисляются выражения с ***** и **/** и **%**
- затем вычисляются выражения с **+** и **-**
- затем вычисляются выражения с **& | ^ !**

**Функции**
Функции могут быть одного или нескольких аргументов, перечисленных через запятую в фигурных скобках
Пример записи **A=ROR{#(100),5}:C=SQR{A*A+B*B}** В качестве аргумента могут выступать числа, переменные, выражения.

**Операторы и команды**
У операторов/команд могут быть параметры. Они указываются через пробел от имени оператора и если их несколько, то перечисляются через запятую. Для некоторых через точку с запятой. В качестве параметров могут выступать числа, переменные, для некоторых – текст или выражения.

**Листинг программы**
Все строки программы нумеруются от 0 до 1023 (до 2047)
В одной строке могут быть записаны несколько выражений и/или функций и/или операторов через **: двоеточие.**
Принята запись функций и операторов только ПРОПИСНЫМИ буквами.
Длинна строки ввода ограничена 63 символами, включая номер. Рекомендуется использовать наибольшее заполнение для экономии памяти, если надо. Лишние символы в конце строки игнорируются.
Программа выполняется до последней строки и останавливается. Остановить или закончить выполнение раньше можно командами STOP, END . Для постоянной работы используйте бесконечный цикл.
В данной реализации есть оператор обработки ошибок, который позволяет выполнить переход на другую часть программы при возникновении ошибки в основной части без остановки выполнения программ.

**Встроенный редактор**
Добавление строки - набрать строку с номером
Удаление строки – набрать номер строки
Копирование строки - команда **COPY .** Для копирования массива строк команда COPY может быть использована и в программе. Для удаления строки командой **COPY** надо скопировать несуществующую строку в удаляемую строку или из консоли просто набрать ее номер без текста. Таким образом можно менять структуру программы из самой программы

**Метки**
Меткой перехода является номер строки. Количество ограничено размером самой программы. От 0 до 2047
Если строки не существует, то выполняется ближайшая с большим номером. Пример допустимых меток: 10, 200, 210.

**Потоки ввода-вывода**

**Поток КОНСОЛЬ** – по умолчанию. Текстовый ввод/вывод.

**Поток ТЕРМИНАЛ** - ЖКИ дисплей, подключенный к контроллеру и клавиатура

**Поток RS232** - USART контроллера с возможностью работы через 485 /422 интерфейс. Бинарный и текстовый ввод/вывод

**Поток Ethernet** UDP или TCP/IP клиент/сервер

**Быстродействие**
Среднее время выполнения простых команд – 150 мкс @ 40MHz

**Прошивка контроллера.**
Контроллер программируется программой - загрузчиком один раз с помощью программатора.
В дальнейшем загрузка и обновление **PICµBasic** производится с помощью программы BOOTLOADER
Обновление не затрагивает программу пользователя и регистрацию.

По желанию заказчика в язык могут быть добавлены команды аппаратной обработки любых датчиков, интерфейсов, протоколов.

# Среда PICµBASIC V3.4.x
# Список команд, операторов, функций.

## Функции, операторы, команды - сводная таблица V3.4.x

| **Стандартные команды управления** | | | |
|---|---|---|---|
| | Действие | Пример использования | Описание |
| CLR | Очищает все переменные | **CLR** | Всем переменным присваивается значение 0, очищаются все стеки. Нельзя применять внутри циклов и подпрограмм. |
| RUN | Запуск программы | **RUN**<br>**RUN 10** | Очищает все переменные, запускает программу с нулевого или указанного адреса. Доступна только с консоли. Очищаются все стеки. |
| STOP | Останов программы | **25 STOP** | Останавливает программу в текущей строке. Должен быть единственным или последним в строке. ( кроме оператора REM, это касаемо и остальных с этими ограничениями). |
| BREAK | Прерывает выполнение программы | **BREAK** | Прерывает выполнение программы в текущем месте. Доступна только с консоли и во время выполнения программы. |
| CONTINUE | Продолжение выполнения | **CONTINUE** | Продолжает выполнение с текущего места. Применяется с консоли после остановки оператором STOP, BREAK |
| END | Завершает выполнение программы | **56 IF Z=0 THEN END :REM Окончание программы** | Завершает выполнение программы в текущей строке. Должен быть единственным или последним в строке |
| **Операторы передачи управления.** | | | |
| GOTO | Оператор перехода | **33 GOTO 100** (-> 100)<br>**33 GOTO +10** (->43)<br>**33 GOTO -5** (->28) | Производится переход на строчку с номером 100. Номер строки может быть представлен переменной. Должен быть единственным или последним в строке<br>Для относительного перехода указать смещение со знаком |
| GOSUB | Оператор вывода подпрограммы | **55 GOSUB 200**<br>**55 GOSUB +10**<br>**55 GOSUB -5** | Производит вызов подпрограммы со строки номером 200 Допускается до 8 вложений. Должен быть единственным или последним в строке.(кроме комментариев)<br>Для относительного перехода указать смещение со знаком |
| RETURN | Возврат из подпрограммы | **100 PRINT X:RETURN** | Возврат из подпрограммы. Должен быть единственным или последним в строке. |
| NOERR | Обработчик ошибок | **NOERR 200** | В случае ошибки при выполнении программы, управление будет передано строке с номером 200. Как правило, далее потребуется очистка стеков / переменных командой CLR , если ошибка произошла внутри циклов или подпрограмм. При указании номера равным нулю, функция будет отключена. |
| **Операторы управления стеком** (NEW v3.4.0) | | | |
| §+ | Инкремент указателя стека | **10 §+:§+** | Резервирует две ячейки в стеке. Значение данных в памяти не изменяется |

| | | | |
|---|---|---|---|
| §- | Декремент указателя стека | **25 §-**<br><br>**30 §-:a=§:§+:§+** | Смещает указатель на предыдущую запись. Значение данных в памяти не изменяется<br>Читает в переменную а предыдущую запись , восстанавливает указатель стека. |
| §^ | Очистка указателя стека | **§^** | Обнуляет указатель. Стек пуст. Значение данных в памяти не изменяется |
| §? | Получить указатель | **§?** | Помещает в стек значение указателя<br>Позволяет узнать переполнение / опустошение стека, общее количество записей в нем |
| **Операторы цикла и сравнения** | | | |
| IF THEN | Условный оператор | **10 IF X<10 THEN X=X+1**<br><br>**20 IF X=5+a THEN GOTO 100**<br><br>**30 IF Z=65 THEN GOSUB 200** | Если условие истина, то вычисляется выражение<br>Если условие истина, то производится переход на строчку с меткой 100<br>Если условие истина, то производится вызов подпрограммы меткой 200<br>В условии могут быть числа, переменные, выражения, функции. |
| FOR TO NEXT | Оператор цикла | **5 FOR I=32 TO 127**<br>**6 PRINT $(I);I;"=";#(I)**<br>**7 NEXT I**<br><br>**5 Z=128**<br>**10 FOR I=Z-127 TO Z**<br>**20 #(I)=I+1**<br>**25 PRINT I**<br>**30 NEXT I** | Производит действие 127-32+1 раз с наращиванием переменной цикла. В качестве действия, вывод переменной как символа с кодом I и вывод значения переменной I как числа . Допускается до 8 вложений.<br>После оператора TO может быть переменная. Переменной цикла может присваиваться число, переменная, результат выражения.<br>Не допускается GOTO за пределы цикла.<br>В строке после FOR..TO и NEXT только комментарии. |
| EXFOR | Принудительный выход из цикла | **EXFOR 300** | Применяется внутри цикла. Управление будет предано на номер строки, которая должна быть за пределами текущего цикла. В случае вложенных циклов, для каждого цикла – свой выход.<br>Должен последним или единственным в строке. |
| CASE | Условный переход | **CASE A,25,300,26,310**<br><br>**CASE A,25,+10,26,+11** | Параметры – имя переменной для сравнения; первое значение с которым сравнивается; номер строки перехода, если равно; второе значение для сравнения; номер строки перехода если равно. Количество ограничено длинной строки. Все значения могут быть числами или переменными.<br>Если A=25 то переход на строку 300<br>Если A=26 то переход на строку 310<br>Если не совпало ни с одним значением переход на следующую строку.<br>Для относительного перехода указать смещение со знаком |
| **Таймеры и задержки** | | | |
| PAUSE | Пауза в миллисекундах | **25 PAUSE 10**<br>**26 PAUSE Z** | Приостанавливает выполнение программы на указанное число миллисекунд. Максимально до 32767<br>Не рекомендуется активировать данный оператор со значением более 50 при активном WEB сервере<br>Используйте тогда следующую конструкцию<br>FOR §=0 TO 100<br>PAUSE 10<br>NEXT § - как пример задержки на 1 секунду<br>Или используйте таймер при задержках более 2-х секунд с проверкой окончания : (пример на 20 секнд)<br>10 TIMER 1,20<br>11 IF TIMER{1}=0 THEN GOTO 13<br>12 GOTO 11<br>13 …… |
| TIMER | Функция | **A=TIMER{x}** | Переменной присваивается значение системного таймера с номером X , где X от нуля до семи. Всего 8 независимых таймеров обратного отсчета. |
| TIMER | Оператор | **TIMER x,300** | Установка в системный таймер X значения 300. Каждую секунду от этого значения будет отниматься единица, до достижения нулевого значения. Таймер работает в фоновом режиме, независимо от программы пользователя. Максимальное значение = 32767 |
| **Блоки данных** | | | |

| | | | |
|---|---|---|---|
| DATA | Определения блока данных | **DATA #(i),255,26,27,5,6**<br>**DATA #(i),"Alarm"** | Загружает блок данных. Максимальное количество данных в строке определяется длиной строки, которая не должна превышать 64 символа включая номер строки. Максимальное количество строк с данными ограничено только размером свободной памяти программ. Переменной с индексом **i** присваивается первое число, с индексом **i+1** присваивается следующее и т.д. В случае текста присваиваются коды символов, в каждую по одному символу последовательно |
| | | **DATA $(i),255,26,27,5,6**<br>**DATA $(i),"Alarm"** | Ввод данных, как текст. Аналогично предыдущему, только данные будут располагаться начиная с **i+1.** В **i+0** будет помещена длинна строки. Приемником массива данных могут быть только индексные переменные и массив не должен пересекать границы блоков по 256 байт. |
| **Математические и логические функции** | | | |
| SQR | Вычисление квадратного корня | **10 A=SQR{x}**<br>**20 A=SQR{145}:PRINT A;** | Переменной А присваивается значение квадратного корня переменной x. Аргументом функции выступает переменная, число, выражение. |
| ABC | Возвращает абсолютное значение числа | **20 T=ABC{ #(25)}**<br>**25 S=ABC{X}** | Возвращает абсолютное значение числа. Аргументом функции выступает переменная, число, выражение. |
| ROL | Сдвиг влево | **10 A=ROL{A,5}**<br>**20 A=ROL{B}**<br>**A=ROL{25+4,2}** | Сдвигает число в переменной влево на указанное число бит, при этом в младший разряд втягиваются нули. По умолчанию сдвигает на 1 бит. Аргументом функции выступает переменная, число, выражение. |
| ROR | Сдвиг вправо | **10 A=ROR {A,5}**<br>**20 A=ROR {B}** | Сдвигает число в переменной вправо на указанное число бит, при этом в старший разряд втягиваются нули. По умолчанию сдвигает на 1 бит Аргументом функции выступает переменная, число, выражение. |
| RND | Случайное число | **A=RND{}** | Генерация псевдослучайных чисел в диапазоне от -32768 до 32767 (0-255 для индексных) . Аргументов нет |
| **Операторы ввода вывода** | | | |
| PRINT | Оператор вывода. Вывод на консоль сообщений | **PRINT**<br>**PRINT ;;**<br>**PRINT "Это сообщение"**<br>**PRINT X**<br>**PRINT $(45);**<br>**PRINT "X= ",X,"Y= ",B;**<br>**PRINT X,Y,Z;**<br>**PRINT .2,A;**<br>**PRINT *$(0)** | Вывод списка в консоль. Элементы списка разделяются запятой или точкой с запятой. Если точка с запятой, то следующий элемент выводится с табуляцией на 8. Если запятая - то слитно. Табуляция выполняется пробелами.<br>В случае завершения списка точкой с запятой – строка не переводится. Количество элементов списка от нуля. Сам элемент списка может отсутствовать(табуляция будет выполняться).<br>В качестве элементов списка выступают:<br>Обычные переменные, выводятся в виде чисел со знаком без незначащих нулей.<br>Индексные переменные<br>**#(x)** выводятся как число 0-255 без незначащих нулей.<br>**@(x)** выводятся как HEX запись числа 00-FF.<br>**$(x)** выводятся как символ с кодом в переменной<br>***$(x)** выводится как строка длинной в первом байте переменной<br>- Текст **"asdf"** заключенный в кавычки<br>- Оператор форматирования вывода обычных переменных вида **.X** , где X =0-3, – количество знаков после запятой. 0 – отключает десятичную точку.<br>- Оператор форматирования длинны числа **%x**, где x указывает, сколько последних цифр выводится от написания числа. Значение **x=0** – отменяет действие. Диапазон X= 0-3 для индексных и 0-5 для обычных.<br>- Оператор форматирования **~y**, где y указывает, сколько пробелов печатать перед выводом числа, и также наличие его указывает замену незначащих нулей на пробелы. **y=0** отменяет действие. Y= 0-9<br>Форматирование действует от оператора и до конца списка, либо до появления другого этого же типа. Пробелы, указанные внутри кавычек, выводятся. Длинна буфера вывода - 63 байта для одного элемента списка. |

| | | | |
|---|---|---|---|
| INPUT#<br>INPUT$<br>INPUT@ | Оператор ввода данных с консоли с ожиданием ввода | **INPUT# X**<br><br>**INPUT$ $(i)**<br><br>**INPUT@ #(i)** | Ожидает ввода числа в консоли в формате от-32768 до 32767 или 0-255 для 8-битных и выражения и присваивает переменной. Если входящий поток нельзя интерпретировать как число или выражение , то переменной присваивается нулевое значение. Списки не поддерживаются.<br>Ожидает ввода текста в консоли до 63 символов. В ячейку $(i) будет помещено количество введенных символов а в ячейки $(i+1) и далее будет помещен сам текст, до символа 0x0D или до 63 символов.<br>Ожидает ввода 1 байта в консоли в HEX записи. Если входящий поток нельзя интерпретировать как число, то индексной переменной присваивается нулевое значение. Лишние символы будут отброшены. |
| **Операторы и функции работы с EEPROM** | | | |
| WREE | Запись в EEPROM | **WREE #(1),A**<br><br>**WREE 10,b**<br>**WREE F,A**<br>(Update v3.4.1) | Записать переменную **#(1)** или число по адресу **A** . адрес должен быть в диапазоне 0-959. Используется одна ячейка EEPROM. Для 16 бит переменных запишется младший байт в указанный адрес, затем старший байт в следующий адрес, будет использовано 2 ячейки EEPROM. При записи числа, указанного явно, используется двухбайтовая запись. Параметрами выступают переменная, число. |
| RDEE | Чтение из EEPROM | **#(1)=RDEE {A}**<br><br>**F=RDEE {A}**<br>(Update v3.4.1) | Прочесть 1 байт из EEPROM по адресу **A** в переменную. Параметрами выступают переменная, число, выражение.<br>Прочесть 2 байта из EEPROM по адресу **A** – младший и по по адресу **A+1** –старший в переменную **F**.<br>Параметрами выступают переменная, число, выражение. |
| WRPE | Запись в память строки или массива | **WRPE A,$(i),C**<br>**WRPE $(i),C**<br><br>**WRPE *A,$(i),C** | Запись строки в память по адресу C. Количество указывается явно=A или содержится в первом байте индексной переменной. Записываются все байты $(i) последовательно. Если длинна указана явно, то первой запишется длинна, затем все данные **$(i)** длинной **A**.<br>В случае указания массива - ***** длинна данных не записывается. Только все данные **$(i)** длинной **A** Максимально пишется до 128 байт |
| RDPE | Чтение из памяти строки или массива | **RDPE A,$(i),C**<br>**RDPE $(i),C**<br><br>**RDPE *A,$(i),C** | Чтение строки по адресу C длинной A или как указано в первом прочтенном байте. Приемником массива данных могут быть только индексные переменные и массив не должен пересекать границы блоков по 256 байт. В **$(i)** будет записано число символов или из указанного явно или из прочтенного первого байта, в остальные будут записаны данные.<br>В случае указания массива - ***** длинна данных не записывается. Только данные длинной **A** в переменную **$(i).** Максимально читается до 128 байт |

## Операторы и функции работы с часами реального времени

Часы реального времени реализованы на микросхеме DS1307. Если не установлены – то следующие операторы и функции работать не будут. На остальное влияния не оказывает.

| **Числовые функции** | | | |
|---|---|---|---|
| YEAR | Функция | **A=YEAR{}** | Переменной присевается текущий год , две последние цифры (0-99) |
| MONTH | Функция | **B=MONTH{}** | Переменной присваивается текущий месяц (1-12) |
| DAY | Функция | **C=DAY{}** | Переменной присваивается текущее число (1-31) |
| DAYS | Функция | **C=DAYS{}** | Переменной присваивается текущий день недели (1-7) |
| HOUR | Функция | **#(i)=HOUR{}** | Переменной присваивается текущее значение часа (0-23) |
| MIN | Функция | **#(i)=MIN{}** | Переменной присваивается текущее значение минут (0-59) |
| SEC | Функция | **#(i)=SEC{}** | Переменной присваивается текущее значение секунд (0-59) |
| DATE | Функция | **(i)=DATE{}** | Переменной с индексом **i** присваивается текущее число, с индексом **i+1** присваивается текущий месяц, с индексом **i+2** присваивается текущий год. Приемником /источником массива данных могут быть только индексные переменные и массив не должен пересекать границы блоков по 256 байт. |
| TIME | Функция | **#(i)=TIME{}** | Переменной с индексом **i** присваивается текущий час, с индексом **i+1** присваиваются минуты, с индексом **i+2** присваиваются секунды. |

| | | | |
|---|---|---|---|
| TCMP | Функция | **A=TCMP{#(a), #(b)}** | Функция сравнения текущего времени в заданном интервале. Переменной **A** присваивается 1 в случае текущего времени час:мин:сек в интервале от **#(a)-**час: **#(a+1)-**мин: **#(a+2)-**сек до **#(b)-**час: **#(b+1)-**мин: **#(b+2)** и ноль в противном случае. Если #(b)<#(a) то интервал считается от #(a) до 23:59:59 и от 00:00:00 до #(b) |
| **Текстовые функции** | | | |
| SDATE | Оператор | **SDATE $(i)** | Переменной с индексом (i) присваивается длинна текста, с индексом (i+1) и далее - текст формата «15-01-15» содержащий текущую дату |
| STIME | Оператор | **STIME $(i)** | Переменной с индексом (i) присваивается длинна текста, с индексом (i+1) и далее - текст формата «12:20:07» содержащий текущее время |
| SDT | Оператор | **SDT $(i)** | Переменной с индексом (i) присваивается длинна текста, с индексом (i+1) и далее - текст формата «12:20:07 15-01-15» содержащий текущее время и дату |
| **Операторы установок** | | | |
| YEAR | Оператор | **YEAR 15** | Устанавливает год. Параметрами могут быть числа, переменные. |
| MONTH | Оператор | **MONTH 5** | Устанавливает месяц |
| DAY | Оператор | **DAY #(5)** | Устанавливает день |
| DAYS | Оператор | **DAYS #(5)** | Устанавливает день недели (1-7) |
| HOUR | Оператор | **HOUR i** | Устанавливает час |
| MIN | Оператор | **MIN 10** | Устанавливает минуты |
| SEC | Оператор | **SEC 30** | Устанавливает секунды |

## Группа операторов SET

Все команды этого раздела доступны только с консоли.
Предназначены для настройки контроллера и PICμBASIC. Дополнительные операторы этой группы, относящиеся к периферийным модулям, или модулям расширения смотрите в описаниях команд этих модулей.

| | | | |
|---|---|---|---|
| LGNA | Установка логина администратора | **SET LGNA "ADMIN"** | В кавычках указывается текст. Вход администратора обеспечивает доступ ко всем функциям. Если логин = "" то аутентификация отключена. Максимальная длинна 8 символов. Чувствительно к регистру. |
| LGNU | Установка логина пользователя | **SET LGNU "USER"** | В кавычках указывается текст. Вход пользователя используется для доступа к WEB серверу. Если логин = "" то аутентификация отключена. Максимальная длинна 8 символов. Чувствительно к регистру. |
| PSWA | Установка пароля администратора | **SET PWSA "ADMIN"** | В кавычках указывается текст. Максимальная длинна 8 символов. Чувствительно к регистру. |
| PSWU | Установка пароля пользователя | **SET PSWU "USER"** | В кавычках указывается текст. Максимальная длинна 8 символов. Чувствительно к регистру. |
| IP | Установка IP адреса | **SET IP 192.168.1.2** | Установка IP адреса контроллера |
| MASK | Установка маски сети, последний октет | **SET MASK 128** | Установка маски сети 255.255.255.128, три первых октета равны 255. |
| DG | Установка шлюза по умолчанию, последний октет | **SET DG 1** | Установка шлюза по умолчанию 192.168.1.1, три первых октета берутся из IP адреса. |
| DHCP | Вкл /выкл получения адреса по DHCP | **SET DHCP ON**<br>**SET DHCP OFF** | Включает/выключает DHCP клиент. Если включен, то получает адрес по DHCP. Если сервер DHCP не доступен, то используется свои настройки. |
| NAME | Установка имени контроллера | **SET NAME "HOME_PLC"** | В кавычках указывается текст. Максимальная длинна 8 символов. |
| AUTORUN | Вкл /выкл автозапуска программы | **SET AUTORUN ON**<br>**SET AUTORUN OFF** | Автозапуск программы по включения питания со строки с наименьшим номером. |
| WEBEXT | Вкл /выкл доступа WEB сервера к файлам Flash Disk | **SET WEBEXT ON**<br>**SET WEBEXT OFF** | Доступ WEB сервера к Flash диску для чтения любых файлов. |
| WEBINT | Вкл /выкл доступа WEB сервера к внутренним файлам | **SET WEBINT ON**<br>**SET WEBINT OFF** | Доступ WEB сервера к файлам в прошивке процессора. |

### Служебные и информационные операторы

| | | | |
|---|---|---|---|
| NEW | Очистить программу | **NEW** | Стирает текущую программу, форматирует память программ. |

| | | | |
|---|---|---|---|
| COPY | Копирует строку программы | **COPY 10,20** | Копирует программную строку с номером 10 в строку с номером 20. Предыдущие данные в строке 20 очищаются. Номер строки может быть представлен переменной. |
| REM | Оператор комментирования | **REM Это комментарий**<br>**20 REM GOTO 100** | Комментарий, интерпретатор не исполняет эту строчку<br>Исключение оператора GOTO, строчка превращается в комментарий |
| LIST | Выводит листинг | **LIST**<br>**LIST 10,100**<br>**LIST A,B** | Выводит весь листинг программы<br>Выводит листинг со строки 10 по строку 100 включительно. Номер строки может быть представлен переменной. |
| LERR | Last ERROR | **LERR** | Показывает последнюю ошибку в консоль.<br>Применяется в случае остановки по ошибке запущенной программы. |
| VER | Оператор | **VER** | Показывает версию П/О в консоль |
| BOOT | Оператор | **BOOT** | Отмена инсталляции BASIC и запуск бутлоадера |
| REBOOT | Оператор | **REBOOT** | Перезагрузка контроллера |
| STATUS | Оператор | **STATUS** | Получение данных о состоянии контроллера. |
| CODE | Оператор | **CODE "1234-5678"** | Ввод кода |
| **Операторы и функции работы со строками** | | | |
| SCOMP | Сравнение двух массивов | **A=SCOMP{#(i),#(k),5}** | Возвращает 1 в случае совпадения или 0 при не совпадении. Сравниваются два массива #(i) и #(k) длинной каждый 5 байт. Источником массива данных могут быть только индексные переменные длинной до 128 байт и массив не должен пересекать границы блоков по 256 байт. Возвращает 1 в случае нулевой длинны. |
| SCOMP$ | Сравнение двух массивов текста | **A=SCOMP${$(i),$(k)}** | Возвращает 1 в случае совпадения или 0 при не совпадении. Сравниваются два массива с началом $(i) и $(k) длинной $(i) байт от $(i+1) до $(i+$(i)) и от $(k+1) до $(k+$(i)) соответственно. Возвращает 1 в случае нулевой длинны. Максимальная длина 128 байт. |
| INSTR$ | Поиск вхождения строки | **A=INSTR${$(i),"OK"}** | Осуществляет в переменной $(i), интерпретированной как строка текста, поиск вхождения подстроки заданной явно или строковой переменной. Возвращает 0 если не найдено или номер символа с которого найдено совпадение |
| VAL | Преобразование текста в число | **A=VAL{$(5),2}**<br><br>**#(15)=VAL{@(5),2}** | В переменную **A** помещает число из текста взятого у текстовой переменной $(5) с позиции 2 до символа разделителя(пробел, запятая, конец строки...). Запись согласно правилам текстовых переменных.<br>Длинна текста для преобразования не должна превышать 5 байт.<br>В переменную **#(15)** помещает число из текста взятого у текстовой переменной $(5) с позиции 2 длинной 2 символа, как HEX запись. Допустимые символы 0-9, A-F. |
| MID$ | Оператор подстроки | **MID$ $(5),$(50),5,2** | В текстовую переменную $(5) помещает символы взятые у текстовой переменной $(50) с позиции 5 длинной 2 символа. Запись согласно правилам текстовых переменных. |
| VPRINT | Оператор вывода Приемником является индексная переменная. | **VPRINT $(10) "X= ",X, "Y= ",B** | Виртуальный вывод. Полностью аналогичен оператору PRINT. Все выводимые данные помещаются в указанную индексную переменную, как текст, со второго байта. Первый байт будет содержать длину данных. |

**Операторы и функции работы с портами ввода - вывода**

| | | | |
|---|---|---|---|
| PORT | Функция | **#(i)=PORT{A}** | Порт ввода/вывода настраивается на ввод<br>Переменой присваивается 1 если состояние порта ввода/вывода с номером **A** =лог.1 или 0 если лог.0<br>Номера портов от 1 до 34. По включению порты настроены на ввод. Опрос производится в момент выполнения функции. Если требуется узнать изменения за прошедший период – используйте функцию **GKEY** |
| PORT* | Функция | **#(i)=PORT*{A}** | Опрос порта без изменения направления ввода-вывода.<br>Переменой присваивается 1 если состояние порта ввода/вывода с номером **A** =лог.1 или 0 если лог.0<br>Номера портов от 1 до 34. По включению порты настроены на ввод. Опрос производится в момент выполнения функции. Если требуется узнать изменения за прошедший период – используйте функцию **GKEY** |

| | | | |
|---|---|---|---|
| PORT | Оператор | **PORT A,i** | Порт ввода/вывода с номером **A** настраивается на вывод, и устанавливается в ноль, если **i**=0;<br>в единицу , если **i**=1;меняет свое состояние на противоположное , если **i**=2. |
| PWM1<br>PWM2<br>PWM3<br>PWM4<br>PWM5 | Модуль ШИМ | **PWM1 1,#(I)**<br><br>**PWM3 1,50** | Включение и настройка модуля PWM, установка заполнения.<br>Первый параметр 0 – PWM выключен, настроен на ввод , 1- включен, частота 2543 Гц, 2 – частота 10172 Гц, 3- частота 40690 Гц.<br>Второй параметр – заполнение ШИМ 0-255 .<br>PWM1 – 1 канал, PWM2 – 2 канал, и т.д.<br>Каналы разбиты на 2 группы по частоте. Первая - 1и 2. Вторая 3,4,5. Частота в одной группе одинаковая и выбирается последним оператором. При использовании RFID первая группа отключена.При использовании генератора – отключается вторая группа. |
| ADC | Чтение АЦП | **A=ADC{n}**<br>**#(i)= ADC{2}** | Настраивает выбранный канал АЦП и все каналы с меньшим номером, как аналоговые входы АЦП. Читает значение с выбранного канала АЦП в переменную.<br>Выходное число 0-1023 для 16 бит переменной,<br>0-255 для 8-бит. Номера каналов 1-9<br>Для отключения модуля АЦП и аналоговых входов нужно указать номер канала равный нулю. При этом порты, ранее занятые АЦП будут настроены на ввод. |
| BEEP | Генерация звука | **BEEP C,D** | Генерация звука с номером ноты = C, длительностью =D. Длительность кратно 25 Ms. Номера нот 1-31 от До первой октавы до Ми пятой октавы. |
| GCLK | Счетчик импульсов | **A=GCLK{x}** | Счетчик импульсов за 1 mS, 10 mS, 100 mS, 1000 mS для x= 1,2,3,4 соответственно<br>При переполнении - возвращается отрицательное число - 32768. Не применим для индексных переменных.<br>Максимальная частота измерения - 9,4 МГц |
| CLK1<br>CLK2<br>CLK3 | Генератор частоты (меандр) на выводе CCP3,CPP4,CPP5 соответственно | **CLK1 y,x**<br>**CLK2 y,x**<br>**CLK3 y,x** | Включает генератор частоты От 2.5 кГц до 5,2 МГц .<br>Отключает вторую группу PWM. Генератор работает независимо и постоянно от момента включения.<br>Диапазон частот разбит на 3 диапазона Y=0,1,2,3<br>0 – выключен, порт настраивается на ввод<br>В каждом 128 частот. X = 0-127<br>1 диапазон, частота = 10416666/(2x+2)<br>2 диапазон, частота = 2604167/(2x+2)<br>3 диапазон, частота = 651041,7/(2x+2) |

### Сетевые операторы и функции

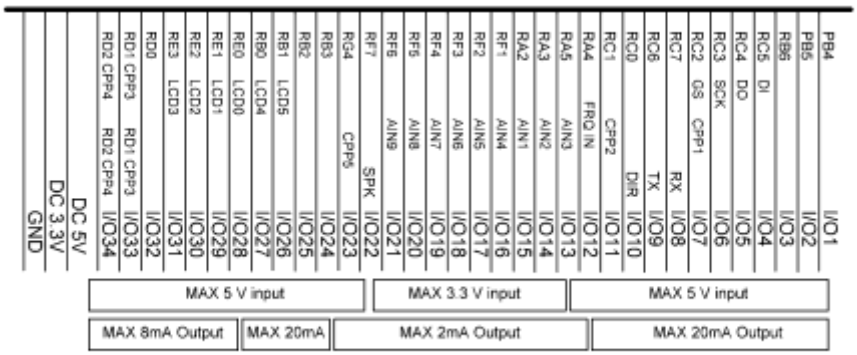
| | | | |
|---|---|---|---|
| PING | Функция | **A=PING{192.168.1.1}**<br>**A=PING{K.L.M.N}** | Переменной **A** присваивается время в миллисекундах ожидания ответа на ping заданного узла. В случае недоступности A=-32768. Может выполняться до 3 секунд. |
| IPNEW | Функция | **A=IPNEW{}** | Попытка получить новый IP адрес по DHCP. IP адрес изменится в случае удачного запроса. Контроллер обычно загружается быстрее DHCP сервера по сбросу питания, и требуется обновление IP через какое то время после старта (для конфигураций с включенным DHCP). Если DHCP отключен - обновление не произойдет.<br>Возвращает 1 в случае обновления по DHCP, 0 если DHCP не доступен |
| IPCFG | Оператор | **IPCFG #(0)** | В переменную **#(0)** помещается структура текущих сетевых настроек.<br>В первые девять байт #(0) - #(8) имя контроллера, как текст (может быть выведен как $(0))<br>#(9)- #(12) текущий IP, 4 байта<br>#(13) – DG последний октет<br>#(14) – MASK последний октет<br>#(15) - reserved (00) |
| UDPSND | Функция | **A=UDPSND{$(0),192.168.1.1:253}**<br>**A=UDPSND{#(0),K.L.M.N:P}** | Посылает UDP пакет на указанный IP : PORT с текстом из переменной $(0), до 63 байт<br>Посылает UDP пакет на указанный IP : PORT длинной 64 байта из #(0)-#(63)<br>Возвращает 1 в случае успеха, 0 если маршрут не доступен<br>Может выполняться до 2 секунд. В случае выполнения из консоли не выполнения - будет расшифровка ошибок |

| | | | |
|---|---|---|---|
| TCPSND | Функция | **A=TCPSND{$(0),192.168.1.1:80}**<br><br><br><br><br>**A=TCPSND{$(0),K.L.M.N:P, $(500)}** | Посылает запрос на указанный IP : PORT с текстом из переменной $(0), до 63 байт вида :<br>GET **/index.htm?lg=web&ps=as&hs=123456** HTTP/1.1 0x0D 0x0A User-Agent: 'NAME_CONTROLLER' 0x0D 0x0A 0x0D 0x0A<br>Где $(0) = **/index.htm?lg=web&ps=as&hs=123456**<br>Как минимум $(0) должна содержать символ «/», если она пустая, то «/» будет подставлен автоматически.<br>Посылает запрос на указанный IP : PORT с данными из переменной #(0)-#(63) – 64 байта.<br>Возвращает 1 в случае успеха, 0 если не отправлено<br>Может выполняться до 3 секунд.<br>В случае выполнения из консоли не выполнения - будет расшифровка ошибок<br>В переменную $(500), если она указанна, помещает данные, как текст, из буфера последнего принятого пакета.<br>Данные будут обрезаны до размера 63 байт. |
| | | | |

Для правильной работы в общей сети должен быть указан адрес шлюза по умолчанию и маска сети.

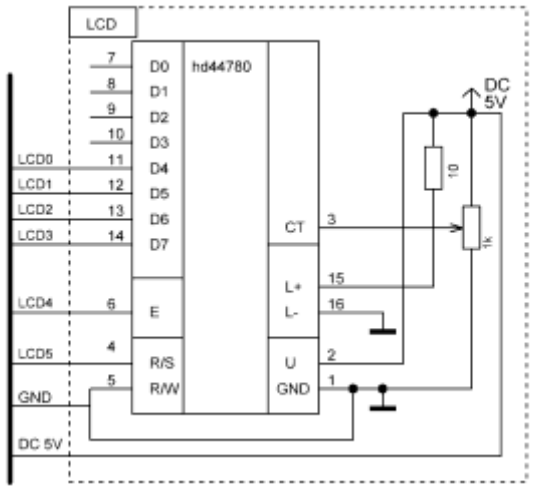
## Операторы и функции работы с периферийными модулями

Все модули расширения подключаются к портам ввода вывода контроллера.
На схеме представлены наименования портов микроконтроллера и соответствие входных или выходных сигналов контроллера. Также указаны максимальные величины подаваемых напряжений на вход и максимальные токи на выход. Данные значения превышать нельзя. Все выходные уровни соответствуют логике 3,3 вольт




### Схема подключения символьного дисплея с контроллером HD44780

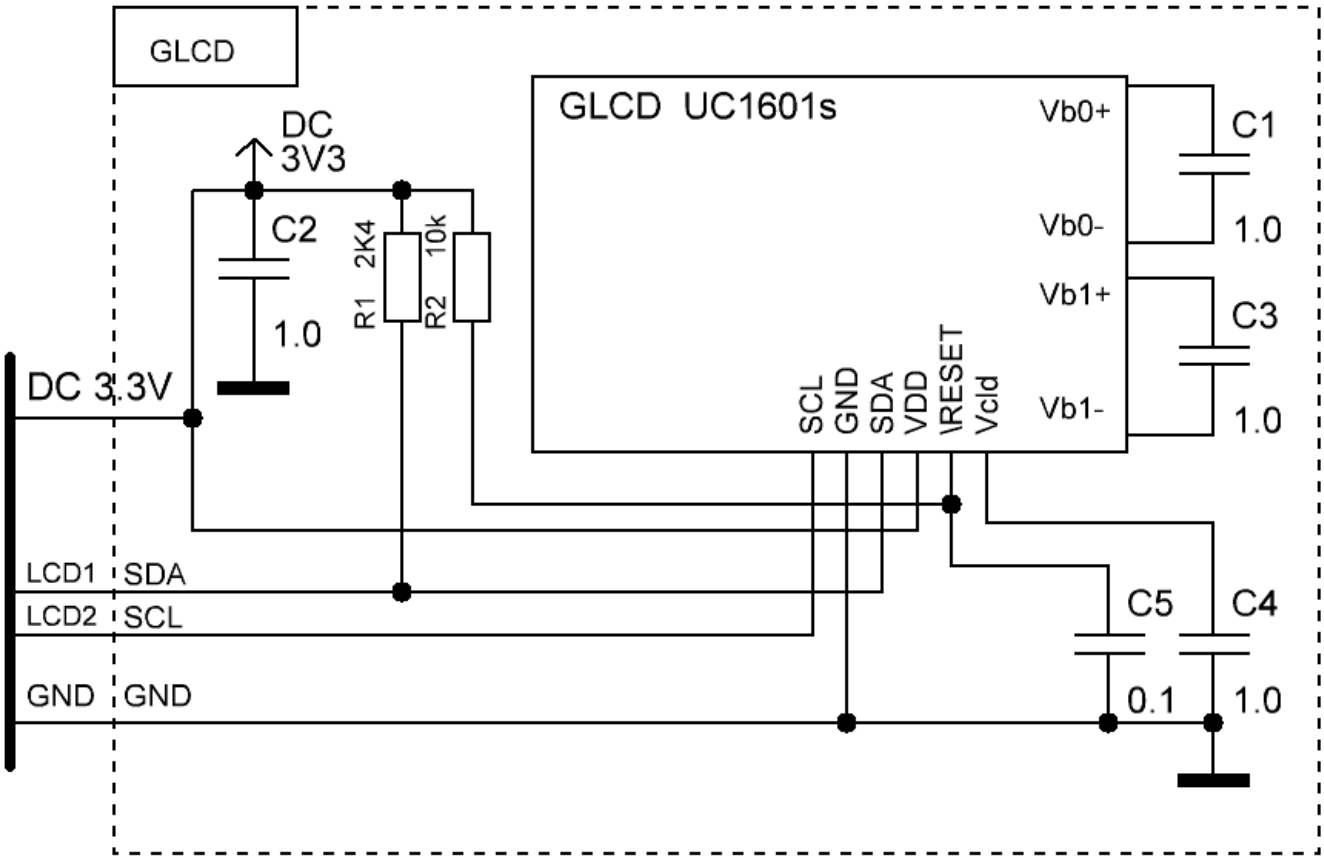
Или любого совместимого от 1*8 до 4*20.




## Операторы и функции работы с дисплеем

| | | | |
|---|---|---|---|
| CLS | Очистка экрана дисплея | **20 CLS** | Очистка экрана дисплея. Курсор устанавливается на 0 позицию. |
| AT | Установка курсора | **AT 10** | Установка курсора на позицию 10. Нумерация с нуля. Адреса знакомест для дисплея 2*16 начинаются с 0 для 1 строки и с 64 для второй. Параметрами выступают переменная, число.<br>Для графического - устанавливает адрес в пикселях и номер строки вывода |
| LCDD | Послать данные в дисплей | **20 LCDD $(5)** | Записывает в дисплей значение переменной в регистр данных без перекодировок. Параметрами выступают переменная, число. Коды русских букв не совпадают в дисплее, подробнее смотрите в документации на дисплей HD44780. Для графического записывает 8 точек по вертикали в текущей строке и текущем адресе. |
| LCDC | Послать команду в дисплей | **20 LCDC #(3)** | Записывает в дисплей значение переменной в регистр команд. Подробнее смотрите в документации на дисплей HD44780. Например включить курсор – 14.<br>Для графического только 0/1 - включение инверсии вывода текста. |
| LPRINT | Оператор вывода. Вывод на ЖКИ сообщений | **LPRINT "Это сообщение"**<br>**LPRINT X**<br>**LPRINT $(45)**<br>**LPRINT "X= ",X,"Y= ",Y**<br>**LPRINT X,Y,Z**<br>**LPRINT #2,A** | Полностью аналогичен оператору PRINT<br>Перевод строки - не поддерживается. Используйте оператор AT. |
| LINIT | Включение и инициализация дисплея | **LINIT X,y,z** | X=0 - OFF ; X=1 - HD44780 ; X=2 - UC6101s, значение контрастности , режима ориентации. При нулевых значениях - устанавливаются по умолчанию. |

### Схема подключения графического дисплея с контроллером UC1601s




На этом контроллере выпускается широкий спектр графических дисплеев
RDT06569-IE - 128x32 точки, видимая область 65.5x30.16
RDX0032-GC - 128x32 точки, видимая область 76.0x23.0
RDX0048-GC - 128x32 точки, видимая область 26.0x8.0
RDX0077-GS - 128x64 точки, видимая область 67.5x34.6

RDX0120-GC - 64x32 точки, видимая область 39.86x23.0
RDX0154-GC - 132x64 точки, видимая область 62.0x32.0

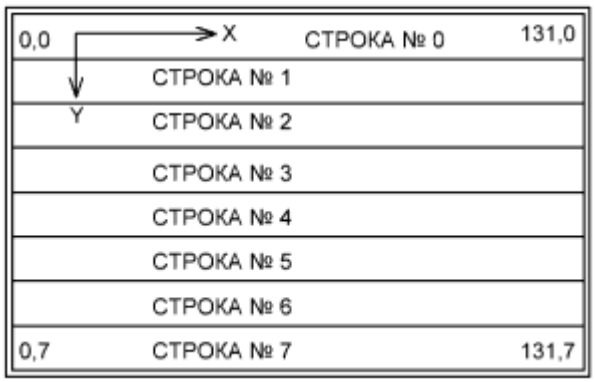
В данном контроллере дисплей может использоваться в текстовом и графическом виде.

Для текстового вывода используется знакогенератор 6*8 с латинскими и русскими буквами с кодами символов от 0x20 до 0xFF

Также имеются символы псевдографики и различные пиктограммы.

### Ориентация дисплея

Программно можно управлять ориентацией дисплея. Доступно четыре режима поворота координатных осей.
В случае дисплеев с разрешением менее 64*132 будет отображаться только часть картинки.



Существует два положения курсора - символьный 132*8 и графический 132*64.
Символьный курсор - количество точек с начала строки и номер строки, графический – количество точек от начала координат.

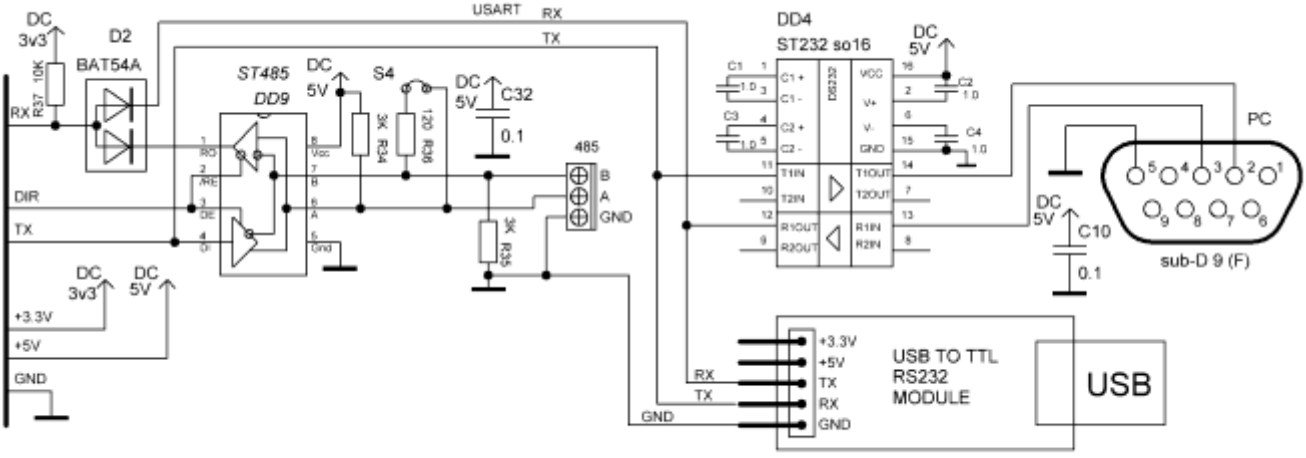
## Операторы и функции работы с дисплеем

| | | | |
|---|---|---|---|
| CLS | Очистка экрана дисплея | **20 CLS** | Очистка экрана дисплея. Курсор устанавливается на 0 позицию. |
| AT | Установка текстового курсора | **AT 10,2,0** | Управление текстовым курсором<br>Устанавливает адрес с начала строки в пикселях и номер строки вывода. Управляет обычный=0 или инверсный вывод = 1, если третий операнд отсутствует, значение инверсии не меняется. |

| | | | |
|---|---|---|---|
| LCDD | Послать данные в дисплей | **20 LCDD $(5)** | Записывает в дисплей значение переменной в регистр данных без перекодировок. Параметрами выступают переменная, число. Записывает 8 точек по вертикали в текущей строке и текущем адресе. Младший бит сверху. Инкремент адреса в строке. Предназначен для вывода картинок из RAW массива, файла. |
| LCDC | Послать команду в дисплей | **20 LCDC #(3)** | Записывает в дисплей значение переменной в регистр команд. См. документацию на UC1601s |
| LPRINT | Оператор вывода. Вывод на ЖКИ сообщений | **LPRINT "Это сообщение"**<br>**LPRINT X**<br>**LPRINT $(45)**<br>**LPRINT "X= ",X,"Y= ",Y**<br>**LPRINT X,Y,Z**<br>**LPRINT #2,A** | Полностью аналогичен оператору PRINT<br>Перевод строки - не поддерживается. Используйте оператор AT. |
| LINIT | Включение и инициализация дисплея | **LINIT X,y,z** | X=0 - OFF ; X=1 - HD44780 ; X=2 - UC6101s, режима ориентации MODE = 0-3, значение контрастности 0-15. При нулевых значениях - устанавливаются по умолчанию. |
| POINT | Установка графического курсора, сброс/ установка/ инверсия пикселя | **POINT X,Y,z** | Устанавливает графический курсор в координаты X,Y.<br>z=0 – очистка пикселя. (x = 0-131, y = 0-63)<br>z=1 – установка пикселя.<br>z=2 – инверсия пикселя.<br>z отсутствует – только установка курсора |
| DRAW | Рисование линий | **DRAW X,Y,z** | Рисует линию от текущих координат до координат X,Y.<br>z=0 – очистка пикселей в линии. (x = 0-131, y = 0-63)<br>z=1 – установка пикселей в линии.<br>z=2 – инверсия пикселей в линии. |

**Схема подключения UART, RS232/RS485 к шине контроллера.**

Как видим, ничего сложного нет, все стандартно. Используем микросхемы соответствующих интерфейсов.

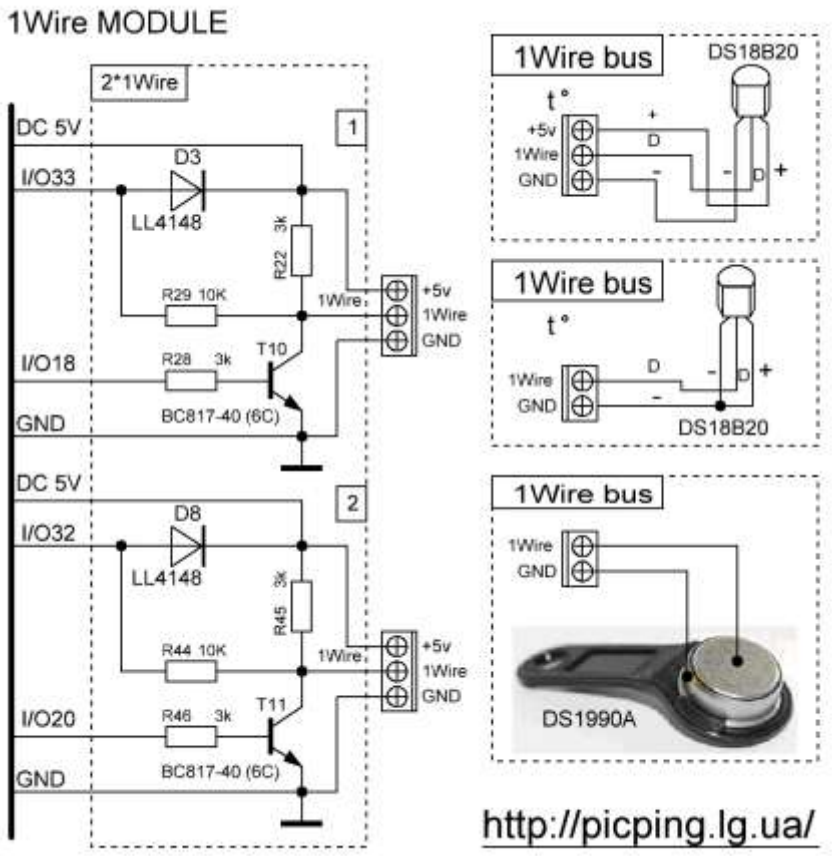



**Операторы и функции работы с периферийными модулями - USART, RS485**

| | | | |
|---|---|---|---|
| UART | Настройка порта | **UART 1,S** | Включает/выключает порт RS232/485 и настраивает скорость. Первый параметр 0 – порт отключен.<br>1 – включен в режиме RS232<br>2 – включен в режиме RS485<br>Второй параметр - скорость порта : 1-1200 8N1, 2-2400 8N1, 3-4800 8N1, 4-9600 8N1, 5-19200 8N1, 6-38400 8N1, 7-115200 8N1 |
| SPRINT | Оператор вывода. Вывод в порт RS232 или RS485 | **SPRINT**<br>**SPRINT "Это сообщение"**<br>**SPRINT "X= ";X;"Y= ";B;**<br>**SPRINT X,Y,Z;** | Полностью аналогичен оператору PRINT<br>Перевод строки - вывод двух символов с кодами 0x0D и 0x0A. |

| | | | |
|---|---|---|---|
| SINPUT#<br><br>SINPUT$<br><br>SINPUT@ | Оператор ввода числовых, текстовых данных<br><br>Окончание ввода – символ 0x0D(Enter). Символ 0x0A(LF) игнорируется. | **SINPUT# X**<br><br>**SINPUT$ $(i)**<br><br>**SINPUT@ #(i)** | Ожидает ввода числа в порту RS232/RS485 в формате от-32768 до 32767 или 0-255 для 8-битных или выражения и присваивает переменной. Окончание ввода – символ 0x0D(Enter). Если входящий поток нельзя интерпретировать как число или выражение , то переменной присвоится значение 0. Выход – ввод данных или BREAK в консоли.<br>Ожидает ввода текста в порту RS232/RS485 до 63 символов. Окончание ввода – символ 0x0D(Enter). В ячейку $(i) будет помещено количество введенных символов а в ячейки $(i+1) и далее (+2,+3…)будет помещен сам текст, включая символ 0x0D.<br>Ожидает ввода 1 байта в порту RS232/RS485 в HEX записи. Окончание ввода – символ 0x0D(Enter). Лишние символы будут отброшены.<br>Если порт не активирован – то прерывание по ошибке. |
| SGET | Оператор ввода | **SGET #(i)** | Ввод массива данных из входного буфера порта RS232/RS485 до 64 байт без ожидания (то что там есть на текущий момент вне зависимости от наличия 0x0D). В ячейку #(i) будет помещено количество введенных байт, а в ячейки #(i+1) и далее (+2,+3…) будет помещены байты из входящего потока без каких либо интерпретаций, как есть. После ввода входящий буфер очищается. Приемником массива данных могут быть только индексные переменные и массив не должен пересекать границы блоков по 256 байт. При переполнении буфера – он сбрасывается на начало приема. |
| SPUT | Оператор вывода | **SPUT #(i)** | Ввод массива данных из переменной #(i) в порт RS232/RS485 до 63 байт, как есть. Количество данных - в ячейке #(i), не выводится. Сами данные в ячейках #(i+1) и далее (+2,+3…). |

**Схема подключения 1-Wire к шине контроллера.**

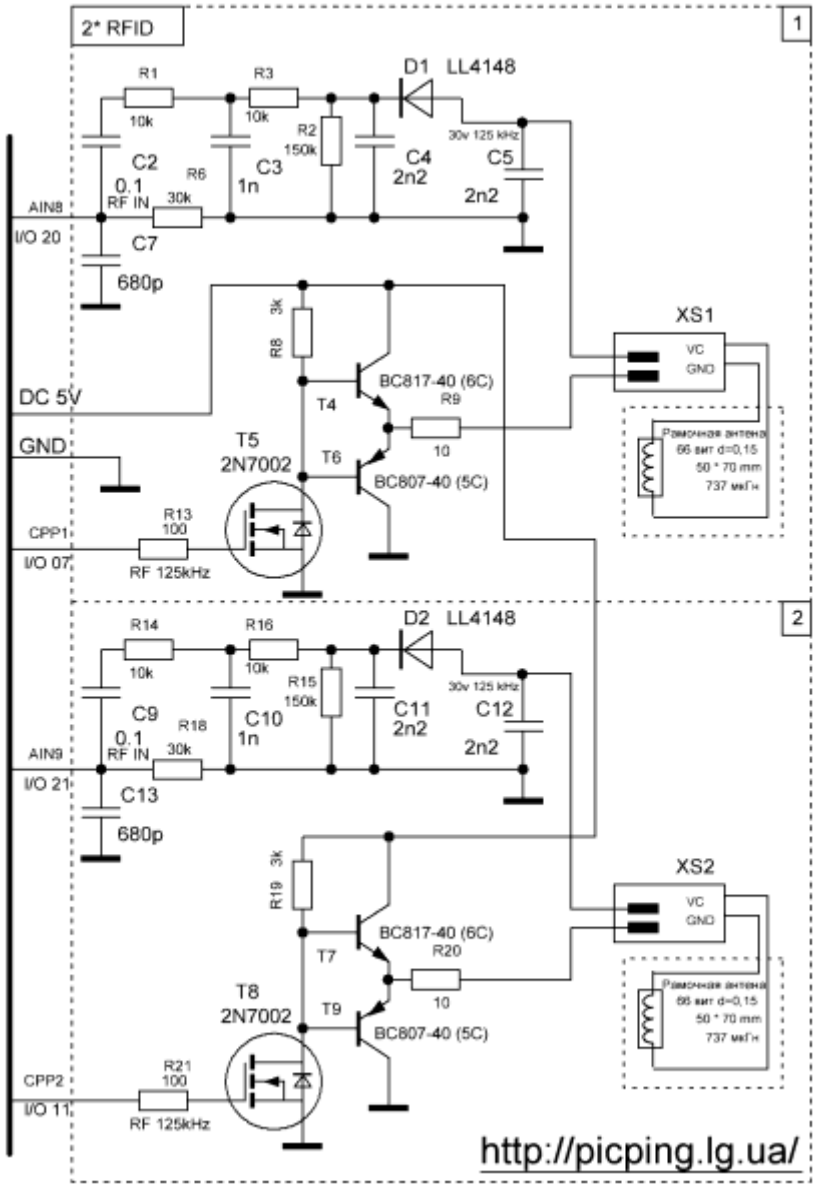



Представлена схема на 2 канала интерфейса 1Wire. Возможно использование каналов, как по отдельности, так и вместе.

**Операторы и функции работы с периферийными модулями - 1-Wire**

| | | | |
|---|---|---|---|
| RROM | Чтение серийного номера 1Wire устройства (33H),например DS1990a или 18B20 | **#(i)=RROM{1}**<br>**#(i)=RROM{2}** | Переменной с индексом **i** присваивается значение «1» в случае обнаружения и «0» в случае отсутствия. Если есть то переменным с индексами от **i+1** до **i+8** присваиваются код устройства - 8 байт Если нет – то не изменяются**.** Выполняется примерно 10 Ms. Параметром выступает номер интерфейса. |
| RTEMP | Функция<br><br>Получение температуры с датчика DS18B20 | **T=RTEMP{1}**<br><br>**T=RTEMP{2,#(i)}** | Переменной **T** присваивается значение температуры со знаком или значение -32768 в случае ошибки/отсутствия датчика. Первым параметром выступает номер интерфейса, вторым параметром может выступать серийный номер датчика -8 байт в переменных от **i** до **i+7** для нескольких датчиков на одной шине. Может отсутствовать. *Выполняется примерно 800 Ms. Производится запуск преобразования для всех датчиков, ожидание, получение данных. Результат - температура с точностью 0,1 градуса. Например полученное значение 125 – это температура 12,5 градуса. |
| STEMP | Оператор | **STEMP 1**<br>**STEMP A** | Запуск преобразования температуры для всех датчиков в указанном канале. |
| GTEMP | Функция<br><br>Получение температуры с датчика DS18B20 | **T=GTEMP{1}**<br><br>**T=GTEMP{2,#(i)}** | Переменной **T** присваивается значение температуры со знаком или значение -32768 в случае ошибки/отсутствия датчика. Первым параметром выступает номер интерфейса, вторым параметром может выступать серийный номер датчика -8 байт в переменных от **i** до **i+7** для нескольких датчиков на одной шине. Может отсутствовать.<br>Производится только получение данных. Ране должна быть выполнена команда STEMP с разницей во времени не менее 800 mS. Результат - температура с точностью 0,1 градуса. Например, полученное значение -34– это температура -3,4 градуса. |

**Схема подключения RFID к шине контроллера.**

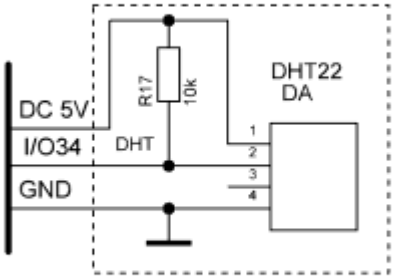



Представлена схема на 2 канала интерфейса RFID. Возможно использование каналов, как по отдельности, так и вместе.

**Операторы и функции работы с периферийными модулями - RFID**

| | | | |
|---|---|---|---|
| RFID | Чтение карты RFID | **#(i)=RFID{1}**<br><br>**#(10)=RFID{b}** | Переменной с индексом **i** присваивается значение «1» в случае обнаружения карты RFID EM marine и «0» в случае отсутствия карты. Если карта есть то переменным с индексами от **i+1** до **i+5** присваиваются код карты - 5 байт. Если нет – то не изменяются**.** Выполняется примерно 150 mS. В качестве параметра выступает номер канала чтения.<br>При использовании отключена первая группа PWM -1 и 2 каналы.<br>Номер канала равный нулю отключает считыватель. |

**Схема подключения AM2302 /DHT22 к шине контроллера.**



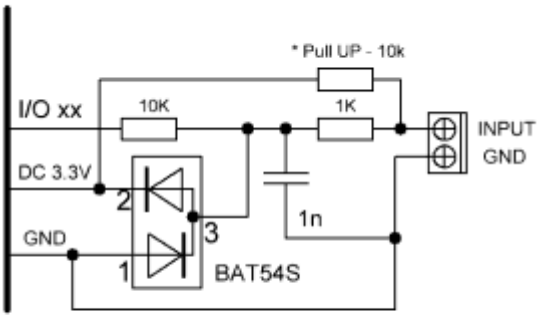

**Операторы и функции работы с периферийными модулями - AM2302 /DHT22**

| | | | |
|---|---|---|---|
| GDHTT | Получение температуры с датчика DHT22 | **T=GDHTT{}** | Переменной присваивается значение температуры с датчика DHT22 в градусах Цельсия. Результат - температура с точностью 0,1 градуса. Например полученное значение 125 – это температура 12,5 градуса. В случае ошибки/отсутствия датчика - значение -32768. |
| GDHTH | Получение влажности с датчика DHT22 | **H=GDHTH{}** | Переменной присваивается значение влажности с датчика DHT22 в процентах. Результат - влажность с точностью 0,1 процента. Например полученное значение 525 – это температура 52,5 процента. В случае ошибки/отсутствия датчика - значение -32768. |

## Дискретные входы

Для использования входов контроллера с датчиками типа «сухой контакт» требуется применение резисторов подтяжки к логической единице (+3,3V). Номинал резистора выбирается исходя из длинны линии и тока через контакт. Также следует принять меры для защиты входа контроллера от импульсных наводок или статического электричества при длинном шлейфе от контакта до контроллера.

Для ответственных приложений следует применять только контакты на размыкание, а в случае использования герконов в качестве датчиков – дублировать контакты. Устанавливать попарно последовательно при работе на размыкание.




Типовая схема дискретного входа с защитой контроллера и общей землей.
В случае невозможности общей земли или при работе с устройствами имеющими связь с сетью переменного тока требуется применять опторазвязки входов и/или выходов или применять реле.

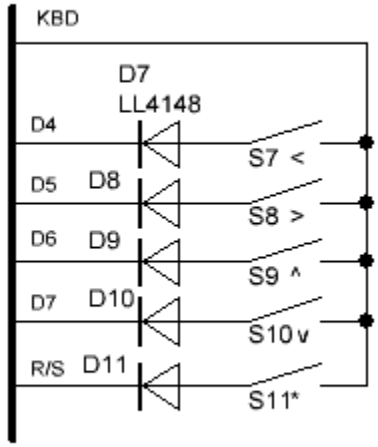
**Операторы и функции работы с дискретными входами**

| | | | |
|---|---|---|---|
| GKEY | Опрос входов без изменения направления ввода-вывода. | **A=GKEY{c,e,b}** | Где **b**- битовая маска опроса, **c**- тип опроса, **e** – номер канала.<br>Необходимые порты должны быть ранее настроены на ввод .<br>Канал 1: I/O 1-I/O15. Канал 2: I/O 16-I/O30. Канал 3: I/O 31-I/O34.<br>Битовая маска – если например b=5 (2^0*1 +2^1*0 + 2^2*1+2^3*0….) то опрашиваются входа I/O 1 и I/O 3 для 1 канала. b=1-32767<br>Тип опроса.<br>- Если c=0 то опрос производится в момент вызова функции.<br>В переменную записывается текущее состояние входов. Выходные данные – если на входе высокий уровень то соответствующий бит будет установлен в единицу и наоборот. Например I/O 1 =0 и I/O 3 =1 , вызов **A=GKEY{0,1,5}** даст значение **A=4.**<br>- Если c=1 то переменной A присваивается битовое значение номера входа, на котором было изменение состояния от момента предыдущего вызова **GKEY** или **WKEY**. Текущее состояние берется базой состояния для следующего вызова **GKEY** . Опрос входов производится в фоновом режиме.<br>Например было изменение на входах 1,3,4 и маска =5 тогда A = 5<br>Минимальное время изменения для регистрации – более 1mS |
| WKEY | Ожидание указанного состояния или изменение состояния входов без изменения направления ввода-вывода. | **A=WKEY{c,e,b}** | Ожидает изменения состояния порта и присваивает значение переменной.<br>Выход - изменение или BREAK в консоли.<br>Необходимые порты должны быть ранее настроены на ввод .<br>Где b- битовая маска опроса, c- тип опроса, e – номер канала.<br>Переменной A присваивается битовое значение номера входа, на котором было изменение состояния.<br>Если c=0 – ожидается появление логического нуля на любом входе, определенным маской. Если на момент вызова уже 0 присутствует, то сразу присваивается значение.<br>Если c=1 – ожидается появление логической единицы на любом входе, определенным маской. Если на момент вызова уже 1 присутствует, то сразу присваивается значение.<br>Если c=2 – ожидается изменение состояния на любом входе, определенным маской. Изменением считается изменение состояния после вызова функции. |

Также можно применять функции и операторы PORT, см. описание в стандартных командах.

## Клавиатура

Подключение 5 кнопок в случае применения дисплея  HD44780 позволяет освободить поры ввода вывода




Как правило это кнопки со стрелками и «ок» для организации разнообразных меню управления.
Функции получения кодов кнопок аналогичны предыдущим, вызываются без параметров, работают после инициализации дисплея.

| | | | |
|---|---|---|---|
| GKEY | Опрос кнопки/кнопок | **A=GKEY{}** | В переменную записывается  текущее состояние кнопок. 0- не нажата. Нажата первая- добавляется 1, вторая  -2, третья -4,четвертая -8, пятая -16. Например 5 соответствует нажатой 1 и3 кнопки |
| WKEY | Ожидание нажатия кнопки/кнопок | **#(i)=WKEY{}** | Ожидает нажатия любой кнопки и присваивает значение переменной. Выход  - нажатие кнопки или BREAK в консоли |
| KEY | Получить код кнопки/кнопок | **A=KEY{15}** (NEW v3.4.1) | Ожидает отпускания всех кнопок, затем ожидает нажатия любой кнопки указанное время в секундах (max=255) и присваивает значение переменной. Возвращает код кнопки, или в 0 в случае таймаута (не нажата). Время может быть указанно переменной, числом, выражением. |

---

## MODBUS RTU MASTER

Данный контроллер поддерживает интерфейс связи с подчиненными контроллерами  DEVICE PICμBASIC http://picping.lg.ua/device_pic_basic/index.htm . В качестве протокола выбран распространенный стандарт MODBUS RTU.

Для управления и контроля подчиненными контроллерами использованы стандартные функции протокола. Применительно к встроенному языку операторы поддерживают обмен значениями переменных BASIC  между контроллерами по инициативе мастер устройства. Для подчиненного устройства обмен происходит в фоновом режиме, не зависимо от исполняемой там программы.

Дополнительно реализован служебный обмен данными для прозрачной работы удаленного терминала и загрузки обновленных прошивок через сеть Ethernet

Общий алгоритм обмена выглядит следующим образом.
- Мастер (WEB PICμBASIC) выполняет программу и с помощью соответствующих операторов  запрашивает переменную или передает переменную в подчиненный контроллер с выбранным адресом.
- Слейв  DEVICE PICμBASIC в фоновом режиме, не прекращая выполнения своей программы, обрабатывает запрос и выполняет указанные действия. Модифицирует свою переменную в соответствии с принятым значением  или  передает значение своей  переменной.  В случае такого обмена имена переменных в обоих контроллерах совпадают.
После выполнения операции Слейв отсылает подтверждающий пакет о выполненных действиях или пакет с данными.
- Мастер принимает ответ и возвращает из функции статус операции – успешно или ошибка, также принимает данные в указанную переменную, если пришли данные. Время выполнения данных функций вариабельно и зависит от времени ответа слейв устройства. Максимальное время ожидания ответа до ошибки тайм аута  - 500 mS.
Дополнительную информацию о временных параметрах обмена  смотрите в приложении.

Доступ в режиме терминала.

Если программа остановлена, и контроллеры находятся в терминальном режиме, то появляется доступ к терминалу слейв устройства. Для этого необходимо в программе терминала, после соединения с WEB PICµBASIC перевести в режим MODBUS с указанием адреса подчиненного устройства. После этого все команды от и к терминалу будут транслироваться на подчиненное устройство. Аналогично работает загрузка и выгрузка программ на Basic. Для удаленного обновления ПО Basic следует использовать Ethernet загрузчик .

В дальнейшем Слейв устройство может работать, как с обменом данными с мастером, или как самостоятельное устройство.

**Адреса MODBUS RTU**

Диапазон адресов – 1-127
Адрес = 0 – широковещательный – принимают все контроллеры. С этим адресом выполняется только одна команда - генерация нового случайного адреса.

Внимание, пред использованием этих команд необходимо включить и настроить порт UART – интерфейс RS485 115200 8N1

| | | | |
|---|---|---|---|
| MDC<br>MDB | Оператор запроса консоли для DEVICE PICµBASIC | **MDC a,data**<br>**MDB a,data** | Отправляет текстовый пакет data, в устройство с адресом **a,** заданным переменной или числом, с номером функции **0x43** или **0x42**. Добавляет заголовок, символы конца строки и контрольную сумму.<br>Ожидание ответа происходит в фоновом режиме для последнего преданного адреса и функции. В случае успешного приема проверяет контрольную сумму и передает содержимое в консоль.<br>В консоль контрольная сумма и заголовок пакета не передается. Для отключения прослушивания – вызвать с нулевым адресом и пустой строкой. |
| MDWR | Функция записи переменной для DEVICE PICµBASIC **16 (0x10)** — запись значений в несколько регистров хранения (Preset Multiple Registers) | **A=MDWR{a,D}**<br>**A=MDWR{a,#(5)}**<br>**A=MDWR{a,$(i)}** | В контроллер DEVICE PICµBASIC с адресом **a** из контроллера WEB PICµBASIC копируется переменная, указанная после запятой. В случае текстовой переменной, копируется весь текст(массив) до 63 байт.<br>Переменной **A** присваивается значение 1 в случае удачной операции или 0 в случае ошибки / недоступности.<br>В контроллере DEVICE PICµBASIC изменение значения происходит в фоновом режиме. |
| MDRD | Функция чтения переменной для DEVICE PICµBASIC **3 (0x03)** — чтение значений из одного или нескольких регистров хранения *(Read Holding Registers)*. | **A=MDRD{a,D}**<br>**A=MDRD{a,#(5)}**<br>**A=MDRD{a,$(i)}** | Из контроллера DEVICE PICµBASIC с адресом **a** в контроллер WEB PICµBASIC копируется переменная, указанная после запятой. В случае текстовой переменной, копируется весь текст(массив) до 63 байт.<br>Переменной **A** присваивается значение 1 в случае удачной операции или 0 в случае ошибки / недоступности.<br>В контроллере DEVICE PICµBASIC чтение происходит в фоновом режиме. |
| MDRI | Функция чтения информации для DEVICE PICµBASIC **17 (0x11)** — Чтение информации об устройстве (Report Slave ID) | **$(0)= MDRI{a}** | В текстовую переменную **$(0)** читается информация об устройстве с адресом **a**. В случае отсутствия или недоступности возвращается пустая строка. |
| **Операторы и функции работы с модулями расширения на шине MODBUS** | | | |
| MDBSR | Функция чтения данных устройства MODBUS ( функция 0x41) Для модулей расширения датчиков и выходов | **A=MDBSR{a,N}** | Отправляет пакет в устройство с адресом = **a**, на чтение одного параметра с адресом параметра = **N**. Ожидает ответ не более 50 мс. Переменной **A** присваивает значение из ответа. Если ответа не было, переменной присваивается значение -32768. Устройство также может отдать значение -32768 если данный параметр не используется в нем. |
| MDBSW | Функция записи данных устройства MODBUS ( функция 0x42) Для модулей расширения датчиков и выходов | **A=MDBSW{a,N,D}** | Отправляет пакет в устройство с адресом = **a**, на запись одного параметра с адресом параметра = **N**, значением **D**. Ожидает ответ не более 50 мс. Переменной A присваивает значение из ответа. Если ответа не было или запись не удачна то переменной присваивается значение -32768 Если запись удачна, то **A=D**. |

| | | | |
|---|---|---|---|
| MDwr | Функция записи **6 (0x06)** — запись значения в один регистр хранения *(Preset Single Register)*. | **A=MDwr{a,M,D}**<br>**A=MDwr{a,#(5),#(6)}** | Во внешнее устройство с адресом **a** из контроллера PICμBASIC записывается значение **D** в регистр с адресом **M**.<br>Переменной **A** присваивается значение 1 в случае удачной операции или 0 в случае ошибки / недоступности. |
| MDrd | Функция чтения **3 (0x03)** — чтение значений из одного или нескольких регистров хранения *(Read Holding Registers)*. | **A=MDrd{a,M,D}**<br>**A=MDrd{a,#(5),#(6)}** | Из внешнего устройства с адресом **a** в контроллер PICμBASIC читается значение регистра с адресом **M** и помещается в переменную **D** .<br>Переменной **A** присваивается значение 1 в случае удачной операции или 0 в случае ошибки / недоступности. В случае не удачи, значение в переменной **D** не изменяется. Поддерживается чтение только одного регистра за 1 раз. |
| | | | |

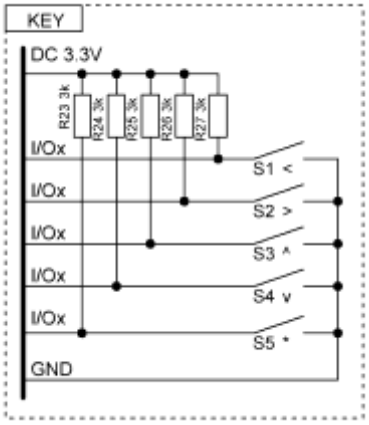
## Клавиатура и силовые выходы

Для организации интерактивного управления к контроллеру подключаются дисплей и кнопки. Рассмотрим подключение кнопок к контроллеру

Как правило достаточно всего пяти кнопок: влево, вправо, вверх , вниз, Ок. Кнопками влево- вправо выбирается параметр ,который требуется изменить. Кнопкам и вверх и вниз производится изменение значения , а кнопкой Ок подтверждается принятие изменений.

Опрос кнопок происходит оператором GKEY, ожидание нажатия - WKEY. Описание операторов в разделе «Дискретные входы. Схема подключения. Список операторов»

Стандартная схема подключения кнопок




Номера входов I/O следует выбирать в диапазоне одного канала проса оператора – 1-15 , 16-30, 31-34

Например I/O17-I/O21
Тогда используем
**A=GKEY{x,2,62}** (анализ с бита 1 по бит 5)
Где x – тип опроса
Если x=0 то в переменную A запишется состояние кнопок в момент вызова функции.
Если x=1 то A присваивается битовое значение номера входа, на котором было изменение состояния от момента предыдущего вызова **GKEY** или **WKEY**. Текущее состояние берется базой состояния для следующего вызова

A = 2*inp1 + 4*inp2 + 8*inp3 + 16*inp4 + 32*inp5

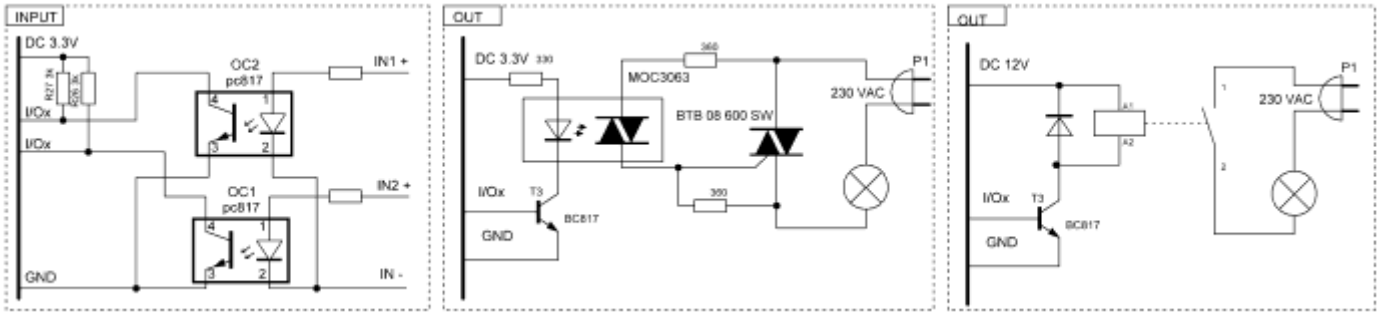
Функция **WKEY** производит ожидание нажатия кнопки, попадающей в маску.

Пример программы опроса кнопок на I/O 17- 21

```
0010 PAUSE 50
0014 REM WAIT ALL KEY UNPRESS
0015 A=GKEY{0,2,62}:IF A<>62 THEN GOTO 10
0018 REM WAIT PRESS ANY KEY
0020 A=WKEY{0,2,62}
0048 REM SELECT AND PRINT NUMBER KEY
0050 CASE A,2,200,4,250,8,300,16,350,32,400
0200 PRINT "PRESS #1":GOTO 10
0250 PRINT "PRESS #2":GOTO 10
0300 PRINT "PRESS #3":GOTO 10
0350 PRINT "PRESS #4":GOTO 10
0400 PRINT "PRESS #5":GOTO 10
```

Управление мощной нагрузкой осуществляется через силовые ключи или реле. Если нагрузка не связана со схемой электрически, то всегда следует применять реле или опторазвязку.

Примеры схем подключения изолированных входов /выходов




Управление из программы - используем оператор PORT
Пример для подключения реле к I/O 4- 7 с управлением от кнопок с предыдущего примера

```
0010 PAUSE 50
0014 REM WAIT ALL KEY UNPRESS
0015 A=GKEY{0,2,62}:IF A<>62 THEN GOTO 10
0018 REM WAIT PRESS ANY KEY
0020 A=WKEY{0,2,62}
0048 REM SELECT AND PRINT NUMBER KEY
0050 CASE A,2,200,4,250,8,300,16,350,32,400
0200 PRINT "RELAY #1 ON/OFF":PORT 4,2:GOTO 10
0250 PRINT "RELAY #2 ON/OFF":PORT 5,2:GOTO 10
0300 PRINT "RELAY #3 ON/OFF":PORT 6,2:GOTO 10
0350 PRINT "RELAY #4 ON/OFF":PORT 7,2:GOTO 10
0400 PRINT "ALL RELAY OFF":PORT 4,0:PORT 5,0:PORT 6,0
0410 PORT 7,0:GOTO 10
```

# ПРИЛОЖЕНИЯ

## Формат пакетов MODBUS MODBUS RTU PICµBASIC

В качестве регистров хранения *(Read Holding Registers)* в контроллерах выступают переменные.

**3 (0x03)** — чтение значений из одного или нескольких регистров хранения *(Read Holding Registers)*.

Запрос состоит из адреса первого элемента таблицы, значение которого требуется прочитать, и количества считываемых элементов. Адрес и количество данных задаются 16-битными числами, старший байт каждого из них передается первым.
В ответе передаются запрошенные данные. Количество байт данных зависит от количества запрошенных элементов. Перед данными передается один байт, значение которого равно количеству байт данных.

Формат пакетов MODBUS

Запрос

1 байт – Адрес устройства
2 байт – Функция 0x03
3 байт - Адрес первой ячейки(параметра) (HIGH)
4 байт - Адрес первой ячейки(параметра) (LOW)
5 байт - Число ячеек(HIGH)
6 байт - Число ячеек(LOW)
7 байт - CRC (LOW)
8 байт - CRC (HIGH)

Ответ

1 байт – Адрес устройства
2 байт – Функция 0x03
3 байт - Счетчик байт данных
4 байт - Данные(addr) singed (HIGH)
5 байт - Данные(addr) singed (LOW)
--/ *+1 байт --/ Данные(addr+1) singed (HIGH)
--/ *+2 байт --/ Данные(addr+1) singed (LOW)
...
7/n-1 байт - CRC (LOW)
8/n байт - CRC (HIGH)

В случае указания числа ячеек = 2 происходит чтение/запись одной переменной singed 16 бит (A-Z, a-z).
В случае указания числа ячеек = 1 происходит чтение/запись индексной переменной.
Любая переменная передастся двумя байтами, в случае индексной переменной, старший = 0.
В случае указания числа ячеек = 0 происходит чтение/запись текстовой переменной с ее длинной (до 63 байт).

**16 (0x10)** — запись значений в несколько регистров хранения (Preset Multiple Registers)

Запрос

1 байт – Адрес устройства

2 байт – Функция 0x10
3 байт - Адрес первой ячейки(параметра) (HIGH)
4 байт - Адрес первой ячейки(параметра) (LOW)
5 байт - Число ячеек (параметра) (HIGH)
6 байт - Число ячеек (параметра) (LOW)
7 байт - Счетчик байт данных
8 байт - Данные(addr) singed (HIGH)
9 байт - Данные(addr) singed (LOW)
--/ *+1 байт --/ Данные(addr+1) singed (HIGH)
--/ *+2 байт --/ Данные(addr+1) singed (LOW)
...
7/n-1 байт - CRC (LOW)
8/n байт - CRC (HIGH)

Ответ

1 байт – Адрес устройства
2 байт – Функция 0x06
3 байт - Адрес первой ячейки(параметра) (HIGH)
4 байт - Адрес первой ячейки(параметра) (LOW)
5 байт - Число ячеек (параметра) (HIGH)
6 байт - Число ячеек (параметра) (LOW)
7 байт - CRC (LOW)
8 байт - CRC (HIGH)

В случае указания числа ячеек = 2 происходит чтение/запись одной переменной singed 16 бит (A-Z, a-z).
В случае указания числа ячеек = 1 происходит чтение/запись индексной переменной.
В случае указания числа ячеек = 0 происходит чтение/запись текстовой переменной с ее длинной (до 63 байт).

**17 (0x11)** — Чтение информации об устройстве (Report Slave ID)

Пример
-> ADR, 0x11, CRC_L, CRC_H
<- ARD, 0x11, кол-во байт, text- xx BYTES , CRC_L, CRC_H

Запрос

1 байт – Адрес устройства
2 байт – Функция 0x11
3 байт - CRC (LOW)
4 байт - CRC (HIGH)

Ответ

1 байт – Адрес устройства
2 байт – Функция 0x11
3 байт - Счетчик байт данных
4 байт и далее – Данные TEXT max 64b
...
/n-1 байт - CRC (LOW)
/n байт - CRC (HIGH)

**Пользовательские функции**

**66 (0x42)** — Данные BOOTLOADER

-> ARD, 0x42, text- xx BYTES , CRC_L, CRC_H
<- ARD, 0x42, text- xx BYTES , CRC_L, CRC_H

Запрос

1 байт – Адрес устройства
2 байт – Функция 0x42
3 байт и далее – Данные TEXT max 64b
/n-1 байт - CRC (LOW)
/n байт - CRC (HIGH)

Ответ

1 байт – Адрес устройства
2 байт – Функция 0x42
3 байт и далее – Данные TEXT max 64b
...
/n-1 байт - CRC (LOW)
/n байт - CRC (HIGH)

**67 (0x43)** — Данные терминала

-> ARD, 0x43, кол-во байт xx, text- xx BYTES , CRC_L, CRC_H
<- ARD, 0x43, кол-во байт xx, text- xx BYTES , CRC_L, CRC_H

Запрос

1 байт – Адрес устройства
2 байт – Функция 0x43
3 байт - Счетчик байт данных
4 байт  и далее – Данные  TEXT max 64b
...
/n-1 байт  - CRC (LOW)
/n    байт  - CRC (HIGH)

Ответ

1 байт – Адрес устройства
2 байт – Функция 0x43
3 байт - Счетчик байт данных
4 байт  и далее – Данные  TEXT max 64b
...
/n-1 байт  - CRC (LOW)
/n    байт  - CRC (HIGH)

## Формат пакетов MODBUS RTU  модулей расширения.

Запрос

1 байт – Адрес устройства
2 байт - Функция
3 байт - Адрес первой ячейки(параметра) (HIGH)
4 байт - Адрес первой ячейки(параметра) (LOW)
5 байт – Чтение - Число ячеек(HIGH) / Запись – Данные singed  (HIGH)
6 байт  - Чтение - Число ячеек(LOW) / Запись  - Данные singed  (LOW)
7 байт  - CRC (LOW)
8 байт  - CRC (HIGH)

Адресное пространство, выбор адреса – перемычками.

1 байт - Адрес = 0x10 - 0x1F  - Блок датчиков
1 байт - Адрес = 0x20 - 0x2F  - Блок силовых выходов

Поддерживаемые функции  устройствами Блок датчиков  и Блок силовых выходов

2 байт - Функции

17 (0x11) — Чтение информации об устройстве (Report Slave ID)
Пример
-> ADR, 0x11, CRC_L, CRC_H
<- ARD, 0x11, text- 32 BYTES , CRC_L, CRC_H

65 (0x41) Чтение данных устройства

-> ADR, 0x41, адрес ячейки -2 байта, число параметров 2 байта = 1(0x01 0x00), CRC_L, CRC_H
<- ARD, 0x41, число байт  - 1байт = 0x02 , Данные  2 байта  singed , CRC_L, CRC_H

66 (0x42) Запись данных в устройство

-> ADR, 0x42, адрес ячейки -2 байта, Данные  2 байта  singed CRC_L, CRC_H
<- ARD, 0x42, адрес ячейки -2 байта, Данные  2 байта  singed CRC_L, CRC_H
В случае успешной записи  данные выходные  =  входным, не успешной  данные выходные  = -32768 (0x80,0x00)

70 (0x46) Запуск загрузчика (reboot)

-> ADR, 0x46, CRC_L, CRC_H
<- ADR, 0x46, CRC_L, CRC_H

Адреса ячеек данных и  их возможные значения
В случае отсутствия/неисправности возвращает значение  -32768

Блок датчиков
0  18B20 #1 температура  (-55 … +127)
1  18B20 #2 температура  (-55 … +127)
2  18B20 #3 температура  (-55 … +127)
3  18B20 #4 температура  (-55 … +127)
4  ADC 1  Значение АЦП (0-255)
5  ADC 2  Значение АЦП (0-255)
6  ADC 3  Значение АЦП (0-255)
7  ADC 4  Значение АЦП (0-255)
8  DHT22  температура (-400.. +800 = -40,0C - +80,0C)
9  DHT22  влажность  (0-1000 = 0,0% - 100,0%)

Блок реле

Формат запроса - ответа

Адрес
Функция
Адрес первой ячейки (HIGH)
Адрес первой ячейки(LOW)
R- Число ячеек(HIGH)/W-data high
R- Число ячеек(LOW)/W-data low
**********

1 быйт - Адрес = 0x20 - 0x2F

2 байт - команды
17 (0x11) — Чтение информации об устройстве (Report Slave ID)

-> ADR, 0x11,crc,crc
<- ARD, 0x11, text- 32 BYTES ,crc,crc

65 (0x41) Чтение данных устройства

-> ADR, 0x41, addr register 2bytes, число параметров 2bytes = 1(01 00), crc,crc
<- ARD, 0x41, число байт 1 bytes = 2 (02 ), data 2 BYTES singed ,crc,crc

66 (0x42) Запись данных в устройство

-> ADR, 0x42, addr register 2bytes, data singed 2bytes (xx xx), crc,crc
<- ARD, 0x42, addr register 2bytes, data singed 2bytes (xx xx), crc,crc

70 (0x46) Запуск загрузчика

-> ADR, 0x46,crc,crc
<- ADR, 0x46,crc,crc

CRC - 2 байта

Адреса регистров

| | | |
|---|---|---|
| 0 | OUT #1 | R/W |
| 1 | OUT #2 | R/W |
| 2 | OUT #3 | R/W |
| 3 | OUT #4 | R/W |
| 4 | IN #1 | R |
| 5 | IN #2 | R |
| 6 | IN #3 | R |
| 7 | IN #4 | R |
| 0x8 | | |
| 0x9 | | |
| 0xA | | |
| 0xB | | |
| 0xc | | |
| 0xd | | |
| 0xe | | |
| 0xf | | |

Для адресов 0-3 возможна запись и чтение.
Запись «1» в регистр обозначает включение, «0» - выключение
Чтение возвращает текущее состояние

Для адресов 4-7 возможно только чтение.
Значение «1» в регистре обозначает замкнутый вход , «0» - разомкнутый
На входы подключаются концевые выключатели или «сухой контакт»

**Обработка ошибок**

Ведущий отправляет запрос к Ведомому, в котором в поле «код функции» указывает ему на необходимое действие.
Байты данных содержат информацию, необходимую для выполнения данной функции.
Ведомый, в случае удачного выполнения этой функции, повторяет код функции в ответе.
При возникновении ошибки, код функции в ответе модифицируется — старший бит выставляется в 1.
В байтах данных передается причина ошибки. Например при исполнении Ведомым функции **0x0F** возникла ошибка, тогда он ответит Ведущему полем функции равным **0x8F**.
В дополнении к изменению кода функции, Ведомый размещает в поле данных уникальный код, который указывает на тип и причину ошибки.

**Стандартные коды ошибок**

01 — Принятый код функции не может быть обработан.

02 — Адрес данных, указанный в запросе, недоступен.

03 — Значение, содержащееся в поле данных запроса, является недопустимой величиной.

04 — Невосстанавливаемая ошибка имела место, пока ведомое устройство пыталось выполнить затребованное действие.

05 — Ведомое устройство приняло запрос и обрабатывает его, но это требует много времени. Этот ответ предохраняет ведущее устройство от генерации ошибки тайм-аута.

06 — Ведомое устройство занято обработкой команды. Ведущее устройство должно повторить сообщение позже, когда ведомое освободится.

07 — Ведомое устройство не может выполнить программную функцию, заданную в запросе. Этот код возвращается для неуспешного программного запроса, использующего функции с номерами 13 или 14. Ведущее устройство должно запросить диагностическую информацию или информацию об ошибках от ведомого.

08 — Ведомое устройство при чтении расширенной памяти обнаружило ошибку паритета. Ведущее устройство может повторить запрос, но обычно в таких случаях требуется ремонт

**CRC-16 -** циклически избыточный код - полином A001h

Физический интерфейс – RS485 115200 8N1

**Адресное пространство**

Диапазон адресов – 1-127
Адрес = 0 – широковещательный – принимают все контроллеры. С этим адресом выполняется только одна команда - генерация нового случайного адреса.

**Временные диаграммы**
Обработка запросов с функциями **3 (0x03), 16 (0x10), 17 (0x11)** всегда выполняется в фонововом режиме, независимо от программы пользователя. Одновременно может выполняться только один запрос. Минимальное время тишины 1,5 ms. Минимальное время до начала ответа – 0mS, типичное 1mS, максимальное 200 mS (если контроллер занят функциями работы с другими критичными во времени интерфейсами). Время ожидания ответа мастером до ошибки таймаута - 500 mS.
Обработка запроса **67 (0x43)** возможна в случае остановленной программы пользователя(консоль). Если запущена, то обрабатывается только одна команда - **BREAK**.
Обработка запроса **66 (0x42)** возможна только в режиме BOOTLOADER. Минимальное время тишины 300us.
В случае ответа множеством посылок, сам контроллер формирует время тишины 2,5 mS.

Для модулей расширения
Минимальное время тишины 1,5 ms. Минимальное время до начала ответа – 0mS, типичное 1mS, максимальное 30 mS.

## Примеры программирования

**Экранное меню с выводом часов и изменением 4-х параметров.**

Используется LCD дисплей и 5 кнопок, с подключением описанным ранее.
Кнопки подключены i/o17-21

Нажатие кнопок сопровождается звуковым сигналом

Изменяемые параметры хранятся в #(0),#(1),#(2),#(3)
i - указатель на выбранный парметр
j – тип отображения на дисплее; 0 часы, 1 заставка, 2 редактирование
$(10) – текстовая переменная с текущим значением времени
Тип нажатых кнопок дополнительно выводится в консоль.

Собственно программа с комментариями

```
0005 LINIT 1:REM ИНИЦИАЛИЗАЦИЯ ДИСПЛЕЯ
0010 PAUSE 20:REM СТАРТ, ОЖИДАНИЕ ОТПУСКАНИЯ
0011 A=GKEY{0,2,62}:IF A<>62 THEN GOTO 10
0012 REM j - ВИД ДИСПЛЕЯ, i-НОМЕР ПРАМЕТРА
0013 REM #(i) САМИ ПАРМЕТРЫ, $(10)- ВРЕМЯ
0018 REM WAIT PRESS ANY KEY
0020 PAUSE 50:A=GKEY{0,2,62}
0022 IF j=0 THEN GOSUB 510
0024 IF A=62 THEN GOTO 20
0025 REM НАЖАТА, Генерация звука
0030 BEEP 15,2
0040 REM SELECT AND PRINT NUMBER KEY
0042 CASE A,60,50,58,60,54,70,46,80,30,90
0043 REM выбор действия по коду кнопки -OK
0050 PRINT "PRESS OK"
0051 CASE j,0,55
```

```
0052 j=0:GOSUB 510
0053 GOTO 10
0055 j=1:GOSUB 520
0056 GOTO 10
0057 REM Нажали влево, меняем индекс
0060 PRINT "PRESS LEFT":j=2
0061 IF i>0 THEN i=i-1
0063 GOSUB 500
0064 GOTO 10
0070 PRINT "PRESS UP":j=2
0071 IF #(i)<255 THEN #(i)=#(i)+1
0072 GOSUB 500
0073 GOTO 10
0074 REM Нажали вниз,Уменьшим по индексу
0080 PRINT "PRESS DOWN":j=2
0081 IF #(i)>0 THEN #(i)=#(i)-1
0082 GOSUB 500
0084 GOTO 10
0085 REM Нажали вправо
0090 PRINT "PRESS RIGHT":j=2
0091 IF i<3 THEN i=i+1
0092 GOSUB 500
0093 GOTO 10
0094 REM Подпрограммы
0500 REM ОТОБРАЖЕНИЕ ПАРАМЕТРА
0501 AT 0:LPRINT " ПРАМЕТР #",i,"     "
0502 AT 40:LPRINT "ЗНАЧЕНИЕ = ",#(i),"    "
0503 RETURN
0504 REM
0510 REM ОТОБРАЖЕНИЕ НАЧАЛО
0511 AT 0:LPRINT "   РАБОТАЕМ       "
0512 STIME $(10):AT 40:LPRINT "   ",*$(10),"       "
0513 RETURN
0514 REM
0520 REM ОТОБРАЖЕНИЕ ПАРМЕТРОВ
0521 AT 0:LPRINT " ПАРАМЕТРЫ  0-3    "
0522 AT 40:LPRINT ~1,#(0),#(1),#(2),#(3)
0523 RETURN
```

Она же в оптимизированном виде

```
0005 LINIT 1:REM ИНИЦИАЛИЗАЦИЯ ДИСПЛЕЯ
0010 PAUSE 20:A=GKEY{0,2,62}:IF A<>62 THEN GOTO 10
0020 PAUSE 50:A=GKEY{0,2,62}:IF j=0 THEN GOSUB 110
0024 IF A=62 THEN GOTO 20
0030 BEEP 15,2
0042 CASE A,60,50,58,60,54,70,46,80,30,90
0050 PRINT "PRESS OK"
0051 CASE j,0,55
0052 j=0:GOSUB 110
0053 GOTO 10
0055 j=1:AT 0:LPRINT " ПАРАМЕТРЫ  0-3    "
0056 AT 40:LPRINT ~1,#(0),#(1),#(2),#(3):GOTO 10
0060 PRINT "PRESS LEFT":j=2:IF i>0 THEN i=i-1
0061 GOTO 100
0070 PRINT "PRESS UP":j=2:IF #(i)<255 THEN #(i)=#(i)+1
0071 GOTO 100
0080 PRINT "PRESS DOWN":j=2:IF #(i)>0 THEN #(i)=#(i)-1
0081 GOTO 100
0090 PRINT "PRESS RIGHT":j=2:IF i<3 THEN i=i+1
0100 AT 0:LPRINT " ПРАМЕТР #",i,"     "
0101 AT 40:LPRINT "ЗНАЧЕНИЕ = ",#(i),"    ":GOTO 10
0110 AT 0:LPRINT "   РАБОТАЕМ       ":STIME $(10)
0111 AT 40:LPRINT "   ",*$(10),"       ":RETURN
```

При этом коды клавиш и текущее время можно одновременно просматривать на WEB странице.

# Расширения

Модуль COZIR™ Ambient Sensor

Данный модуль расширения поставляется по отдельному заказу.

**Операторы и функции работы с модулем Cozir**

| | | | |
|---|---|---|---|
| CZINI | Оператор Инициализация | **CZINI** | Инициализация датчика COZIR.<br>Устанавливается режим опроса, и усреднение данных 1/16.<br>Устанавливается автокалибровка 1/8 дней<br>Очищается буфер приема. Выполняется примерно 400mS. |
| CZCMD | Оператор команд для Cozir | **CZCMD "A 32"**<br>**CZCMD "@ 1.0 8.0"**<br><br>**CZCMD "@ 0"**<br>**CZCMD "K 2"**<br><br>**CZCMD "X 400"** | Подает команду датчику COZIR. Сама команда пишется в кавычках<br>Примеры команд: (параметр(ы) перечисляются через пробел: «A_32»)<br>«A 32» A – установка цифрового фильтра 1/32, допустимо 1,2,4,8 … 64<br>«@ 1.0 8.0» - автокалибровка 1/8 дней<br>«@ 0» - автокалибровка выключена<br>«K 2» - установка режима опроса, совместимого с функцией CZRD (выполняется в команде CZINI)<br>«X 400» - калибровка CO2 400 ppm<br>Формат остальных команд смотрите в описании на COZIR |
| CZRD | Функция | **A=CZRD{i}**<br><br>**A=CZRD{#(0)}**<br>**A=CZRD{1}**<br>**A=CZRD{2}**<br>**A=CZRD{3}** | Получает данные с датчика Cozir. Датчик должен быть предварительно настроен (CZINI). Ожидание ответа до 100 mS.<br>Если **i** равно 1 то переменной **A** присваивается значение влажности в процентах , с точностью 0,1%<br>Если **i** равно 2 то переменной **A** присваивается значение температуры в градусах цельсия, с точностью 0,1 градус<br>Если **i** равно 3 то переменной **A** присваивается значение CO2 в PPM<br><br>В случае недоступности датчика, переменной **A** присваивается значение -32768 |

Примечание.
1. Перед работой с датчиком порт UART должен быть настроен на скорость 9600 8N1
2. Обратите внимание, что все настройки датчика COZIR сохраняются в его энергонезависимой памяти, поэтому только датчик может быть сконфигурирован один раз. Он не должен быть настроен каждый раз, когда он включен.

## Пример

```
LIST
0010 PAUSE 1000:UART 1,4:CZINI
0011 REM PAUSE 100:CZCMD "X 400":PAUSE 100
0015 FOR i=1 TO 50
0020 PRINT "----------------------------"
0021 A=CZRD{1}
0022 PRINT .1,"Влажность   = ",A," %"
0023 A=CZRD{2}
0024 PRINT .1,"Температура = ",A," град.C"
0025 A=CZRD{3}
0026 PRINT .0,"Сод. CO2   = ",A," ppm"
0029 PAUSE 1000
0030 NEXT i

OK
RUN
----------------------------
Влажность   = 53.9 %
Температура = 26.8 град.C
Сод. CO2   = 528 ppm
----------------------------
Влажность   = 54.2 %
Температура = 26.8 град.C
Сод. CO2   = 521 ppm
----------------------------
Влажность   = 54.2 %
Температура = 26.8 град.C
Сод. CO2   = 519 ppm
----------------------------
Влажность   = 54.3 %
Температура = 26.8 град.C
Сод. CO2   = 525 ppm
----------------------------
Влажность   = 54.3 %
Температура = 26.8 град.C
Сод. CO2   = 530 ppm
----------------------------
Влажность   = 54.3 %
Температура = 26.8 град.C
Сод. CO2   = 528 ppm
----------------------------
Влажность   = 54.2 %
Температура = 26.8 град.C
Сод. CO2   = 521 ppm
----------------------------
Влажность   = 54.2 %
Температура = 26.8 град.C
Сод. CO2   = 521 ppm
OK
```

Продолжение следует.  Схемотехника совершенствуется, набор команд и операторов увеличивается. Следите за новыми версиями.


*Замечания и предложения - gennadiy.v@gmail.com*
*Поддержка и актуальные версии http://picping.lg.ua/*



**Для коммерческого использования требуется  регистрация каждой копии П/О.*
**Исчезнет надпись DEMO и возможно появятся дополнительные возможности.*
**Вы можете заказать любые дополнения или купить проект целиком.*