Описание базовой прошивки для модулей / узлов на базе PIC18Fxxxx. Версия 4.47

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

Список добавлений и исправлений.

4.06.09 ver4.47 Дополнительно оптимизирована работа с мульти-мастерным каналом I2C.
Для отсутствия потерь пакетов, при ответе в режиме slave, пользователю рекомендуется не превышать время нахождения в обработчиках, связанных с Hi Int (вектора: HiInt; preHiInt; 1000Hz; 100Hz).
Общее время нахождения в этих обработчиках должно быть не более 25мкс, иначе может произойти потеря пакета. Если это случится, мастеру об этом будет сообщено отсутствием ACK и он сможет повторить запрос.

18.05.09 ver4.46 Значение 0xFF в REG_USER2 расценивается как несуществующее и происходит восстановление заводских установок. Это дополнительная страховка от случайного запрета восстановления.

13.05.09 ver4.45 Дополнительно оптимизирована работа с MicroLan.

11.05.09 ver4.44 Веден табличный расчет CRC8 и CRC16. Время расчета уменьшено более чем в 10 раз.

30.04.09 ver4.43 При работе в командами 3, 4, 16 MODBUS теперь используется адресация слов 16бит. Поэтому физический байтовый адрес в 2 раза меньше указанного словного.

29.06.08 ver4.40 Добавлена обработка встроенного теста с записью квитанции в EEPROM**.**
Изменен адрес ячейки EEPROM коррекции хода RTC с 0xF4 на 0xEF.

9.03.08 ver4.39 Добавлена поддержка стандартных команд MODBUS 0x03, 0x04, 0x10.
См. раздел «***Работа в качестве Slave устройства в сети MODBUS».***

9.01.08 ver4.37 Внесены изменения в раздел ***«Передача управления программам пользователя»*** пункты «При установке REG_USER<4>=1» и «При установке REG_USER2<6>=1».

30.11.07 ver4.36 Добавлен раздел «***Возврат к заводским установкам»***

12.07.07 ver4.36 Исправлена ошибка в описании раздела ***«Передача управления программам пользователя »*** - адрес **пользовательского вектора** в EEPROM ячейки **0xFA и 0xFB**.

18.03.07 ver4.34 Добавлены вектора пользователя, вызываемые при обработке чтения / записи через slave – канал I2C. Внесены уточнения в раздел ***«Передача управления программам пользователя »*** .

17.03.07 ver4.33 Введена возможность запрета сброса пользовательских векторов и восстановления заводских установок после 10 сбросов в течении 1 минуты после подачи питания : внесены уточнения в раздел ***«Передача управления программам пользователя »*** .

7.03.07 ver4.32 Упрощена работа с MODBUS в режиме мастера : внесены уточнения в раздел ***«Работа с UART (RS232, RS485, RS422, USB)»,*** подраздел «***Работа контроллера в качестве мастера в сети MODBUS***» .
Внесены изменения в раздел ***«Распределение ресурсов микроконтроллера»,*** подраздел «***RAM память***» .

30.01.07 ver4.31 Упрощена работа с MODBUS в режиме мастера : внесены уточнения в раздел ***«Работа с UART (RS232, RS485, RS422, USB)»,*** подраздел «***Работа контроллера в качестве мастера в сети MODBUS***» .
Внесены изменения в раздел ***«Распределение ресурсов микроконтроллера»,*** подраздел «***RAM память***» .
Добавлен подраздел «***Часы реального времени***».

24.09.06 ver4.22 Внесены уточнения в раздел ***«Распределение ресурсов микроконтроллера»*** подраздел «***FLASH память*** …таблица вызовов стандартных подпрограмм…» .

22.08.06 ver4.21 Внесены уточнения в раздел ***«Передача управления программам пользователя»*** подразделы «При установке REG_USER_2<7>=1 …» и «При установке REG_USER_2<6>=1…» .

04.08.06 ver4.20 Внесены уточнения в раздел ***«Использование программой пользователя встроенных подпрограмм для работы в качестве MASTER I2C»***.
Снято ограничение на одновременную работу встроенных подпрограмм I2C_RD / I2C_WR с работой MODBUS. Разнесены 3 рабочие ячейки для мастера I2C с MODBUS-овскими.
При вызове стандартных подпрограмм I2C_RD / I2C_WR не проверяется количество байт в пакете, что позволило удлинить пакет до 256 байт не считая SL_AD_I2C и WD_AD_I2C.

19.04.06 ver4.17 Внесены уточнения в раздел ***«Работа с UART (RS232, RS485, RS422, USB) // Работа контроллера в качестве мастера в сети MODBUS»***

Основные принципы работы

Базовая прошивка для модулей и узлов на основе PIC18Fxxxx создана , как универсальная платформа, позволяющая получить полный доступ по последовательным каналам ко всем ресурсам микроконтроллеров PIC18Fxxxx и независимо и прозрачно от этого загружать и выполнять программы пользователя.
В зависимости от типа, модуль/узел может иметь один или несколько последовательных интерфейсов.
Доступ к ресурсам микроконтроллера поддерживается через UART (RS232 / RS485 / RS422 / USB) и I2C.
Также прошивка поддерживает последовательные интерфейсы SPI и MicroLan в режиме мастера.
Прошивка обеспечивает транзитный доступ через UART (-ы) ко всем доступным другим интерфейсам и следовательно ко всем узлам/модулям к ним подключенным. Наличие одновременно нескольких интерфейсов дает возможность создавать древовидную сетевую структуру из многих узлов/модулей и многих уровней.
В случае, когда узел/модуль имеет два UART-а, обеспечивается симметричный взаимный доступ одного к другому. При работе с UART используется протокол MODBUS.
Кроме работы с интерфейсами, прошивка обеспечивает взаимодействие системы с программой пользователя. Имеется набор пользовательских векторов, которые позволяют передавать управление к соответствующим обработчикам событий: сброс, прерывания высокое и низкое, время, получение по MODBUS неизвестной команды. Также имеется набор подпрограмм, позволяющих пользователю из своей программы обращаться к имеющимся интерфейсам для доступа в другие узлы.
Из вышесказанного видно, что есть три основные возможности использования узла/модуля с описываемой прошивкой.
Первый путь - это использование одного из интерфейсов для полного доступа к ресурсам микроконтроллера. Пользователь может напрямую обращаться к портам ввода-вывода и узлам микроконтроллера(АЦП, ШИМ, таймерам, компараторам и т.д.). При этом, он вначале настраивает режимы нужных узлов, а потом просто опрашивает их состояние. Этот путь, также, обеспечивает и просто транзитный режим, для полного доступа к ресурсам узлов/модулей подключенных к смежным интерфейсам транзитного узла.
Второй путь - это использование прошивки для загрузки программы пользователя, для последующего ее монопольного выполнения. При этом прошивка фактически является программатором, а пользовательская программа полностью управляет узлом/модулем.
Третий путь - это комбинация доступа ко всем ресурсам через последовательный интерфейс с работой загруженной программы пользователя. При этом загруженная программа полностью управляет работой узла/модуля, а через интерфейсы прозрачно обеспечивается доставка параметров / данных и доступ в другие узлы/модули.

Для работы с базовой прошивкой, разработаны оболочка и DLL работающие под Windows .
Оболочка обеспечивает полный доступ к ресурсам узлов / модулей подключенных через UART и узлов / модулей подключенных к предыдущим через смежные интерфейсы.
Также она позволяет легко заносить программы пользователя во FLASH память любого из подключенных узлов / модулей , включая модули подключенные через смежные интерфейсы.
DLL позволяет создавать собственные программы под Windows для взаимодействия с объектами управления/сбора данных.

Эти программы доступны на сайте www.fractal.com.ru и CD-Fractal .

Этот документ отражает свойства прошивки, и не содержит сведений о специфических свойствах конкретного прибора и ресурсах используемого микроконтроллера. При работе с конкретным прибором, пользователю могут понадобиться описания на соответствующий прибор и используемый микроконтроллер .

Эти документы доступны на CD-Fractal .

Работа с UART (RS232, RS485, RS422, USB).

Если в модуле/узле предусмотрен хотя бы один из интерфейсов :RS232, RS485, RS422, USB , которые подключены ко встроенному UART , то прошивка поддерживает работу с этими интерфейсами по протоколу MODBUS и в качестве мастера и в качестве slave-узла.
При работе качестве slave-узла MODBUS, поддерживается обработка 17 команд, обеспечивающих полный доступ ко всем внутренним ресурсам, включая запись/верификацию программной FLASH памяти, доступ в другие интерфейсные каналы и т.д. При приеме и отправке пакетов автоматически производится подсчет и проверка CRC16.
*Команды, приходящие по протоколу **MODBUS**, для работы с внутренними ресурсами, используют **прямую** адресацию ресурсов микроконтроллера*.
В модуле/узле может быть два независимых UART. В этом случае оба канала полностью независимы и равноправны. Т.е. через них может одновременно и прозрачно осуществляться доступ к одним и тем же или к разным ресурсам модуля / узла. Так же прошивка обеспечивает взаимный и симметричный доступ от одного канала к другому.
Для работы с каждым каналом, в памяти микроконтроллера отводится отдельный буфер приема/передачи и комплект управляющих регистров:

- для UART1 буфер => RAM(0x500…0x5FF), REG_MODBUS1 => RAM(0x053);
- для UART2 буфер => RAM(0x600…0x6FF), REG_MODBUS2 => RAM(0x073).

Режимами работы каждого канала управляют соответствующие регистры REG_MODBUSx:

REG_MODBUSx<7>=1 идет прием телеграммы,
REG_MODBUSx<6>=1 идет передача телеграммы,
REG_MODBUSx<5>=1 телеграмма принята-готова к обработке
REG_MODBUSx<4>=1 контроллер готов к приему новой телеграммы
REG_MODBUSx<3>=1 добавить к телеграмме CRC16,предать и получить ответ, проверить CRC16
REG_MODBUSx<2>=1 совпадение циркулярного адреса
REG_MODBUSx<1>=1 совпадение ADR_MODBUS
REG_MODBUSx<0>=1 совпадение контрольной суммы (CRC).

В соответствии с протоколом MODBUS, обмен информацией между мастером и slave производится пакетами только по инициативе мастера. Т.е. slave только отвечает мастеру на его запросы. За пакет принимается любая последовательная передача данных, следующая после паузы не менее 3.5 байт на текущей скорости обмена, с паузами между байтами не более 1.5 байт, и до ближайшей паузы более 3.5 байт.
Пакеты длиннее 256 байт игнорируются. Причем числом 256 ограничена общая длинна пакета. Поскольку в каждый отдельный момент времени контроллер работает либо на прием, либо на передачу, используется один буфер и для приема и для передачи.
После любого вида сброса контроллер настраивает UART (-ы) на скорости обмена и адреса узлов соответствующие значениям занесенным в ячейки EEPROM :

- для UART1 скорость обмена => EEPROM (0x0FC, 0x0FD), адрес => EEPROM (0x0FF);
- для UART2 скорость обмена => EEPROM (0x0F6, 0x0F7), адрес => EEPROM (0x0F9).

Считанные значения скорости загружаются в SPBRGx и могут быть оперативно изменены.
Адреса загружаются в RAM(0x052)<=UART1 и в RAM(0x072)<=UART2 и тоже могут быть изменены.

Работа в качестве Slave устройства в сети MODBUS.

Первый байт в пакете запроса мастера - всегда адрес slave-узла. Второй - команда. Далее, как правило, идут два байта адреса ресурса внутри контроллера - сначала старший, потом младший.

Адресация внутренних ресурсов при доступе по MODBUS прямая.

Следующий байт пакета в большинстве команд это число читаемых/записываемых байт данных.

Обратите внимание – адреса для различных интерфейсов (если в приборе имеется, к примеру, и UART и I2C) **индивидуальны** и хранятся в разных ячейках и поэтому могут, как совпадать, так и быть разными. Кроме своего индивидуального адреса обрабатываются команды с циркулярным адресом, равным 0. Это удобно для передачи мастером информации во все узлы одновременно.

Последние 2 байта пакета всегда интерпретируются как контрольная сумма CRC16. При несовпадении CRC16 запрос мастера игнорируется, ответный пакет не формируется. Так же ответ не формируется на запросы с циркулярным адресом т.е. при работе с циркулярным адресом имеют смысл только команды записи.

Если адрес узла совпал и CRC16 соответствует пакету, то производится разбор команды. В этом случае контроллер либо даст соответствующий запросу ответ, либо квитанцию с кодом ошибки. Ошибка выдается при приеме несуществующей команды(если не разрешена обработка телеграммы в пользовательской программе) или несоответствии параметров запроса команде. Например: в команде записи указано, что будет записываться 5 байт, а передано 6 байт данных.

В ответном пакете контроллер сохраняет структуру запроса. Первый байт ответа это свой адрес. Второй код команды. Третий и четвертый адрес ресурса. Пятый количество байт читаемых/записываемых данных или номер бита в битовых операциях. Последние 2 байта это CRC16 посчитанное контроллером для ответного пакета. При обнаружении несоответствия параметров конкретной команды или при несуществующей команде, контроллер формирует квитанцию об ошибке: первый байт - свой адрес, второй команда запроса с установленным в “1” старшим битом, третий - код ошибки, четвертый-пятый байты - CRC16.

Работа контроллера в качестве мастера в сети MODBUS.

Если микроконтроллеру не запрещена работа с конкретным каналом (REG_MODBUSx=0x00), то при отсутствии обменов по шине он всегда находится в готовности к приему телеграммы в режиме slave-узла (REG_MODBUSx=0x10).

Для формирования мастер-телеграммы необходимо:

- при необходимости, дождаться завершения текущих обменов -> REG_MODBUSx=0x10 ;
- при необходимости, приостановить работу с MODBUS , записав в REG_MODBUSx=0x00 ;
- разместить отправляемую телеграмму в соответствующем каналу буфере обмена MODBUS ;
- занести в регистр N_TXx <= число байт мастер-телеграммы MODBUS без CRC16 ;
- разрешить подсчет CRC16 и отправку мастер-телеграммы + CRC16, записав REG_MODBUSx=0x08 ;
- ждать ответную телеграмму, проверяя -> REG_MODBUSx
  =0x11(CRC=OK)
  =0x10(CRC=BAD)
  =0x18(нет ответной телеграммы)

Полученная телеграмма размещена в буфере соответствующем номеру канала.

В REG_MODBUSx <0> до следующего обмена хранится признак совпадения CRC16 (1=совпало).

В RAM(0x56) или в RAM(0x76) - число принятых байт включая CRC16 для каналов UART1 и UART2 соответственно.

Основной формат команды мастера для команд 0x70…0x7D

ADRMOD	COMAND	Adr_Hi	Adr_Lo	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

Команды поддерживаемые прошивкой.

03h **Read Holding Registers** (чтение регистров типа 4)
04h **Read Input Registers** (чтение регистров типа 3)
10h **Preset Multiple Registers** (запись регистров типа 4)
70h команда чтение RAM (оперативной памяти контроллера)
71h команда запись RAM (оперативной памяти контроллера)
72h команда чтение бита RAM (оперативной памяти контроллера)
73h команда запись бита RAM (оперативной памяти контроллера)
74h команда чтение EEPROM (энергонезависимой памяти данных)
75h команда запись EEPROM (энергонезависимой памяти данных)
76h команда чтение FLASH (памяти программ)
77h команда запись FLASH (памяти программ)
78h команда чтение идентификатора контроллера
79h команда инициализации контроллера
7Ah команда чтение шины I2C
7Bh команда запись шины I2C
7Ch команда ОБМЕН по шине SPI
7Dh команда ОБМЕН со смежным каналом UART

03h -> Команда Read Holding Registers

ADRMOD	03h	Adr_Hi	Adr_Lo	0	N*reg	CRC_Lo	CRC_Hi

ответ

ADRMOD	03h	N*2	data 1 Hi	data 1 Lo	…	data N Hi	data N Lo	CRC_Lo	CRC_Hi

04h -> Команда Read Input Registers

ADRMOD	04h	Adr_Hi	Adr_Lo	0	N*reg	CRC_Lo	CRC_Hi

ответ

ADRMOD	04h	N*2	data 1 Hi	data 1 Lo	…	data N Hi	data N Lo	CRC_Lo	CRC_Hi

10h -> Команда Preset Multiple Registers

ADRMOD	10h	Adr_Hi	Adr_Lo	0	N*reg	N*2	data1 Hi	data1 Lo	…	dataN Hi	dataN Lo	CRC_Lo	CRC_Hi

ответ

ADRMOD	10h	Adr_Hi	Adr_Lo	0	N*reg	CRC_Lo	CRC_Hi

70h -> Команда ЧТЕНИЕ RAM

ADRMOD	70h	Adr_Hi	Adr_Lo	N	CRC_Lo	CRC_Hi

ответ

ADRMOD	70h	Adr_Hi	Adr_Lo	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

71h -> Команда ЗАПИСЬ RAM

ADRMOD	71h	Adr_Hi	Adr_Lo	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

ответ

ADRMOD	71h	Adr_Hi	Adr_Lo	N	CRC_Lo	CRC_Hi

72h -> Команда ЧТЕНИЕ бита RAM

ADRMOD	72h	Adr_Hi	Adr_Lo	N_Bit	CRC_Lo	CRC_Hi

ответ

ADRMOD	72h	Adr_Hi	Adr_Lo	N_Bit	0 / 0FFh	CRC_Lo	CRC_Hi

73h -> Команда ЗАПИСЬ бита RAM

ADRMOD	73h	Adr_Hi	Adr_Lo	N_Bit	0/1…0FFh	CRC_Lo	CRC_Hi

ответ

ADRMOD	73h	Adr_Hi	Adr_Lo	N_Bit	CRC_Lo	CRC_Hi

74h -> Команда ЧТЕНИЕ EEPROM

ADRMOD	74h	Adr_Hi	Adr_Lo	N	CRC_Lo	CRC_Hi

ответ

ADRMOD	74h	Adr_Hi	Adr_Lo	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

75h -> Команда ЗАПИСЬ EEPROM

ADRMOD	75h	Adr_Hi	Adr_Lo	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

ответ

ADRMOD	75h	Adr_Hi	Adr_Lo	N	CRC_Lo	CRC_Hi

76h -> Команда ЧТЕНИЕ FLASH

ADRMOD	76h	Adr_Hi	Adr_Lo	N	CRC_Lo	CRC_Hi

ответ

ADRMOD	76h	Adr_Hi	Adr_Lo	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

77h -> Команда ЗАПИСЬ FLASH

ADRMOD	77h	Adr_Hi	Adr_Lo	40h	DATA_1	…	DATA_64	CRC_Lo	CRC_Hi

ответ

ADRMOD	77h	Adr_Hi	Adr_Lo	40h	CRC_Lo	CRC_Hi

78h -> Команда чтение идентификатора

ADRMOD	78h	CRC_Lo	CRC_Hi

ответ

ADRMOD	78h	Id_1	…	Id_252	CRC_Lo	CRC_Hi

79h -> Команда инициализация контроллера

ADRMOD	79h	55h	0AAh	CRC_Lo	CRC_Hi

7Ah -> Команда ЧТЕНИЕ ШИНЫ I2C

ADRMOD	7Ah	Slave_Adr	Word_Adr	N	CRC_Lo	CRC_Hi

ответ

ADRMOD	7Ah	Slave_Adr	Word_Adr	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

7Bh -> Команда ЗАПИСЬ ШИНЫ I2C

ADRMOD	7Bh	Slave_Adr	Word_Adr	N	DATA_1	…	DATA_N	CRC_Lo	CRC_Hi

ответ

ADRMOD	7Bh	Slave_Adr	Word_Adr	N	CRC_Lo	CRC_Hi

7Ch -> Команда ОБМЕН по шине SPI

ADRMOD	7Ch	CS	0	N	Dout_1	…	Dout_N	CRC_Lo	CRC_Hi

ответ

ADRMOD	7Ch	CS	0	N	Din_1	…	Din_N	CRC_Lo	CRC_Hi

7Dh -> Команда ОБМЕН со смежным каналом UART

ADRMOD	7Dh	Tx_1	…	Tx_N	CRC_Lo	CRC_Hi

ответ

Rx_1	…	Rx_M	CRC_Lo	CRC_Hi

Формат квитанции об ошибке:

Квитанция об ошибке

ADRMOD	Command+80h	N Error	CRC_Lo	CRC_Hi

КОДЫ ОШИБОК
0x01-несуществующий код команды
(!обратите внимание, если данный узел/модуль не поддерживает какой-либо ресурс, например не имеет шины I2C, то при соответствующем запросе будет выдана эта ошибка)
0x02-длинна телеграммы запроса не соответствует содержанию команды
0x03-запрос на 0 байт
0x04-запрошенное количество байт больше чем 249
0x05-номер бита >7
0x06-попытка записи в несуществующий адрес
0x07-попытка чтения закрытого ресурса
0x08-попытка записи во FLASH не 64 байта
0x09-попытка записи не с начала блока 64 байта
0x0A-попытка записи закрытой области FLASH
0x0B-ошибка верификации после записи FLASH
0x0C-ошибка при приеме обязательной последовательности 55h 0AAh
0x0D-при обращении к каналу I2C шина занята > 10мс
0x0E-коллизия при обмене по I2C
0x0F-нет сигнала подтверждения ACK от Slave-устройства I2C
0x10-смежный канал UART занят

****

03h **Read Holding Registers** (чтение регистров типа 4)
04h **Read Input Registers** (чтение регистров типа 3)

Длинна пакета запроса мастера всегда 8 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-03h/04h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5- 0
6-Число запрашиваемых регистров N
7-Младший байт CRC
8-Старший байт CRC

Эти команды реализованы для совместимости с оборудованием, поддерживающим команды протокола MODBUS.
Поскольку команда ориентирована на обмен с 16-битными регистрами, RAM представляется, как последовательность 16-битных ячеек. Причем младший байт регистровой пары, располагается по младшему адресу RAM. При этом адрес реальной ячейки RAM будет в 2 раза больше чем указан в команде.
Эта команда позволяет получить доступ и к регистрам специального назначения (SFR) PIC18Fxxxx для управления ядром микроконтроллера и его периферией. Адреса SFR-регистров в PIC18Fxxxx расположены с 0xF50…0xF80(в зависимости от типа микроконтроллера) по 0xFFF.
В команде допустимо задавать любой адрес RAM с 0x0000 по 0xFFFF.
При чтении реально не существующих ресурсов будет читаться 0.
Чтение массива данных гарантировано дает непрерывный массив – во время чтения блока данных все прерывания запрещены. Следствием этого может быть отставание системных часов, но только при непрерывном чтении массивов более 100 байт **и** на скорости 115200 и выше (в этом случае часы могут отстать не более 0.5% от времени непрерывного чтения). Одиночное чтение массивов до 249 байт, или работа с такими массивами на скоростях передачи меньше 115200 не оказывает на системные часы заметного влияния.
Количество запрашиваемых регистров может быть от 1 до 124.
При запросе другого количества будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.

ОТВЕТ контроллера на команду 03h/04h
Длинна ответной телеграммы = число запрашиваемых байт(шестой байт запроса *2) + 5 .
!Обратите внимание, что при чтении первым выдается старший байт регистровой пары, следующим младший.

1-Адрес узла
2-03h/04h
3-число передаваемых байт данных
4-DATA_RD{ADR_Hi ADR_Lo + 1}
5-DATA_RD{ADR_Hi ADR_Lo }
6-DATA_RD{ADR_Hi ADR_Lo + 3}
7-DATA_RD{ADR_Hi ADR_Lo + 2}
.....
N*2+2 -DATA_RD{ADR_Hi ADR_Lo + (N*2-1)}
N*2+3 -DATA_RD{ADR_Hi ADR_Lo + (N*2-2)}
N*2+4 -Младший байт CRC
N*2+5 -Старший байт CRC

ОТВЕТ контроллера на команду 03h/04h ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-83h/84h (команда со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

10h **Preset Multiple Registers** (запись регистров типа 4)

Длинна пакета запроса мастера = число записываемых байт(седьмой байт запроса) + 9 .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-10h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5- 0
6-Число запрашиваемых регистров N
7-Число запрашиваемых байт
8-DATA_WR{ADR_Hi ADR_Lo + 1}
9-DATA_ WR {ADR_Hi ADR_Lo }
10-DATA_ WR {ADR_Hi ADR_Lo + 3}
11-DATA_ WR {ADR_Hi ADR_Lo + 2}
.....
N*2+6 -DATA_ WR {ADR_Hi ADR_Lo + (N*2-1)}
N*2+7 -DATA_ WR {ADR_Hi ADR_Lo + (N*2-2)}
N*2+8 -Младший байт CRC
N*2+9 -Старший байт CRC

Эти команды реализованы для совместимости с оборудованием, поддерживающим команды протокола MODBUS.
Поскольку команда ориентирована на обмен с 16-битными регистрами, RAM представляется, как последовательность 16-битных ячеек. Причем младший байт регистровой пары, располагается по младшему адресу RAM.
При этом адрес реальной ячейки RAM будет в 2 раза больше чем указан в команде.
Эта команда позволяет получить доступ и к регистрам специального назначения (SFR) PIC18Fxxxx
для управления ядром микроконтроллера и его периферией. Адреса SFR-регистров в PIC18Fxxxx
расположены с 0xF50…0xF80(в зависимости от типа микроконтроллера) по 0xFFF.
В команде допустимо задавать любой адрес RAM с 0x0000 по 0xFFFF.
При записи реально не существующих ресурсов записываемые данные никуда не попадут.
Слитная запись массива данных гарантирована – во время записи блока данных все прерывания запрещены. Следствием этого может быть отставание системных часов, но только при непрерывной записи массивов более 100 байт **и** на скорости 115200 и выше (в этом случае часы могут отстать не более 0.5% от времени непрерывного чтения).
Одиночная запись массивов до 249 байт, или работа с такими массивами на скоростях передачи меньше 115200 не оказывает на системные часы заметного влияния.
Количество записываемых регистров может быть от 1 до 124.
При записи другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера приемного буфера равного 256 байт.

ОТВЕТ контроллера на команду 10h
Длинна квитанции = 8 байт.

1-Адрес узла
2-10h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5- 0
6-Число записанных байт (N)
7-Младший байт CRC
8-Старший байт CRC

ОТВЕТ контроллера на команду 10h запись RAM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна ответной телеграммы = 5 байт.

1-Адрес узла
2-90h (команда 10h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

70h команда чтение RAM

Длинна пакета запроса мастера всегда 7 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-70h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Число запрашиваемых байт (N)
6-Младший байт CRC
7-Старший байт CRC

Адрес RAM интерпретируется как реальный адрес RAM внутри PIC18Fxxxx.
Эта команда позволяет получить доступ и к регистрам специального назначения (SFR) PIC18Fxxxx для управления ядром микроконтроллера и его периферией. Адреса SFR-регистров в PIC18Fxxxx расположены с 0xF50…0xF80(в зависимости от типа микроконтроллера) по 0xFFF.
В команде допустимо задавать любой адрес RAM с 0x0000 по 0xFFFF.
При чтении реально не существующих ресурсов будет читаться 0.
Чтение массива данных гарантировано дает непрерывный массив – во время чтения блока данных все прерывания запрещены. Следствием этого может быть отставание системных часов, но только при непрерывном чтении массивов более 100 байт **и** на скорости 115200 и выше (в этом случае часы могут отстать не более 0.5% от времени непрерывного чтения). Одиночное чтение массивов до 249 байт, или работа с такими массивами на скоростях передачи меньше 115200 не оказывает на системные часы заметного влияния.
Количество запрашиваемых байт может быть от 1 до 249.
При запросе другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.

ОТВЕТ контроллера на команду 70h чтение RAM
Длинна ответной телеграммы = число запрашиваемых байт(пятый байт запроса) + 7 .

1-Адрес узла
2-70h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Число запрашиваемых байт (N)
6-DATA_RD{ADR_Hi ADR_Lo}
7-DATA_RD{ADR_Hi ADR_Lo + 1}
.....
N+5-DATA_RD{ADR_Hi ADR_Lo + (N-1)}
N+6-Младший байт CRC
N+7-Старший байт CRC

ОТВЕТ контроллера на команду 70h чтение RAM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-F0h (команда 70h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

71h команда запись RAM

Длинна пакета запроса мастера = число записываемых байт(пятый байт запроса) + 7 .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-71h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Число записываемых байт (N)
6-DATA_WR{ADR_Hi ADR_Lo}
7-DATA_WR{ADR_Hi ADR_Lo + 1}
.....
N+5-DATA_WR{ADR_Hi ADR_Lo + (N-1)}
N+6-Младший байт CRC
N+7-Старший байт CRC

Адрес RAM интерпретируется как реальный адрес RAM внутри PIC18Fxxxx.
Эта команда позволяет получить доступ и к регистрам специального назначения (SFR) PIC18Fxxxx для управления ядром микроконтроллера и его периферией. Адреса SFR-регистров в PIC18Fxxxx расположены с 0xF50…0xF80(в зависимости от типа микроконтроллера) по 0xFFF.
В команде допустимо задавать любой адрес RAM с 0x0000 по 0xFFFF.
При записи реально не существующих ресурсов записываемые данные никуда не попадут.
Слитная запись массива данных гарантирована – во время записи блока данных все прерывания запрещены. Следствием этого может быть отставание системных часов, но только при непрерывной записи массивов более 100 байт **и** на скорости 115200 и выше (в этом случае часы могут отстать не более 0.5% от времени непрерывного чтения).
Одиночная запись массивов до 249 байт, или работа с такими массивами на скоростях передачи меньше 115200 не оказывает на системные часы заметного влияния.
Количество записываемых байт может быть от 1 до 249.
При записи другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера приемного буфера равного 256 байт.

ОТВЕТ контроллера на команду 71h запись RAM
Длинна квитанции = 7 байт.

1-Адрес узла
2-71h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Число записанных байт (N)
6-Младший байт CRC
7-Старший байт CRC

ОТВЕТ контроллера на команду 71h запись RAM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна ответной телеграммы = 5 байт.

1-Адрес узла
2-F1h (команда 71h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

72h команда чтение бита RAM

Длинна пакета запроса мастера всегда 7 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-72h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Номер бита в байте (N)
6-Младший байт CRC
7-Старший байт CRC

Адрес байта RAM интерпретируется как реальный адрес RAM внутри PIC18Fxxxx.
Эта команда позволяет получить битовый доступ и к регистрам специального назначения (SFR) PIC18Fxxxx для управления ядром микроконтроллера и его периферией. Адреса SFR-регистров в PIC18Fxxxx расположены с 0xF50…0xF80(в зависимости от типа микроконтроллера) по 0xFFF.
В команде допустимо задавать любой адрес RAM с 0x0000 по 0xFFFF.
При чтении реально не существующих ресурсов будет читаться 0.
Номер бита считается с 0 и до 7.
При запросе чтения бита с номером >7 будет отправлена телеграмма с кодом ошибки.
Если указанный в запросе бит =0, то в ответной телеграмме шестой байт будет равен 0.
Если указанный в запросе бит =1, то в ответной телеграмме шестой байт будет равен 0FFh.

ОТВЕТ контроллера на команду 72h чтение бита RAM
Длинна ответной телеграммы = 8 байт .

1-Адрес узла
2-72h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Номер бита в байте (N)
6-00 / FFh
7-Младший байт CRC
8-Старший байт CRC

ОТВЕТ контроллера на команду 72h чтение бита RAM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-F2h (команда 72h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

73h команда запись бита RAM

Длинна пакета запроса мастера всегда 8 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-72h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Номер бита в байте (N)
6-00 / 01...FFh
7-Младший байт CRC
8-Старший байт CRC

Адрес байта RAM интерпретируется как реальный адрес RAM внутри PIC18Fxxxx.
Эта команда позволяет получить битовый доступ и к регистрам специального назначения (SFR) PIC18Fxxxx для управления ядром микроконтроллера и его периферией. Адреса SFR-регистров в PIC18Fxxxx расположены с 0xF50…0xF80(в зависимости от типа микроконтроллера) по 0xFFF.
В команде допустимо задавать любой адрес RAM с 0x0000 по 0xFFFF.
Команда особенно удобна при работе с SFR регистрами, т.к. во многих случаях,
при работе с ними не допустимы операции чтение-модификация запись.
При чтении реально не существующих ресурсов будет читаться 0.
Номер бита считается с 0 и до 7.
При записи бита с номером >7 будет отправлена телеграмма с кодом ошибки.
Если 6-й байт запроса =0, то в соответствующий бит адресуемого байта будет занесен 0.
Если 6-й байт запроса >0, то в соответствующий бит адресуемого байта будет занесена 1.

ОТВЕТ контроллера на команду 73h запись бита RAM
Длинна ответной квитанции = 7 байт .

1-Адрес узла
2-73h
3-Старший байт адреса RAM контроллера (ADR_Hi)
4-Младший байт адреса RAM контроллера (ADR_Lo)
5-Номер бита в байте (N)
6-Младший байт CRC
7-Старший байт CRC

ОТВЕТ контроллера на команду 73h запись бита RAM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-F3h (команда 73h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

74h команда чтение ячеек EEPROM

Длинна пакета запроса мастера всегда 7 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-74h
3-Старший байт адреса EEPROM контроллера (ADR_Hi)
4-Младший байт адреса EEPROM контроллера (ADR_Lo)
5-Число запрашиваемых байт (N)
6-Младший байт CRC
7-Старший байт CRC

Адрес EEPROM интерпретируется как реальный адрес EEPROM внутри PIC18Fxxxx.
В команде допустимо задавать любой адрес EEPROM с 0000h по 0FFFFh.
В зависимости от типа микроконтроллера, реально адресуется от 256 до 1024 байт EEPROM .
При чтении реально не существующих ресурсов будет повторяться смежная область реально существующей EEPROM .
Количество запрашиваемых байт может быть от 1 до 249.
При запросе другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.

ОТВЕТ контроллера на команду 74h чтение EEPROM
Длинна ответной телеграммы = число запрашиваемых байт(пятый байт запроса) + 7 .

1-Адрес узла
2-74h
3-Старший байт адреса EEPROM контроллера (ADR_Hi)
4-Младший байт адреса EEPROM контроллера (ADR_Lo)
5-Число запрашиваемых байт (N)
6-DATA_RD{ADR_Hi ADR_Lo}
7-DATA_RD{ADR_Hi ADR_Lo + 1}
.....
N+5-DATA_RD{ADR_Hi ADR_Lo + (N-1)}
N+6-Младший байт CRC
N+7-Старший байт CRC

ОТВЕТ контроллера на команду 74h чтение EEPROM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-F4h (команда 74h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

75h команда запись EEPROM

Длинна пакета запроса мастера = число записываемых байт(пятый байт запроса) + 7 .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-75h
3- 0
4-Младший байт адреса EEPROM контроллера (ADR_Lo)
5-Число записываемых байт (N)
6-DATA_WR{ADR_Hi ADR_Lo}
7-DATA_WR{ADR_Hi ADR_Lo + 1}
.....
N+5-DATA_WR{ADR_Hi ADR_Lo + (N-1)}
N+6-Младший байт CRC
N+7-Старший байт CRC

Адрес EEPROM интерпретируется как реальный адрес EEPROM внутри PIC18Fxxxx.
В команде допустимо задавать адрес EEPROM с 000h по 3FFh.
В зависимости от типа микроконтроллера, реально адресуется от 256 до 1024 байт EEPROM .
Количество записываемых байт может быть от 1 до 249.
При записи другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера приемного буфера равного 256 байт.

ОТВЕТ контроллера на команду 75h запись EEPROM
Длинна квитанции = 7 байт.

1-Адрес узла
2-75h
3- 0
4-Младший байт адреса EEPROM контроллера (ADR_Lo)
5-Число записанных байт (N)
6-Младший байт CRC
7-Старший байт CRC

Квитанция формируется контроллером после окончания процесса записи EEPROM.

ОТВЕТ контроллера на команду 75h запись EEPROM ПРИ ОШИБКЕ (несоответствии параметров)
Длинна ответной телеграммы = 5 байт.

1-Адрес узла
2-F5h (команда 75h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

76h команда чтение FLASH-памяти программ

Длинна пакета запроса мастера всегда 7 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-76h
3-Старший байт адреса FLASH контроллера (ADR_Hi)
4-Младший байт адреса FLASH контроллера (ADR_Lo)
5-Число запрашиваемых байт (N)
6-Младший байт CRC
7-Старший байт CRC

Адрес FLASH интерпретируется как реальный адрес FLASH внутри PIC18Fxxxx.
В команде допустимо задавать любой адрес FLASH с 0000h по 0FFFFh.
В зависимости от типа микроконтроллера, реально адресуется от 16 до 64 Кбайт FLASH.
При чтении реально не существующих ресурсов будет читаться 0FFh .
Доступ к адресам с 0 по 1FFFh закрыт, там находится основная программа. При чтении этих адресов будет выдаваться квитанция об ошибке.
Количество запрашиваемых байт может быть от 1 до 249.
При запросе другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.

ОТВЕТ контроллера на команду 76h чтение FLASH
Длинна ответной телеграммы = число запрашиваемых байт(пятый байт запроса) + 7 .

1-Адрес узла
2-76h
3-Старший байт адреса FLASH контроллера (ADR_Hi)
4-Младший байт адреса FLASH контроллера (ADR_Lo)
5-Число запрашиваемых байт (N)
6-DATA_RD{ADR_Hi ADR_Lo}
7-DATA_RD{ADR_Hi ADR_Lo + 1}
.....
N+5-DATA_RD{ADR_Hi ADR_Lo + (N-1)}
N+6-Младший байт CRC
N+7-Старший байт CRC

ОТВЕТ контроллера на команду 76h чтение FLASH ПРИ ОШИБКЕ (несоответствии параметров)
Динна квитанции = 5 байт.

1-Адрес узла
2-F6h (команда 76h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

77h команда запись FLASH-памяти программ

Длинна пакета запроса мастера = 71 байт .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-77h
3-Старший байт адреса FLASH контроллера (ADR_Hi)
4-Младший байт адреса FLASH контроллера (ADR_Lo)
5-40h
6-DATA_WR{ADR_Hi ADR_Lo}
7-DATA_WR{ADR_Hi ADR_Lo + 1}
.....
69-DATA_WR{ADR_Hi ADR_Lo + 3Fh)}
70-Младший байт CRC
71-Старший байт CRC

Адрес FLASH интерпретируется как реальный адрес FLASH внутри PIC18Fxxxx.
В команде допустимо задавать любой адрес FLASH с 0000h по 0FFFFh.
В зависимости от типа микроконтроллера, реально адресуется от 16 до 64 Кбайт FLASH.
Доступ к адресам с 0 по 1FFFh закрыт, там находится основная программа.
При попытке записи в эти адреса будет выдаваться квитанция об ошибке.
Запись ведется только блоками по 64 байта.
При попытке записи другого количества байт будет отправлена телеграмма с кодом ошибки.
Предварительно перед записью производится стирание блока.
Указываемый адрес должен быть кратен 64, т.е. младшие 6 бит ADR_Lo должны равняться 0.
В противном случае будет выдана квитанция об ошибке.
Ответная телеграмма выдается после завершения процесса записи и верификации.
Если верификация дала ошибку, то будет выдана квитанция об ошибке.

ОТВЕТ контроллера на команду 77h запись FLASH
Длинна квитанции = 7 байт.

1-Адрес узла
2-77h
3-Старший байт адреса FLASH контроллера (ADR_Hi)
4-Младший байт адреса FLASH контроллера (ADR_Lo)
5-40h
6-Младший байт CRC
7-Старший байт CRC

Квитанция формируется контроллером после завершения процесса записи и верификации FLASH.

ОТВЕТ контроллера на команду 77h запись FLASH ПРИ ОШИБКЕ (несоответствии параметров, ошибке верификации)
Длинна ответной телеграммы = 5 байт.

1-Адрес узла
2-F7h (команда 77h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

78h команда чтение идентификатора контроллера

Длинна пакета запроса мастера = 4 байта .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-78h
3-Младший байт CRC
4-Старший байт CRC

При подаче этой команды контроллер выдает 252 байта информации о себе.
Эта информация записывается в процессе производства и не может быть изменена пользователем.
Идентификатор содержит информацию о типе устройства, его конкретном исполнении, версии основной программы, дате выпуска, серийный номер, а также дополнительную информацию.

ОТВЕТ контроллера на команду 78h идентификатора контроллера
Длинна ответной телеграммы = 256 байт.

1-Адрес узла
2-78h
3-Id_0
4-Id_1
.....
254-Id_0FBh
255-Младший байт CRC
256-Старший байт CRC

ОТВЕТ контроллера на команду 78h чтение FLASH ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-F8h (команда 78h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

79h команда инициализация контроллера

Длинна пакета запроса мастера = 6 байт .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-79h
3-55h
4-0AAh
5-Младший байт CRC
6-Старший байт CRC

При подаче этой команды и в случае совпадения контрольных байт 55h, 0AAh и CRC контроллер выполняет команду RESET. Другими словами выполняется «горячий» перезапуск контроллера.

ОТВЕТ контроллера на команду 79h не формируется.

ОТВЕТ контроллера на команду 79h инициализация контроллера ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-F9h (команда 79h со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

7Ah Команда ЧТЕНИЕ ШИНЫ I2C

Длинна пакета запроса мастера всегда 7 байт.
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-7Ah
3-Адрес модуля на шине I2C (Slave_Adr)
4-Адрес ресурса в модуле I2C (Word_Adr)
5-Число запрашиваемых байт (N)
6-Младший байт CRC
7-Старший байт CRC

Эта команда позволяет получить доступ и к другим контроллерам через локальную шину I2C и фактически подключить их ресурсы к сети MODBUS.
После получения этой команды контроллер, выступая в качестве мастера шины I2C, инициирует обмен с slave-модулем на шине I2C имеющим адрес = Slave_Adr. Полученные в результате обмена от slave-модуля N байт данных начиная с внутреннего адреса ресурса Word_Adr , контроллер отправляет в ответной телеграмме.
В команде допустимо задавать любой адрес Slave_Adr с 00h по 0FFh. Но т.к. младший бит Slave_Adr интерпретируется устройствами на шине I2C как признак записи / чтения, контроллер игнорирует младший бит Slave_Adr. Для контроллера , например, адреса 54h и 55h это адрес одного и того же модуля на шине I2C.
При выходе на шину I2C контроллер соблюдает правила работы с мультимастерной шиной, проверяя, не нарушает ли он обмен по шине другого мастера. Если шина занята, то контроллер ждет освобождения шины 10 мс и если шина продолжает быть занятой то отправляется квитанция с кодом ошибки 0Dh.
Для осуществления чтения данных из другого модуля формируется пакет обмена по шине I2C состоящий из двух частей. Первая часть пакета это запись в slave-модуль адреса ресурса внутри модуля Word_Adr. Это два байта – Slave_Adr и Word_Adr. Вторая часть пакета это собственно чтение . Оно состоит из N+1 байт. Это снова Slave_Adr + N байт читаемых данных. В случае отсутствия на шине I2C устройства с адресом Slave_Adr или сбое в обмене вместо ответной телеграммы контроллер сформирует квитанцию об ошибке.
Количество запрашиваемых байт может быть от 1 до 249.
При запросе другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.

ОТВЕТ контроллера на команду 7Ah чтение шины I2C.
Длинна ответной телеграммы = число запрашиваемых байт(пятый байт запроса) + 7 .

1-Адрес узла
2-7Ah
3- Адрес модуля на шине I2C (Slave_Adr)
4- Адрес ресурса в модуле I2C (Word_Adr)
5-Число запрашиваемых байт (N)
6-DATA_RD{ Slave_Adr (Word_Adr) }
7-DATA_RD{{ Slave_Adr (Word_Adr+1) }
.....
N+5-DATA_RD{ Slave_Adr (Word_Adr+ N-1) }
N+6-Младший байт CRC
N+7-Старший байт CRC

ОТВЕТ контроллера на команду 7Ah чтение шины I2C ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-FAh (команда 7Ah со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

7Bh Команда ЗАПИСЬ ШИНЫ I2C

Длинна пакета запроса мастера = число записываемых байт(пятый байт запроса) + 7 .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-7Bh
3-Адрес модуля на шине I2C (Slave_Adr)
4-Адрес ресурса в модуле I2C (Word_Adr)
5-Число записываемых байт (N)
6- DATA_WR { Slave_Adr (Word_Adr) }
7- DATA_WR {{ Slave_Adr (Word_Adr+1) }
.....
N+5- DATA_WR { Slave_Adr (Word_Adr+ N-1) }
N+6-Младший байт CRC
N+7-Старший байт CRC

Эта команда позволяет получить доступ и к другим контроллерам через локальную шину I2C и фактически подключить их ресурсы к сети MODBUS.
После получения этой команды контроллер, выступая в качестве мастера шины I2C, инициирует обмен с slave-модулем на шине I2C имеющим адрес = Slave_Adr. В результате обмена в slave-модуль, начиная с внутреннего адреса ресурса Word_Adr, записываются N байт данных, пришедших в команде MODBUS .
В команде допустимо задавать любой адрес Slave_Adr с 00h по 0FFh. Но т.к. младший бит Slave_Adr интерпретируется устройствами на шине I2C как признак записи / чтения, контроллер игнорирует младший бит Slave_Adr. Для контроллера , например, адреса 54h и 55h это адрес одного и того же модуля на шине I2C.
При выходе на шину I2C контроллер соблюдает правила работы с мультимастерной шиной, проверяя не нарушает ли он обмен по шине другого мастера. Если шина занята, то контроллер ждет освобождения шины 10 мс и если шина продолжает быть занятой то отправляется квитанция с кодом ошибки 0Dh.
Для осуществления записи данных в другой модуль формируется один пакет обмена по шине I2C. Пакет состоит из N+2 байт. Это два байта – Slave_Adr и Word_Adr. + N байт записываемых данных. В случае отсутствия на шине I2C устройства с адресом Slave_Adr или сбое в обмене вместо ответной телеграммы контроллер сформирует квитанцию об ошибке.
Количество записываемых байт может быть от 1 до 249.
При записи другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.

ОТВЕТ контроллера на команду 7Bh запись шины I2C.
Длинна ответной квитанции = 7 байт.

1-Адрес узла
2-7Bh
3- Адрес модуля на шине I2C (Slave_Adr)
4- Адрес ресурса в модуле I2C (Word_Adr)
5-Число записываемых байт (N)
6-Младший байт CRC
7-Старший байт CRC

ОТВЕТ контроллера на команду 7Bh запись шины I2C ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-FBh (команда 7Bh со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

7Ch Команда ОБМЕН по шине SPI

Длинна пакета запроса мастера = число байт обмена(пятый байт запроса) + 7 .
При другой длине запроса будет отправлена квитанция с кодом ошибки.

1-Адрес узла
2-7Ch
3-Комбинация сигналов выборки устройства SPI (CS)
4-0
5-Число байт (N)
6- Dout_1_WR
7- Dout_2_WR
.....
N+5- Dout_ N _WR
N+6-Младший байт CRC
N+7-Старший байт CRC

Эта команда позволяет получить доступ и к модулям с интерфейсом SPI . Это, как правило, модули АЦП.
Если в узле/модуле уже присутствует интерфейс I2C, то аппаратный узел микроконтроллера уже занят и узел/модуль программно эмулирует SPI. При этом после получения команды три соответствующие линии SCK, Din, Dout автоматически настраиваются на нужное направление передачи и остаются в этом состоянии до изменения другими командами их состояния. При использовании аппаратного узла SPI , линии инициализированы сразу после сброса.
На линии RA0, RA1, RA2 выдаются соответственно 3 младших бита CS для адресной работы с модулями SPI.
Команда самостоятельно **не** изменяет режимы работы этих линий. Поэтому, если необходима адресная работа с модулями SPI , то необходимо заранее проинициализировать эти линии как выходы.
В противном случае они останутся входами и комбинация CS не будет выведена на них. Этот режим удобен при работе с единственным SPI –модулем (линии RA0, RA1, RA2 можно использовать для других целей).
Поскольку шина SPI дуплексная, команда обеспечивает одновременную передачу и прием информации по шине.
Т.е. при каждом импульсе по тактовой линии SCK одновременно записывается и считывается по одному биту.
Обратите внимание что если , к примеру, необходимо только чтение N байт, то Вы в команде все равно должны указать N записываемых байт и они будут синхронно выдаваться вместе с принимаемыми. В этом случае, если Вы пошлете байты 0 , то выход Dout при приеме будет постоянно равен 0.
Пример : Если Вам необходимо послать в модуль SPI один байт и потом в этом же цикле получить в ответ два байта, то в команде надо указать общую длину обмена по SPI N=3, первый байт записываемых данных – команда, второй и третий фиктивные =0, в ответ в поле данных получите 3 байта : первый фиктивный(он загружался синхронно с передаваемой командой), второй и третий – это то что Вам надо.

Количество записываемых байт может быть от 1 до 249.
При записи другого количества байт будет отправлена телеграмма с кодом ошибки.
Максимальное количество байт ограничено числом 249 из за размера передающего буфера равного 256 байт.
Учитывая такую возможную длину посылки, можно одной командой проводить несколько циклов обмена с одним устройством.

ОТВЕТ контроллера на команду 7Ch обмен по шине SPI.
Длинна ответной телеграммы = число байт обмена(пятый байт запроса) + 7 .

1-Адрес узла
2-7Ch
3-Комбинация сигналов выборки устройства SPI (CS)
4-0
5-Число байт (N)
6- Din_1_WR
7- Din_2_WR
.....
N+5- Din_ N _WR
N+6-Младший байт CRC
N+7-Старший байт CRC

ОТВЕТ контроллера на команду обмен по шине SPI ПРИ ОШИБКЕ (несоответствии параметров)
Длинна квитанции = 5 байт.

1-Адрес узла
2-FCh (команда 7Ch со взведенным старшим битом)
3-Код ошибки
4-Младший байт CRC
5-Старший байт CRC

! При несовпадении CRC в запросе - запрос мастера игнорируется , ответный пакет не формируется.

****

7Dh Команда ОБМЕН со смежным каналом UART

Длинна пакета запроса мастера = число байт пакета передаваемого в смежную шину + 2 .
Общая длинна пакета не должна быть больше 255 байт.

1- Адрес узла
2- 7Dh
3- Tx_1
4- Tx_2
.....
N+2- Tx_ N
N+3- Младший байт CRC16
N+4- Старший байт CRC16

Эта команда позволяет получить доступ и к смежному каналу UART. Команда будет работать только в модулях с двумя UART-ами, в других модулях при запросе с этой командой прейдет ответ с ошибкой «несуществующая команда». Команда 7Dh обеспечивает удобный доступ в подсети, при этом модуль, получивший команду, играет роль «умного репитера» - концентратора. Т.е. обращаясь к узлу с конкретным адресом, пользователь получает доступ к его подсети, которая может состоять из многих узлов, которые в свою очередь могут иметь свои собственные подсети и т.д.
Механизм работы команды:

- принятый пакет проверяется на совпадение CRC16 и адреса
- в случае совпадения, проверяется свободна ли смежная шина, если смежная шина занята, то возвращается квитанция с кодом ошибки 0x10
- если смежная шина свободна, то команда обрабатывается дальше
- отстригаются первые два байта(адрес и сама команда) и последние два байта(CRC16)
- ко вложению считается новая CRC16 и отправляется в смежную шину(фактически телеграмма стала короче на 2 байта)
- взводится признак ожидания ответа
- квитанция не передается
- ожидается ответ сколь угодно долго
- пришедший ответ проверяется на CRC16 и только(адрес и содержание пакета не проверяется!)
- в случае совпадения CRC16 пакет БЕЗ ИЗМЕНЕНИЙ передается в запросившую сеть
- после этого признак ожидания ответа сбрасывается
- любой повторный запрос из основной сети с правильными адресом и CRC16, пришедший до ответа подсети сбрасывает признак ожидания ответа, если это не новая команда 7Dh

Команда обладает рекурсивными свойствами, т.е. вложение может содержать такую же команду или команды, тогда следующие узлы ретранслируют вложение в свои подсети. При этом каждый раз вложение становится короче на два байта, а маршрут пакета определяется адресами указанными в командах 7Dh. Обратная телеграмма пойдет тем же маршрутом, но всякий раз будет передаваться без изменений и, конечно же, с проверкой CRC16.
Обратите внимание - ответный пакет , если таковой вообще имеется, не является квитанцией команды 7Dh.

Программа обрабатывает запросы с обоих UART-ов . Другими словами оба последовательных канала равноправны и симметричны. Т.е. от какой бы шины не пришел запрос с командой 7Dh, при совпадении адреса и CRC16, противоположная шина будет играть роль подсети.

ОТВЕТ контроллера на команду 7Dh обмен со смежным каналом UART в случае ее ответа.
Длинна ответной телеграммы = длине телеграммы полученной от смежного канала.

1- Rx_1
2- Rx_2
.....
N- Rx_ N
N+1- Младший байт CRC16
N+2- Старший байт CRC16

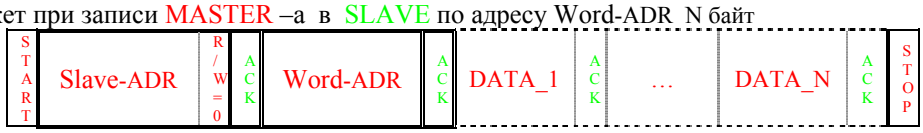
ОТВЕТ контроллера на команду 7Dh обмен со смежным каналом UART при занятом канале
Длинна квитанции = 5 байт.

1- Адрес узла
2- FDh (команда 7Dh со взведенным старшим битом)
3- 0x10
4- Младший байт CRC
5- Старший байт CRC

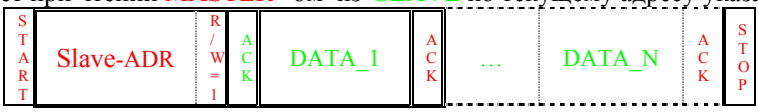
! При несовпадении CRC в запросе - запрос мастера игнорируется .

Работа с I2C

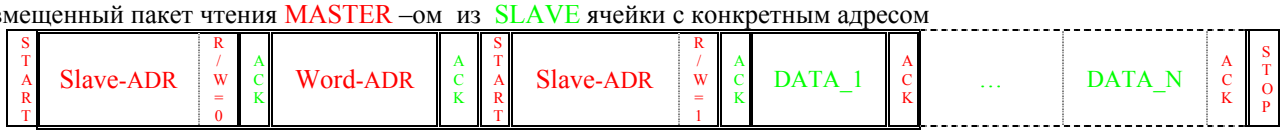
По шине I2C прошивка поддерживает мульти-мастерный и slave режимы. При этом, как и принято, первый байт пакета расценивается как адрес I2C-устройства и признак чтения/записи. Второй *байт* при записи интерпретируется как адрес ячейки внутри микроконтроллера.
*Для полного доступа ко всем ресурсам микроконтроллера по шине **I2C** принята **страничная** адресация.*
Так, при обращении к ячейкам модуля с адресами 0...7Fh, обеспечивается доступ к RAM с адресами 0...7Fh. При обращении к ячейкам 80...0FFh обеспечивается доступ к одной из страниц общего пространства RAM размером 128 байт.
Номер подставляемой страницы лежит в ячейке RAM(0x051). Если номер станицы = 0 (по умолчанию), то будет подставлена страница с SFR-регистрами микроконтроллера. Это регистры специальных функций, они полностью определяют режимы работы микроконтроллера. При обращении к соответствующим регистрам SFR пользователь может записать/прочитать ячейку EEPROM, прочитать/стереть/записать программную FLASH память.
Для работы с шиной используется аппаратный модуль MSSP микроконтроллера PIC18Fxxxx. Микроконтроллер при сбросе настраивается на работу по шине I2C с частотой ~400 кГц в режиме 7-ми битного адреса.
Обмен ведется при помощи двух линий SCL и SDA. Схемотехника шины базируется на использовании подключения устройств с открытым коллектором (стоком). Это позволяет упростить выходные каскады устройств подключенных к шине. Пассивное состояние обоих линий «1» обеспечивается резисторами подтяжки.
Линия SCL используется для стробирования информации по линии SDA. Упрощенно можно назвать эти линии «тактовая» и «данных» соответственно. Мастер выдает сигнал SCL , а по линии SDA идет выдача данных либо мастером в цикле записи, либо Slave –ом в цикле чтения. На каждый бит данных приходится один импульс сигнала SCL. Slave- устройство всегда имеет возможность приостановить мастера, придерживая сигнал SCL в низком уровне. Закончив неотложные дела, Slave отпускает мастера дальше. После приема каждого байта принимающая сторона подтверждает прием сигналом ACK, придерживая сигнал SDA в низком уровне на 9-м импульсе SCL. Таким образом, передающая сторона получает подтверждение, что его данные дошли. Всякий обмен начинается с комбинации «старт» (при высоком SCL спад SDA) и заканчивается комбинацией «стоп» (при высоком SCL фронт SDA). Эти комбинации формирует мастер.
Для обмена данными мастера со Slave используются два основных вида пакетов - пакет записи и пакет чтения. Возможны так же совмещенные пакеты в виде слитного пакета состоящего из цикла записи разделенного с циклом чтения комбинацией «повторный старт». При этом каждый из циклов имеет формат обычных пакетов, но использование между ними комбинации «повторный старт» гарантирует слитную передачу пакета, без захвата шины другими мастерами.
Первый байт в пакете всегда Slave – адрес с признаком чтения / записи в младшем бите. Этот байт всегда передает мастер. Остальные байты это читаемые или записываемые данные.
Пакет при записи MASTER –а в SLAVE по адресу Word-ADR N байт



Пакет при чтении MASTER –ом из SLAVE по текущему адресу указателя Word-ADR N байт



Совмещенный пакет чтения MASTER –ом из SLAVE ячейки с конкретным адресом



Работа контроллера в качестве Slave устройства на шине I2C.

Для работы в качестве Slave- устройства контроллер имеет Slave – адрес, хранящийся в ячейке EEPROM(0xFE). Этот адрес может быть изменен пользователем. Обратите внимание – адрес Slave-устройства I2C и адрес узла MODBUS - это разные адреса и хранятся они в разных ячейках.
В цикле записи первый байт следующий после Slave – адреса встроенная программа расценивает как адрес ресурса внутри контроллера. Это так называемый Word– адрес. Word– адрес перезаписывается во внутренний указатель при каждом пакете записи с совпадающим Slave- адресом. Следующие за этим операции чтения/записи производятся начиная с ячейки адресуемой Word– адресом с последующим автоинкрементом внутреннего указателя.

Поскольку для адресации внутреннего ресурса при доступе по шине I2C используется только один байт, то для удобства обращения ко всем внутренним ресурсам принята страничная адресация пространства RAM :
- вся RAM разбивается на страницы по 128 байт;
- при обращении к ячейкам RAM 0...7Fh, обеспечивается доступ к RAM с адресами 0...7Fh;
- при обращении к ячейкам RAM 80...0FFh обеспечивается доступ к одной из страниц размером 128 байт общего пространства RAM;
- номер подставляемой страницы лежит в ячейке REG_PAGE RAM 51h ;
- если номер станицы = 0 (по умолчанию), то будет подставлена страница с SFR-регистрами.
После включения питания контроллера Word– адрес фактически прямо адресует так называемую станицу быстрого доступа – начальные 128 байт RAM и все регистры специальных функций - SFR-регистры.
Как уже было сказано, каждое последующее чтение/запись производится с автоинкрементом внутреннего указателя. При значениях указателя =7Fh и =0FFh автоинкремент не производится. Так же автоинкремент не производится при обращении к SFR-регистрам с адресами 0A6h, 0A8h и 0F5h. Эти регистры инициируют работу с EEPROM и FLASH. При записи в TRISC, бланкируются биты 3 и 4, чтобы не изменить режим I2C.
Максимальная длинна пакета в принципе не ограничена, но учитывая что автоинкремент внутреннего указателя через границу 128 байт не производится, максимальное количество данных в пакете 128 байт.

Чтение / запись / стирание EEPROM и FLASH в режиме Slave устройства на шине I2C.
Для чтения ячейки EEPROM необходимо заранее указать адрес ячейки EEPROM путем записи в SFR-регистр EEADR (при REG_PAGE=0 Word-ADR= 0A9h) . После этого считать значение регистра EEDATA (при REG_PAGE=0 Word-ADR= 0A8h). При каждом последующем чтении без переопределения Word-ADR будет выдаваться значение следующей ячейки EEPROM. При этом автоинкремент внутреннего указателя Word-ADR не производится и он продолжает указывать на регистр EEDATA, а производится автоинкремент регистра EEADR, что удобно для последовательной выборке данных из EEPROM .

Для записи ячейки EEPROM необходимо заранее указать адрес ячейки EEPROM путем записи в SFR-регистр EEADR (при REG_PAGE=0 Word-ADR= 0A9h) . После этого записать в одном пакете новое значение ячейки в регистр EEDATA (при REG_PAGE=0 Word-ADR= 0A8h) сопровождаемое обязательной последовательностью из двух байт 55h и 0AAh. После этого внутренняя программа занесет новое значение по указанному адресу. За один пакет можно записать только один байт. Если контроллер не получит в одном пакете обязательной последовательности после получения новых данных, то запись производится не будет.
При каждой последующей аналогичной записи без переопределения Word-ADR будет записываться вновь получаемые данные в следующую ячейку EEPROM. При этом автоинкремент внутреннего указателя Word-ADR не производится и он продолжает указывать на регистр EEDATA, а производится автоинкремент регистра EEADR, что удобно для последовательной записи данных в EEPROM .

Для чтения ячейки FLASH необходимо заранее указать адрес путем записи в SFR-регистры TABLPTRL, TABLPTRH, TABLPTRU (при REG_PAGE=0 Word-ADR= 0F6h, 0F7h, 0F8h соответственно) нужного адреса FLASH . Запись всех регистров может быть сделана в одном пакете. После этого считать значение регистра TABLAT (при REG_PAGE=0 Word-ADR= 0F5h). При каждом последующем чтении без переопределения Word-ADR будет выдаваться значение следующей ячейки FLASH. При этом автоинкремент внутреннего указателя Word-ADR не производится и он продолжает указывать на регистр TABLAT, а производится автоинкремент регистра TABLPTR, что удобно для последовательной выборке данных из FLASH.
Область FLASH 0...1FFFh закрыта от операций чтения.

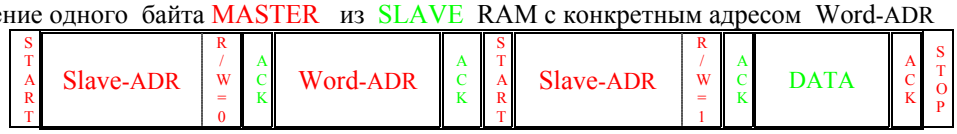
Для записи ячеек FLASH необходимо заранее указать начальный адрес путем записи в SFR- регистры TABLPTRL, TABLPTRH, TABLPTRU (при REG_PAGE=0 Word-ADR= 0F6h, 0F7h, 0F8h соответственно) нужного адреса FLASH. Запись всех регистров может быть сделана в одном пакете. Адрес обязательно должен быть кратен 8, т.е. 3 младшие бита TABLPTRL должны быть равны 0, иначе попытки записи во FLASH будут игнорироваться. После определения адреса, необходимо записать в одном пакете 8 новых значений в регистр TABLAT (при REG_PAGE=0 Word-ADR= 0F5h) сопровождаемых обязательной последовательностью из двух байт 55h и 0AAh. После этого внутренняя программа занесет новые значения, начиная с указанного адреса. За один пакет можно записать только 8 байт. Если контроллер не получит в одном пакете обязательной последовательности после получения новых данных, то запись производится не будет. При каждой последующей аналогичной записи без переопределения Word-ADR будет записываться вновь получаемые данные в следующие ячейки FLASH. При этом автоинкремент внутреннего указателя Word-ADR не производится и он продолжает указывать на регистр TABLAT , а производится увеличение регистра TABLPTR на 8 записанных ячеек, что удобно для последовательной записи данных в FLASH. Необходимость работы с 8 ячейками вызвана внутренней архитектурой микроконтроллера.
Процесс записи занимает примерно 2мс, все это время контроллер «придерживает» мастера на шине.
Область FLASH 0...1FFFh закрыта от операций записи.

Для стирания ячеек FLASH необходимо заранее указать адрес путем записи в SFR- регистры TABLPTRL, TABLPTRH, TABLPTRU (при REG_PAGE=0 Word-ADR= 0F6h, 0F7h, 0F8h соответственно) нужного адреса FLASH. Запись всех регистров может быть сделана в одном пакете. Адрес обязательно должен быть кратен 64, т.е. 6 младших бит TABLPTRL должны быть равны 0, иначе попытки стирания FLASH будут игнорироваться. После определения адреса, необходимо записать в одном пакете в регистр EECON1 (при REG_PAGE=0 Word-ADR= 0A6h) значение 10h сопровождаемое обязательной последовательностью из двух байт 55h и 0AAh. После этого внутренняя программа сотрет FLASH, начиная с указанного адреса. За один пакет можно стереть только 64 байта. Если контроллер не получит в одном пакете обязательной последовательности после получения байта 10h, то стирание производится не будет. После стирания автоинкремент внутреннего указателя Word-ADR не производится и он продолжает указывать на регистр EECON1 , автоинкремент регистра TABLPTR тоже не производится, что удобно для последующей записи данных во FLASH. Необходимость работы с 64 ячейками вызвана внутренней архитектурой микроконтроллера.
Процесс стирания занимает примерно 2мс, все это время контроллер «придерживает» мастера на шине.
Область FLASH 0...1FFFh закрыта от операций стирания.

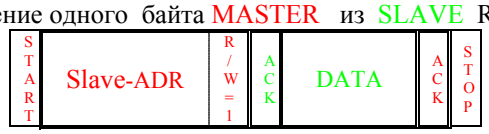
Аппаратно slave – устройство работает в режиме в котором при приеме не использует возможность приостановки мастера. В связи с этим, пользователю необходимо соблюдать рекомендацию не злоупотреблять временем нахождения в обработчиках, связанных с Hi Int (вектора: HiInt; preHiInt; 1000Hz; 100Hz). Общее время нахождения в этих обработчиках должно быть не более 25мкс, иначе может произойти потеря пакета. Если это случится, мастеру об этом будет сообщено отсутствием ACK.

Примеры некоторых команд:

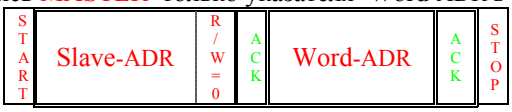
Чтение одного байта MASTER из SLAVE RAM с конкретным адресом Word-ADR



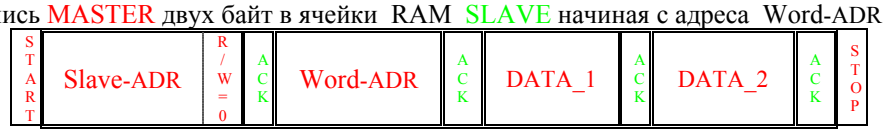
Чтение одного байта MASTER из SLAVE RAM по текущему адресу указателя Word-ADR



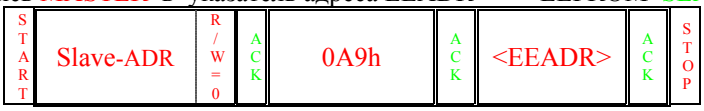
Запись MASTER только указателя Word-ADR в SLAVE



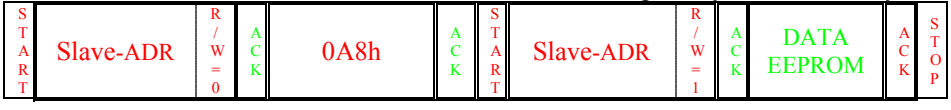
Запись MASTER двух байт в ячейки RAM SLAVE начиная с адреса Word-ADR



Запись MASTER в указатель адреса EEADR EEPROM SLAVE

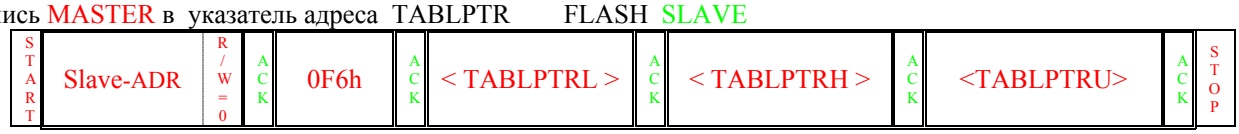


Чтение одного байта MASTER из EEPROM SLAVE по заранее установленному в EEADR адресу



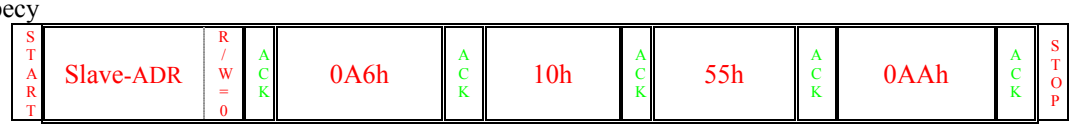
Подача MASTER команды запись ячейки EEPROM SLAVE по заранее установленному в EEADR адресу

Запись MASTER в указатель адреса TABLPTR FLASH SLAVE

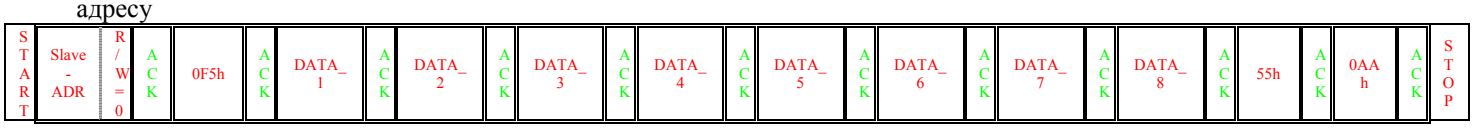


Чтение одного байта MASTER из FLASH SLAVE по заранее установленному в TABLPTR адресу

Подача MASTER команды стирание блока 64 байта в FLASH SLAVE по заранее установленному в TABLPTR адресу



Подача MASTER команды запись блока 8 байт в FLASH SLAVE по заранее установленному в TABLPTR адресу



Использование пользовательских векторов для управления потоком данных через slave-I2C.
См. раздел «***Передача управления программам пользователя».***

Работа контроллера в качестве MASTER устройства на шине I2C.

Работа в качестве MASTER по командам MODBUS
Микроконтроллер инициирует выход на шину в качестве мастера после получения одной из двух команд MODBUS : - **7Ah** - команда чтение шины I2C и **7Bh** - команда запись шины I2C. Формат команд описан в разделе «Реализация протокола MODBUS».
Выход на шину производится с соблюдением правил работы с мультимастерной шиной. Если шина занята, то контроллер ждет освобождения шины 10 мс и если шина продолжает быть занята то в MODBUS отправляется квитанция с кодом ошибки 0Dh. Если шина свободна, то микроконтроллер передает в шину пакет, соответствующий пришедшей команде. Общий формат пакетов обмена описан в начале раздела «Реализация шины I2C».

Для команды шины I2C 7Ah :
- комбинация старт
- передается Slave-ADR с признаком записи
- принимается сигнал квитирования ACK от Slave
- передается Word-ADR
- принимается сигнал квитирования ACK от Slave
- комбинация повторный старт
- передается Slave-ADR с признаком чтения
- принимается сигнал квитирования ACK от Slave
- принимается 1-й байт от Slave
- передается сигнал квитирования ACK=0 в Slave
- принимается … байт от Slave
- передается сигнал квитирования ACK=0 в Slave
- принимается N-й байт от Slave
- передается сигнал квитирования ACK=1 в Slave
- комбинация стоп

Для команды запись шины I2C 7Bh :
- комбинация старт
- передается Slave-ADR с признаком записи
- принимается сигнал квитирования ACK от Slave
- передается Word-ADR
- принимается сигнал квитирования ACK от Slave
- передается 1-й байт в Slave
- принимается сигнал квитирования ACK от Slave
- передается … байт в Slave
- принимается сигнал квитирования ACK от Slave
- передается N-й байт в Slave
- принимается сигнал квитирования ACK от Slave
- комбинация стоп

Во время передачи пакета постоянно контролируется шина на предмет возникновения коллизий (при синхронном начале передачи пакетов двумя или более разными мастерами). В случае возникновения коллизии в MODBUS отправляется квитанция с кодом ошибки 0Eh. Так же контроллер принимает квитанции ACK от Slave. В случае отсутствия ACK в MODBUS отправляется квитанция с кодом ошибки 0Fh.
Максимальная частота шины, выдаваемая мастером, 400 кГц.
Slave – устройство, при необходимости, может придержать обмен зажимая сигнал SCL на нужное время.

Использование программой пользователя встроенных подпрограмм для работы в качестве MASTER I2C.

При написании своих программ, пользователь может использовать для доступа к Slave-устройствам I2C встроенные подпрограммы I2C_RD и I2C_WR. Эти подпрограммы обеспечивают простой доступ к ресурсам, подключенным к модулю по шине I2C. Подпрограммы работают по алгоритмам уже описанным в этом разделе.

Адрес I2C_RD = 0x1E00.
Адрес I2C_WR = 0x1E04.

Вызов подпрограмм необходимо производить командами CALL.
Перед вызовом необходимо разместить:
RAM 60h=SlaveAdr
RAM 61h =WordAdr
RAM 62h=число байт (значение 0x00 соответствует 256 байтам)
FSR2 =начало буфера для приема/передачи данных.
КОДЫ ОШИБОК возвращаемые в WREG
00h-нормальное завершение обмена
0Dh-при обращении к каналу I2C шина занята > 10мс
0Eh-коллизия при обмене по I2C
0Fh-нет сигнала подтверждения ACK от Slave-устройства I2C .

Не рекомендуется обращаться к этим подпрограммам из программ вызываемых по прерываниям 10 000, 1000, 100 Гц, т.к. обмен по шине I2C занимает в любом случае время большее чем допустимо для этих программ.

Работа с SPI

В зависимости от того присутствует ли в узле/модуле шина I2C, шина SPI может быть реализована как программно так и с использованием аппаратного порта если он свободен от I2C. При программной реализации максимальная скорость обмена по шине SPI не более 700 кГц.
При аппаратной реализации - скорость обмена по шине SPI 2 МГц.
Поддерживается только мастер-режим. Для обмена по шине используются три линии:
SCK – выход тактового сигнала, Din – вход данных, Dout – выход данных. Все линии однонаправленные.
Для обмена по шине мастер формирует на линии SCK тактовые импульсы – по восемь на каждый байт.
По линиям Din и Dout соответственно, модуль синхронно с тактовым сигналом принимает и передает данные. При программной реализации рабочим является переход SCK из 0 в 1 т.е. фронт. Данные на линии Dout устанавливаются минимум за 100нс до фронта SCK. Данные с линии Din защелкиваются через 100нс после фронта SCK. При аппаратной реализации рабочими являются те же перепады, но при необходимости режим SPI можно легко изменить изменив биты CKE и CKP в регистре SSPSTAT.
Шина полнодуплексная т.е. обмен ведется одновременно в обе стороны.
Для подключения нескольких Slave-устройств используются прямая адресация устройства через дополнительные линии RA0, RA1, RA2. Т.к. адресных линий всего 3 то может быть адресовано до 8 Slave-устройств. Если Slave-устройств не более 3, то рекомендуется использовать четыре комбинации адресных сигналов: 111 – не выбрано ни одно из устройств,
011 – выбрано устройство у которого CS подключен к линии RA2,
101 – выбрано устройство у которого CS подключен к линии RA1,
110 – выбрано устройство у которого CS подключен к линии RA0.
Такой подход позволяет обойтись без дополнительных схем дешифрации.

Шина SPI доступна для пользователя двумя способами:

- При помощи команды MODBUS - **7Ch –** обмен по шине SPI;
- Используя стандартную библиотеку подпрограмм.

При каждом обращении к шине модуль автоматически настраивает линии SCK, Din, Dout в необходимое направление передачи.
На адресные линии RA0, RA1, RA2 выдаются соответственно 3 младших бита CS.
Команда самостоятельно **не** изменяет режимы работы этих линий. Поэтому, если необходима адресная работа с модулями SPI , то необходимо заранее проинициализировать эти линии как выходы.
В противном случае они останутся входами и комбинация CS не будет выведена на них. Этот режим удобен при работе с единственным SPI –модулем (линии RA0, RA1, RA2 можно использовать для других целей).

Работа в качестве MASTER по командам MODBUS
Формат команд описан в разделе «Реализация протокола MODBUS»

Использование программой пользователя встроенных подпрограмм для работы в качестве MASTER SPI.
При написании своих программ, пользователь может использовать для доступа к Slave-устройствам SPI встроенную подпрограмму SPI . Эта подпрограмма обеспечивают простой доступ к модулям, подключенным по шине SPI.
Адрес подпрограммы SPI = 0x1E10.
Вызов подпрограмм необходимо производить командами CALL.
Перед вызовом необходимо разместить:
RAM 58h=число байт (фактически общее число требуемых импульсов по SCK /8)
RAM 5Ah= CS
FSR2 =начало буфера для приема/передачи данных.
КОДЫ ОШИБОК возвращаемые в WREG
00h-нормальное завершение обмена
03h-число запрашиваемых байт =0
04h-число запрашиваемых байт >249
Т.к. подпрограмма пользуется общими ресурсами с участком резидентной программы отвечающим за обработку телеграмм приходящих из MODBUS, надо следить, чтобы вызов подпрограмм не производился при работе с MODBUS.
Для этого необходимо дождаться когда `REG_MODBUS -> RAM 53h =B'00010000' - контроллер готов к приему новой телеграммы.`
После этого обнулить регистр(тем самым временно запретив работу с MODBUS), провести обмен по SPI, восстановить `REG_MODBUS==B'00010000'.`
Не рекомендуется обращаться к этой подпрограмме из программ вызываемых по прерываниям 10 000, 1000, 100 Гц, т.к. обмен может занимать время большее чем допустимо для этих программ.

Работа с MicroLan

Все линии узла/модуля незадействованные в реализации основных интерфейсов (MODBUS и I2C) могут быть использованы для работы с термодатчиками фирмы *Dallas Semiconductor* DS18S20 или DS18B20.
При работе модуль использует безадресный метод обмена, поэтому к каждой линии можно подключить только один термодатчик. Термодатчики могут подключаться как с выделенным питанием , так и с питанием от линии(другими словами по 3-проводной или по 2-проводной схемам).
Схемотехника шины MicroLan требует подтяжки на стороне узла/модуля соответствующих линий данных к +5В резисторами ~5 кОм.
Обратите внимание – есть некоторые типы модулей на которых эти резисторы не предусмотрены;
на некоторых узлах для включения этих резисторов необходимо установить соответствующие джамперы!

Для работы с MicroLan в RAM модуля выделены:
- RAM 5Ch - секвенсор (0=>стоп/конец преобразования, 1=>запуск преобразования)
- RAM 5Dh - битовая маска линий разрешенных в MicroLan для порта RA (1=> разрешено)
- RAM 5Eh - битовая маска линий разрешенных в MicroLan для порта RB (1=> разрешено)
- RAM 5Fh - битовая маска линий разрешенных в MicroLan для порта RC или RF->для узлов с PIC18F6xxx (1=> разрешено)
- RAM 300h...3EFh - область памяти с результатами преобразования.

Если пользователь не работает с MicroLan, то область памяти RAM 300h...3EFh остается свободной.
Для работы с линиями в режиме MicroLan необходимо установить соответствующие биты в регистрах RAM 5Dh…5Fh. Это достаточно сделать один раз – значение этих ячеек программа не меняет.
Специально устанавливать режимы линий микроконтроллера не надо, программа это делает автоматически для разрешенных линий.
Для запуска преобразования необходимо записать в RAM 5Ch значение 0x01.
Результат преобразования по всем разрешенным линиям одновременно будет готов не более чем через 1с.
После окончания преобразования RAM 5Ch сбросится в 0.

После инициализации RAM 5Dh…5Fh и запуска преобразования RAM 5Ch=1 модуль выдает необходимые последовательности в разрешенные линии.
Если соответствующая линия в момент начала обмена не была в «1» **или** не пришел импульс «присутствия» от термодатчика, то обмен с этой линией в этот конкретный запуск больше не ведется – линия будет находится в пассивном состоянии до следующей попытки.(для диагностики обрывов/КЗ на линии)
В «нормальные» линии передается запускающая последовательность.
После на эти линии подается питание +5В - для запитки датчиков, при двухпроводном подключении, на время преобразования. По прошествии необходимого датчику для преобразования времени, подаются необходимые последовательности для считывания результата.
Из каждого датчика считывается по 9 байт – все ячейки включая CRC8.
Далее модуль производит подсчет CRC8 и сравнение с пришедшей CRC8 для всех линий.
Результат сравнения модуль дописывает в 10-й байт.
Массив данных по всем возможным линиям расположен в RAM 300h...3EFh.
Для каждого датчика отведено по 10 байт – 9 принятых и 1 диагностический.
Данные расположены в фиксированных местах с шагом 10 байт начиная с линии RA0 и кончая RC7
вне зависимости от того, разрешена или запрещена соответствующая линия.
Обратите внимание - программа считает что во всех портах по 8 линий - т.е. всего 24 линии.
Это необходимо учитывать для точного нахождения соответствующих данных.
В каждом пакете данных результат измерения температуры расположен в первых двух байтах(см. описания DS18S20 или DS18B20), принятый CRC8 в девятом байте, десятый байт –диагностический.
Если принятый и вычисленный CRC8 совпадают, то в диагностическом байте взводится бит №1=1 (! при КЗ в линии он совпадет! – будут приняты все 0 и следовательно CRC8=0 !).
Бит №0 взводится в 1 если соответствующая линия в момент начала обмена была в «1» **и** пришел импульс «присутствия» от термодатчика.
Таким образом если обмен с датчиком прошел нормально и совпал CRC8, то будут установлены оба эти бита и 10-й байт пакета относящегося к этой линии будет равен «0000 0011»=3.

Распределение ресурсов микроконтроллера

FLASH память

Микроконтроллер может иметь от 32 до 64 кБайт программной FLASH памяти. Из них под резидентную программу отведено 8 кБайт. Это FLASH память с адресами 0x0000… 0x1FFF. Доступ к этому участку памяти закрыт. При необходимости, резидентную программу можно стереть и записать новую программатором.
Программы пользователя могут располагаться с адреса 0x2000 и занимать всю оставшуюся память.
В некоторых микроконтроллерах , таких как PIC18F2620, PIC18F6621 программы пользователя размещаются с адреса 0x4000 из за более крупного разбиения в них FLASH-памяти на блоки.
Начало пользовательской области FLASH -памяти для различных типов микроконтроллеров:

- PIC18F252 =>0x2000
- PIC18F2520 =>0x2000
- PIC18F2523 =>0x2000
- PIC18F4520 =>0x2000
- PIC18F2620 =>0x4000
- PIC18F6520 =>0x2000
- PIC18F6527 =>0x2000
- PIC18F6621 =>0x4000

Программа пользователя может быть занесена при помощи внешнего программатора через специальный разъем или, что более удобно, через любой из основных последовательных каналов (UART или I2C) резидентной программой.
Команды записи/чтения/стирания для каждого из интерфейсов уже описаны выше.
Для удобства занесения программ пользователя доступна программа программатора для PC.
Она позволяет заносить HEX-файл во FLASH-память напрямую через MODBUS.

С адреса 0x1E00 расположена таблица вызовов стандартных подпрограмм. Пользователь может использовать эти подпрограммы для упрощения написания своих программ.

0x1E00 I2C_RD - подпрограмма чтения из slave I2C
0x1E04 I2C_WR - подпрограмма записи в slave I2C
RAM 60h=SlaveAdr
RAM 61h =WordAdr
RAM 62h=число байт (значение 0x00 соответствует 256 байтам)
FSR2 =начало буфера для приема/передачи данных.
КОДЫ ОШИБОК возвращаемые в WREG
00h-нормальное завершение обмена
0Dh-при обращении к каналу I2C шина занята > 10мс
0Eh-коллизия при обмене по I2C
0Fh-нет сигнала подтверждения ACK от Slave-устройства I2C
0x1E08 CRC16_Bl - Подсчет контрольной суммы блока данных CRC16
FSR2=начало блока
RAM 58h=длинна
CRC_H->RAM 5Bh
CRC_L->RAM 5Ah
при выходе из п/п FSR2 стоит на байте за последним обработанным
0x1E10 SPI - подпрограмма обмена со slave-устройством SPI(доступна начиная с версии 1.17)
RAM 58h=число байт (фактически общее число требуемых импульсов по SCK /8)
RAM 5Ah= CS
FSR2 =начало буфера для приема/передачи данных.
КОДЫ ОШИБОК возвращаемые в WREG
00h-нормальное завершение обмена
03h-число запрашиваемых байт =0
04h-число запрашиваемых байт >249

Использование п/п I2C и SPI лучше производить не на уровнях прерываний(вектора : инициализация, MAIN, дальнейшая обработка команд пользователя MODBUS, передать управление один раз.)
При вызове этих п/п в теле векторов работающих на уровне прерываний возможна неправильная работа(одни и те же ресурсы могут быть использованы этим вызовом и например обработкой пришедшей команды MODBUS – обработка этих же интерфейсов). Вызов этих процедур из перечисленных векторов гарантирует отсутствие коллизий.
После отработки п/п I2C и SPI регистр числа байт обмена не сохраняется.

RAM память

Узел/модуль может иметь от 1536 до 3936 байт памяти данных.

Объем RAM для различных типов микроконтроллеров:

- PIC18F252 =>1536
- PIC18F2520 =>1536
- PIC18F2523 =>1536
- PIC18F4520 =>1536
- PIC18F2620 =>3986
- PIC18F6520 =>2048
- PIC18F6527 =>3936
- PIC18F6621 =>3840

Из них под прошивку зарезервированы:

- RAM(0x050…0x09F) –рабочие регистры
- RAM(0x300…0x3EF) –буфер MicroLan (если ведется работа с MicroLan)
- RAM(0x500…0x5FF) –буфер UART1
- RAM(0x600…0x6FF) –буфер UART2

Рабочие регистры представляющие интерес для пользователя:

- RAM 50h - REG_USER -регистр пользователя для передачи управления программам пользователя
- RAM 51h - REG_PAGE - регистр страницы для обеспечения доступа ко всей RAM при доступе через I2C
- RAM 52h - ADR_MODBUS1 - регистр адреса узла MODBUS -> для UART1
- RAM 53h - REG_MODBUS1 - регистр управления режимами MODBUS -> для UART1
- RAM 54h - ячейка для формирования сброса при записи 55h, после сброса =0
- RAM 56h - число принятых байт включая CRC16 в ответ на мастер-телеграмму для UART1
- RAM 58h - число байт мастер-телеграммы SPI в стандартных п/п
- RAM 5Ah - адрес / CS / CRC_L в стандартных п/п
- RAM 5Bh - CRC_H в стандартных п/п
- RAM 5Ch - секвенсор MicroLan (0=>стоп/конец преобразования, 1=>запуск преобразования)
- RAM 5Dh - битовая маска линий разрешенных в MicroLan для порта RA (1=> разрешено)
- RAM 5Eh - битовая маска линий разрешенных в MicroLan для порта RB (1=> разрешено)
- RAM 5Fh - битовая маска линий разрешенных в MicroLan для порта RC (1=> разрешено)
- RAM 60h - SlaveAdr I2C для встроенных подпрограмм
- RAM 61h - WordAdr I2C для встроенных подпрограмм
- RAM 62h - число байт обмена I2C для встроенных подпрограмм(значение 0x00 соответствует 256 байтам)
- RAM 6Ah - номер канала UART: '00000001'->UART1, '00000010'->UART2
- RAM 6Ch - N_TX1 - число байт мастер-телеграммы MODBUS для UART1 (без CRC16)(ячейка не портится после выполнения подпрограммы)
- RAM 6Dh - N_TX2 - число байт мастер-телеграммы MODBUS для UART2 (без CRC16))(ячейка не портится после выполнения подпрограммы)
- RAM 70h - REG_USER_2 - регистр пользователя для передачи управления программам пользователя
- RAM 72h - ADR_MODBUS2 - регистр адреса узла MODBUS -> для UART2
- RAM 73h - REG_MODBUS2 - регистр управления режимами MODBUS -> для UART2
- RAM 76h - число принятых байт включая CRC16 в ответ на мастер-телеграмму для UART2
- RAM 7Ch…7Fh - системные часы, наращиваются каждые 1 мс.
- RAM 80h…85h - часы реального времени и календарь, для соответствующих исполнений.

Пользователю предоставлены 80 Байт RAM по адресам 0...4Fh и область RAM начиная с адреса 0x0A0 за исключением буферов UART(-ов) и MicroLan если они используются. Эта память сохраняет свое состояние при горячем рестарте(без просада питания).

При подаче питания (холодном старте) в ячейки RAM с адресами 400h...4FFh копируется идентификатор модуля. При необходимости пользователь может использовать эту информацию до того как его программа займет эту память под свои нужды. Идентификатор пользователь может считать и после «затирания» ячеек RAM с адресами 400h...4FFh при получении ответа на команду MODBUS 78h.

EEPROM память

Узел/модуль может иметь от 256 до 1024 байт EEPROM памяти данных.

Объем EEPROM для различных типов микроконтроллеров:

- PIC18F252 =>256
- PIC18F2520 =>256
- PIC18F2523 =>256
- PIC18F4520 =>256
- PIC18F2620 =>1024
- PIC18F6520 =>1024
- PIC18F6527 =>1024
- PIC18F6621 =>1024

Из них под резидентную программу зарезервировано 16 Байт по адресам EEPROM(0xF0...0xFF) .

Ячейки представляющие интерес для пользователя:

- EEPROM(0xEF)<= коррекция хода часов реального времени
- EEPROM(0xF4)<= младший байт маркера встроенного теста
- EEPROM(0xF5)<= старший байт маркера встроенного теста
- EEPROM(0xF6)<= младший байт скорости UART2
- EEPROM(0xF7)<= старший байт скорости UART2
- EEPROM(0xF8)<= ((адрес второго I2C))
- EEPROM(0xF9)<= адрес UART2
- EEPROM(0xFA)<= младший вектор пользователя
- EEPROM(0xFB)<= старший вектор пользователя
- EEPROM(0xFC)<= младший байт скорости UART1
- EEPROM(0xFD)<= старший байт скорости UART1
- EEPROM(0xFE)<= адрес I2C (! Не дожжен быть меньше 0x10)
- EEPROM(0xFF)<= адрес UART1

*Таблица скоростей для разных типов микроконтроллеров при тактовой частоте 8.000МГц *4PLL :*

Микроконтроллер	***BRG***	***9600***	***19200***	***57600***	***115200***
PIC18F252	8бит	0xCF	0x67	0x21	0x10
PIC18F2520	16бит	0x0340	0x019F	0x0089	0x0044
PIC18F2523	16бит	0x0340	0x019F	0x0089	0x0044
PIC18F4520	16бит	0x0340	0x019F	0x0089	0x0044
PIC18F2620	16бит	0x0340	0x019F	0x0089	0x0044
PIC18F6520	8бит	0xCF	0x67	0x21	0x10
PIC18F6527	16бит	0x0340	0x019F	0x0089	0x0044
PIC18F6621	16бит	0x033F	0x019E	0x0088	0x0043

Новый модуль поставляется с адресами ADR_MODBUS1 =0x02, ADR_MODBUS2 =0x04 ,
I2C Slave-ADR =0x10 и скоростями 115200.

SFR –регистры

В самых верхних адресах RAM в микроконтроллерах расположены так называемые SFR –регистры.
Все они доступны пользователю. В зависимости от типа микроконтроллера SFR –регистры начинаются с уровня RAM(0xF50 или 0xF60 или 0xF80) и заканчиваются RAM(0xFFF)
При доступе по каналу I2C при REG_PAGE=0 или REG_PAGE=0x1F доступны регистры области RAM(0xF80…0xFFF) соответственно для Word-ADR= 0x80…0xFF . Оставшаяся часть регистровой памяти, если она конечно присутствует, доступна при REG_PAGE=0x1E
Пользоваться SFR –регистрами надо с известной долей осторожности – от их значений полностью зависит состояние микроконтроллера и иногда неправильное изменение даже одного бита может привести к неправильной работе контроллера в целом. Резидентной программой используются узлы микроконтроллера:

TMR3(системное время),
UARTx (MODBUS),
MSSP(I2C).

Для обеспечения правильной работы резидентной программы не рекомендуется менять режимы работы этих узлов.
При самостоятельной работе пользователя с системой прерываний пользователь может запрещать разрешенные системой прерывания и при возврате из своей программы в систему восстанавливать.

Служба времени

В службу времени входят вектора программ пользователя 10 000 Гц, 1000 Гц, 100 Гц и 32 –битный счетчик времени. Счетчик наращивается каждую миллисекунду. Он расположен в ячейках 0x7C – 0x7F.
Обнуление счетчика происходит только при включении, просадке питания, подаче команды MODBUS-> 79h инициализация контроллера или при записи в ячейку инициализации RAM 54h-> 55h.
Таймер переполняется через 1193 часа ~=50 суток. При желании пользователь может записывать в счетчик необходимое ему новое значение. Прерывания 10 000 Гц, 1000 Гц, 100 Гц удобны для тактирования программ пользователя, легкого и быстрого создания различных таймеров и т.д.

Часы реального времени

Некоторые модули и узлы поддерживают работу узла часов реального времени (RTC) и календаря.
Этот узел работает при отсутствии основного питающего напряжения от собственного элемента питания.
Для пользователя это 6 ячеек RAM с 0x80 по 0x85. Формат данных двоично-десятичный.

- 0x80 -единицы и десятки секунд (0…59);
- 0x81 -единицы и десятки минут (0…59);
- 0x82 -единицы и десятки часов (0…23);
- 0x83 -число (1…28/31);
- 0x84 -месяц (1…12);
- 0x85 -год (00…99).

В ячейке EEPROM 0xF4 хранится поправочный коэффициент для точной коррекции хода RTC:

- 0x00=0,
- 0x01=+0.044с/сутки,
- 0x7F=+5.58с/сутки,
- 0xFF=-0.044с/сутки,
- 0x80=-5,625с/сутки.

Ход календаря учитывает високосные года.

Передача управления программам пользователя

Для передачи управления в программы пользователя предусмотрен регистр пользователя REG_USER - ячейка RAM(0x50) . Начиная с версии 4.13 доступен еще и второй регистр REG_USER _2 => RAM(0x70) .Установкой нужных флагов в этих регистрах пользователь разрешает передачу управления по фиксированным векторам. Эти вектора расположены в странице FLASH с 0x2000 по 0x203F или с 0x4000 по 0x403F в зависимости от типа микроконтроллера(см. выше) .
Каждый из флагов разрешает передачу управления на свой вектор после выполнения определенных условий и участков резидентной программы:
- REG_USER<7>=1 передать управление на адрес 0x2000 после завершения участка инициализации резидентной программы;
- REG_USER<6>=1 передать управление на адрес 0x2004 перед завершением цикла MAIN резидентной программы;
- REG_USER<5>=1 передать управление на адрес 0x2008 после обработки прерывания с высоким приоритетом (TMR3 =10 000 Гц);
- REG_USER<4>=1 передать управление на адрес 0x200C после обработки прерывания с низким приоритетом (I2C, UART);
- REG_USER<3>=1 передать управление на адрес 0x2010 для дальнейшей обработки команд пользователя MODBUS;
- REG_USER<2>=1 передать управление на адрес 0x2014 каждые 1000 мкс;
- REG_USER<1>=1 передать управление на адрес 0x2018 каждые 10 000 мкс;
- REG_USER<0>=1 передать управление на адрес 0x201C один раз;

- REG_USER_2<7>=1 передать управление на адрес 0x2020 до обработки прерывания с высоким приоритетом;
- REG_USER_2<6>=1 передать управление на адрес 0x2024 до обработки прерывания с низким приоритетом;
- REG_USER_2<5>=1 передать управление на адрес 0x2028 перед чтением ячейки по запросу через I2C;
- REG_USER_2<4>=1 передать управление на адрес 0x202C перед записью ячейки по запросу через I2C;
- REG_USER_2<0>=1 запретить восстановление заводских установок и очистку векторов пользователя при 10 сбросах в течении 1 минуты после подачи питания.

Для удобства пользователю рекомендуется страницу FLASH с 2000h по 203Fh использовать только для таблицы переходов в свои программы, а программы располагать с адреса 2040h .В этом случае можно переписывать таблицу, не затрагивая уже записанные программы. По векторам, неиспользуемым пользователем, рекомендуется располагать команды RETURN, чтобы случайный взвод ненужного флага в регистре пользователя не привел к уходу программы в неизвестное место.

При установке REG_USER<7>=1 контроллер после завершения участка инициализации резидентной программы попадает на вектор расположенный по адресу 2000h . Здесь должен быть заранее подготовлен переход на пользовательскую программу инициализации (желательно использовать длинный переход GOTO).
Этот бит имеет смысл взводить только в копии регистра в EEPROM 0FAh, т.к. программа инициализации запускается при рестарте, а при рестарте содержимое EEPROM 0FAh копируется в REG_USER.
В программе инициализации удобно настраивать параметры, которые настраиваются один раз. Например, проинициализировать АЦП, ШИМ, Таймеры, разрешить прерывания от своих источников, настроить порты ввода/вывода и т.д. В конце программы должен стоять оператор RETURN.
Время нахождения в этой программе пользователя не ограничено, просто оно добавится ко времени инициализации всего модуля. Если пользователь хочет выполнить свою программу инициализации сразу после ее занесения во FLASH , нужно записать 55h в ячейку RAM 54h и тем самым сформировать рестарт или послать команду по MODBUS 79h – инициализация контроллера.
К моменту запуска этой программы пользователя уже разрешены все прерывания , работает служба времени, обрабатываются MSSP и UART. Уже работает доступ к ресурсам контроллера через I2C, но не обрабатываются приходящие телеграммы MODBUS - они обрабатываются в цикле MAIN .
В случае, если пользователь хочет полностью перехватить управление на свою программу, возврат из своей программы делать не надо. Но в этом случае резидентная программа перестает работать, и все общение с внешним миром будет лежать на программе пользователя.

При установке REG_USER<6>=1 контроллер перед завершением основного цикла MAIN резидентной программы попадает на вектор расположенный по адресу 2004h . Здесь должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
Кроме пользовательской программы в цикле MAIN резидентная программа обрабатывает пришедшие телеграммы MODBUS, пользовательский вектор «передать управление на адрес 201Ch один раз» и некоторые события системы. На обработку системных дел контроллер тратит очень мало времени(исключение составляют

ситуации обработки команд работы с записью / стиранием FLASH и работа с каналом I2C) . Поэтому контроллер фактически постоянно запускает эту программу пользователя если взведен флаг REG_USER<6>. По этому вектору удобно располагать программы работающие постоянно. Программа не должна быть закольцована, т.е. она должна выполняться за один проход, учитывая то , что практически сразу после завершения управление снова передается на ее начало. В конце программы должен стоять оператор RETURN. Время нахождения в этой программе пользователя не ограничено, просто пока контроллер находится в ней не будут выполняться вновь пришедшие команды MODBUS . Во время работы этой программы пользователя в системе разрешены все прерывания и поэтому работа с Slave- I2C и прием телеграммы MODBUS будут осуществляться, служба времени будет нормально работать.

При установке REG_USER<5>=1 после срабатывания прерывания с высоким приоритетом и необходимых проверок резидентной программой управление передается на адрес 2008h . Здесь должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …). В системе прерывание с высоким приоритетом используется для службы времени. Таймер TMR3 настроен для генерации прерываний с частотой 10 000 Гц т.е. каждые 100 мкс. Поэтому при разрешении программы пользователя по этому вектору, пользователю передается управление каждые 100 мкс или при срабатывании прерывания с высоким приоритетом от заранее разрешенного пользователем ресурса. Для того чтобы не нарушать нормальную работу резидентной программы программа пользователя должна выполняться не более 25 мкс, т.е. быть не длиннее чем примерно 200 команд. При этом надо учитывать затрачиваемое время при выполнении пользовательских программ по векторам 1000 Гц, 100 Гц . Не желательно превышение лимита 25 мкс всеми этими программами в сумме, т.к. все они обрабатываются в теле одного прерывания.
В конце программы должен стоять оператор RETURN.
При попадании на этот вектор в контроллере запрещены все прерывания(контроллер находится внутри прерывания с высоким приоритетом) и пользователь может быть уверен в том что контроллер выполняет только его программу, не прерываясь.
Этим вектором удобно пользоваться для точного измерения временных интервалов, быстрой строго периодической обработки различных процессов.

При установке REG_USER<4>=1 после срабатывания прерывания с низким приоритетом и необходимых проверок резидентной программой управление передается на адрес 200Ch . Здесь должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …). В системе прерывание с низким приоритетом используется для обработки прерываний от MSSP и UART . При разрешении программы пользователя по этому вектору, пользователю передается управление при каждом прерывании от MSSP и UART **и** при срабатывании прерывания с низким приоритетом от заранее разрешенного пользователем ресурса. Для того чтобы не нарушать нормальную работу резидентной программы программа пользователя не должна выполняться долго. Т.к. пока она незакончена, не будут обработаны прерывания от MSSP и UART и из-за этого могут быть потеряны принятые ими данные. В конце программы должен стоять оператор RETURN. После возврата, система самостоятельно восстанавливает регистры BSR, WREG, STATUS к состоянию до входа в прерывание.
При попадании на этот вектор в контроллере запрещено прерывание с низким приоритетом(контроллер находится внутри него) . Прерывание с высоким приоритетом разрешено и поэтому служба времени продолжает нормально работать. При входе в обработчик пользователя в WREG лежит B'00010000'.
Этим вектором удобно пользоваться для обработки заранее разрешенных прерываний с низким приоритетом от ресурсов необходимых пользователю.

При установке REG_USER<3>=1 передается управление на адрес 2010h для дальнейшей обработки команд пользователя MODBUS. Управление передается если: получена телеграмма MODBUS , в ней совпал адрес узла MODBUS и CRC и пришедшая команда неизвестна резидентной программе. По адресу 2010h должен быть заранее подготовлен переход на пользовательскую программу разбора (желательно использовать длинный переход GOTO …). Время нахождения в этой программе пользователя не ограничено, просто пока контроллер находится в ней, не будут обрабатываться новые команды MODBUS и основной цикл MAIN .
Во время работы этой программы пользователя в системе разрешены все прерывания и поэтому работа с Slave-I2C и прием телеграммы MODBUS будут осуществляться, служба времени будет нормально работать.
Передача управления по этому вектору происходит при получении валидной телеграммы(совпал адрес узла и правильная CRC16). Пришедшая телеграмма расположена в буфере соответствующего UART (0x500 для UART1 и 0x600 для UART2). После обработки пользователь должен разместить ответную телеграмму в этом же буфере, и занести в FSR2L длинну телеграммы без CRC16 и сделать возврат по "RETLW 0x00".
При возврате по "RETLW 0xXX" будет выдана в ответ квитанция об ошибке с кодом 0xXX.
Если не надо вообще отвечать на пришедшую телеграмму , необходимо вернуться по "RETLW 0xFF".
Этим вектором нужно пользоваться при необходимости в дополнительных пользовательских командах MODBUS .

При установке REG_USER<2>=1 управление передается на адрес 2014h каждые 1000 мс . Здесь должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
При попадании на этот вектор в контроллере запрещены все прерывания(контроллер находится внутри прерывания с высоким приоритетом) и пользователь может быть уверен в том что контроллер выполняет только его программу, не прерываясь.
Для того чтобы не нарушать нормальную работу резидентной программы программа пользователя должна выполняться не более 25 мкс, т.е. быть не длиннее чем примерно 200 команд. При этом надо учитывать уже затраченное время в пользовательских программах по векторам высокого прерывания 10 000 Гц, 100 Гц . Не желательно превышение лимита 25 мкс всеми этими программами в сумме, т.к. все они обрабатываются в теле одного прерывания.
В конце программы должен стоять оператор RETURN.
В теле обработки этого вектора, при использовании страничного регистра BSR, необходимо сохранять его и восстанавливать при выходе.
Этим вектором удобно пользоваться для точного измерения временных интервалов, тактирования различных процессов.
При установке REG_USER<1>=1 управление передается на адрес 2018h каждые 10 000 мс . Здесь должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
При попадании на этот вектор в контроллере запрещены все прерывания(контроллер находится внутри прерывания с высоким приоритетом) и пользователь может быть уверен в том что контроллер выполняет только его программу, не прерываясь.
Для того чтобы не нарушать нормальную работу резидентной программы программа пользователя должна выполняться не более 25 мкс, т.е. быть не длиннее чем примерно 200 команд. При этом надо учитывать уже затраченное время выполнения программ по векторам высокого прерывания 10 000 Гц, 1000 Гц. Не желательно превышение лимита 25 мкс всеми этими программами в сумме, т.к. все они обрабатываются в теле одного прерывания.
В конце программы должен стоять оператор RETURN.
В теле обработки этого вектора, при использовании страничного регистра BSR, необходимо сохранять его и восстанавливать при выходе.
Этим вектором удобно пользоваться для точного измерения временных интервалов, тактирования различных процессов.

При установке REG_USER<0>=1 управление передается на адрес 201Ch . Здесь должен быть заранее подготовлен переход на пользовательскую программу «однократного выполнения» (желательно использовать длинный переход GOTO …).
Проверка этого бита происходит в цикле MAIN после обработки телеграммы MODBUS, но перед обработкой вектора «передать управление на адрес 2004h перед завершением цикла MAIN резидентной программы».
Перед входом в подпрограмму бит сбрасывается, что удобно для вызова этой подпрограммой себя-же на следующем проходе по циклу MAIN. Для этого подпрограмма должна взвести этот бит.
Время нахождения в этой программе пользователя не ограничено, просто пока контроллер находится в ней не будут выполняться вновь пришедшие команды MODBUS . Во время работы этой программы пользователя в системе разрешены все прерывания и поэтому работа с Slave- I2C и прием телеграммы MODBUS будут осуществляться, служба времени будет нормально работать.
В конце программы должен стоять оператор RETURN.
Этим вектором удобно пользоваться для оперативной инициализации ресурсов, обработки параметров по запросу и т.д.

При установке REG_USER_2<7>=1 после срабатывания прерывания с высоким приоритетом управление передается на адрес 2020h еще до обработки этого прерывания основной программой . По этому адресу должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
Этот вектор может быть полезен, если пользователь решил сам обрабатывать прерывание от TMR3 или для него является существенной величина задержки от момента возникновения программы до начала работы его подпрограммы.
Для того чтобы резидентная программа все-таки проделала необходимые действия до возвращения в основную программу , в конце подпрограммы должен стоять оператор RETURN. Но если пользователь считает что он все сделал сам, то можно вернуться в прерванную основную программу минуя обработчик системы. Для этого необходимо сделать POP и уйти по RETFIE FAST.
Регистры BSR, WREG, STATUS сохранены в теневых регистрах и пользователю не надо их сохранять.

От момента попадания микроконтроллера на вектор 0x04 до момента исполнения команды GOTO лежащей по адресу 2020h проходит не более 7 циклов команд.
Надо понимать что на время нахождения в обработчике пользователя задерживается перезагрузка TMR3 и следовательно замедляется ход системного времени.

При установке REG_USER_2<6>=1 после срабатывания прерывания с низким приоритетом управление передается на адрес 2024h еще до обработки этого прерывания основной программой . По этому адресу должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
Этот вектор может быть полезен, если пользователь решил сам обрабатывать прерывания от UART –ов и I2C. Для того чтобы резидентная программа все-таки проделала необходимые действия до возвращения в основную программу , в конце подпрограммы должен стоять оператор RETURN. В этом случае регистры WREG, STATUS, BSR пользователю не надо сохранять, они сохраняются резидентной программой.
Но если пользователь считает что он все сделал сам, то можно вернуться в прерванную основную программу, минуя обработчик системы. Для этого необходимо сделать POP и уйти по RETFIE. В этом случае регистры WREG, STATUS, BSR пользователю надо перед самым выходом восстановить командами:
movff 0x89, wreg; movff 0x8A, status; movff 0x8B, bsr.
От момента попадания микроконтроллера на вектор 0x08 до момента исполнения команды GOTO лежащей по адресу 2024h проходит ровно 10 циклов команд.

При установке REG_USER_2<5>=1 после прихода запроса чтения памяти по каналу I2C управление передается на адрес 2028h еще до выполнения этого запроса основной программой . По этому адресу должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
Этот вектор может быть полезен, если пользователю необходимо отслеживать обращения через канал I2C к конкретным ресурсам и, в случае необходимости, подставлять в качестве ответа необходимые данные.
В регистре FSR2 лежит 12-битный адрес для косвенной адресации ячейки RAM или SFR регистра, вычисленный исходя из WordAdr I2C и REG_PAGE . В RAM(0x61) лежит WordAdr I2C.
Если важен сам факт обращения к конкретной ячейке, то пользователю достаточно проверить на совпадение адрес в FSR2, а потом после необходимых действий вернуться командой RETLW 0x00. При этом основная программа сама обеспечит передачу запрошенных данных и инкремент WordAdr I2C (это необходимо при чтении массива данных).
Если же пользователь хочет сформировать ответ сам, то для передачи необходимых данных надо их записать непосредственно в регистр передатчика I2C SSPBUF и вернуться командой RETLW 0xFF. При этом основная программа сама ничего не передает и не наращивает WordAdr I2C.
Эта пользовательская подпрограмма вызывается из уровня прерывания с низким приоритетом при обработке прерывания от MSSP.

При установке REG_USER_2<4>=1 после прихода запроса записи памяти по каналу I2C управление передается на адрес 202Ch еще до выполнения этого запроса основной программой . По этому адресу должен быть заранее подготовлен переход на пользовательскую программу (желательно использовать длинный переход GOTO …).
Этот вектор может быть полезен, если пользователю необходимо отслеживать обращения через канал I2C к конкретным ресурсам и, в случае необходимости, записывать или нет данные.
В регистре FSR2 лежит 12-битный адрес для косвенной адресации ячейки RAM или SFR регистра, вычисленный исходя из WordAdr I2C и REG_PAGE . В RAM(0x61) лежит WordAdr I2C.
Если важен сам факт обращения к конкретной ячейке, то пользователю достаточно проверить на совпадение адрес в FSR2, а потом после необходимых действий вернуться командой RETLW 0x00. При этом основная программа сама обеспечит прием необходимых данных и инкремент WordAdr I2C (это необходимо при записи массива данных).
Если же пользователь хочет обработать это событие сам, то может использовать пришедшие данные в регистре SSPBUF по своему усмотрению(этот регистр необходимо обязательно прочитать) и вернуться командой RETLW 0xFF. При этом основная программа сама ничего не передает и не наращивает WordAdr I2C.
Эта пользовательская подпрограмма вызывается из уровня прерывания с низким приоритетом при обработке прерывания от MSSP.

Значение REG_USER_(RAM 50h) при всех видах сброса копируется из ячейки EEPROM 0FAh.
Соответственно *Значение REG_USER_2_(RAM 70h)* при всех видах сброса копируется из ячейки EEPROM 0FBh.
Поэтому при необходимости пользователь может взвести нужные биты в ячейках EEPROM и тогда запуск необходимых программ пользователя будет происходить сразу после сброса.

Если пользователь случайно записал неправильное значение в EEPROM , или в его программах содержатся ошибка, приведшая к зависанию(скорее к перезапускам т.к. разрешена собачка) контроллера, то можно вернуть контроллер в исходное состояние. Для этого надо выключить и включить питание контроллера и в течении минуты произвести не менее 10 горячих рестартов путем замыкания на землю вывода MCLR микроконтроллера. После этого в ячейки EEPROM 0FAh и 0FBh будут занесены «0» и в ячейках EEPROM отвечающих за скорости UART -ов будут восстановлены константы для скорости 115200 бод.
Пользователь может запретить эту функцию занесением «1» в REG_USER_2<0>.

Занесение пользовательских программ во FLASH память

Программы пользователя во FLASH память могут быть занесены тремя способами.
Первый и самый простой и удобный – это использование канала MODBUS .
Сам канал у пользователя уже есть, и вероятно этот канал подключен к PC. Если это так, то для загрузки HEX-файла с программами пользователя в контроллер потребуется оболочка, предоставляемая Фирмой Фрактал. Эта программа доступна на нашем сайте www.fractal.com.ru. Пользователь должен запустить эту программу, открыть в ней свой HEX-файл и нажать кнопку «запрограммировать». Это все!
Программа выполняет необходимые преобразования форматов данных, осуществляет передачу информационных пакетов и проверяет квитанции контроллера о выполнении стирания/записи/верификации.
В принципе пользователь может вручную заносить коды по необходимым адресам FLASH памяти, пользуясь командами MODBUS 76h(чтение FLASH) и 77h(запись FLASH). Для этого он может использовать программу терминала MODBUS, программа так же доступна на нашем сайте www.fractal.com.ru.
Все эти операции можно проводить в сети MODBUS с удаленными узлами.

Второй способ - запись FLASH по каналу I2C. При этом способе необходим мастер - I2C, который сформирует необходимые последовательности по шине. Подробно это описано в разделе «Чтение / запись / стирание EEPROM и FLASH в режиме Slave устройства на шине I2C». Если мастер-I2C тоже узел/модуль с каналом MODBUS и этой прошивкой, то достаточно воспользоваться вышеупомянутой оболочкой.

Третий способ - это использование программатора способного программировать PIC18Fxxxx.

Возврат к заводским установкам

Если у пользователя возникла необходимость восстановить значения важных параметров сохраненных в EEPROM по умолчанию, то он может сделать это следующим образом:
Выключить и включить питание контроллера, и в течении минуты произвести не менее 10 горячих рестартов путем замыкания на землю вывода MCLR микроконтроллера. На всех узлах и модулях нашего производства эта линия обязательно присутствует на одном из разъемов. При этом на этом же разъеме обязательно присутствует и земля.
После этого в ячейки EEPROM с 0F6h по 0FFh будут занесены значения по умолчанию:

Адрес MODBUS1 => 02h;
Адрес MODBUS2 => 04h;
Скорость MODBUS1 => 115200;
Скорость MODBUS2 => 115200;
Адрес I2C => 010h;
Старший пользовательский вектор => 00h;
Младший пользовательский вектор => 00h.

!!! Обратите внимание, что эта процедура может быть запрещена самим пользователем, взведением бита REG_USER_2<0>. Также, восстановление может не произойти, если пользовательская программа по каким-то причинам меняет содержимое служебной ячейки, используемой для подсчета числа горячих рестартов.