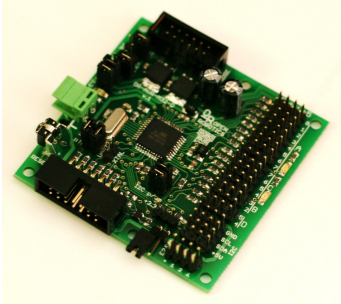
Контроллер общего назначения OR-AVR-M32-D

Версия контроллера 1.00, версия документа 1.00.Б

Инструкция по эксплуатации



Контроллер общего назначения OR-AVR-M32-D предназначен для управления устройствами (сервоприводами, ИК-дальномерами, контактными бамперами и т.д.) мобильного робота колёсного или гусеничного типа на коллекторных двигателях, может выступать как в качестве головного, так и в качестве вспомогательного контроллера.

Описание устройства

Основой контроллера является МК AVR ATMega32. Стабилизация питания осуществляется двумя lowdrop-стабилизаторами и LC-фильтрами. Для подключения различных устройств используются порты RoboGPIO и RoboI2C, порт драйвера двигателей RoobMD2 и разъем RoboBus.

Защита портов GPIO и RoboBus осуществляется токоограничительными резисторами. Для целей отладки и индикации могут быть использованы 2 светодиода. Также на плате установлен светодиод — индикатор питания и кнопка RESET аппаратного сброса.

Варианты использования контроллера:

Контроллер OR-AVR-M32-D может рассматриваться не только как платформа для запуска своих программ, но и как законченное решение для управления подключаемыми к нему устройствами при использовании специальной прошивки, взаимодействующей с головным устройством по шине RoboBus (по линиям UART или I2C). Подробнее о программировании контроллера можно прочитать в инструкции к программатору. Подробнее об использовании готовой прошивки для управления устройствами через этот контроллер по протоколам UART или I2C можно прочитать в документации к прошивке.

Основные характеристики:

Микроконтроллер:	AVR ATMega32 @ 7.3728 МГц *(FLASH: 32 Кб, RAM: 2 Кб, EEPROM: 1 Кб)*
Напряжение питания:	6-16 В
Габариты модуля:	66 x 66 x 16 мм
Порты RoboGPIO:	16 *(из них 8 с функцией АЦП)*
Порты RoboI2C:	4
Допустимая нагрузка*:	0.8 А по линии 5.0 В, 0.8 А по линии 3.3 В.

Подключение внешних устройств:

Правила подключения к портам GPIO описаны в документе «RoboGPIO: Инструкция пользователя».
Правила работы с шиной RoboBus описаны в документе «RoboBus: Инструкция пользователя».

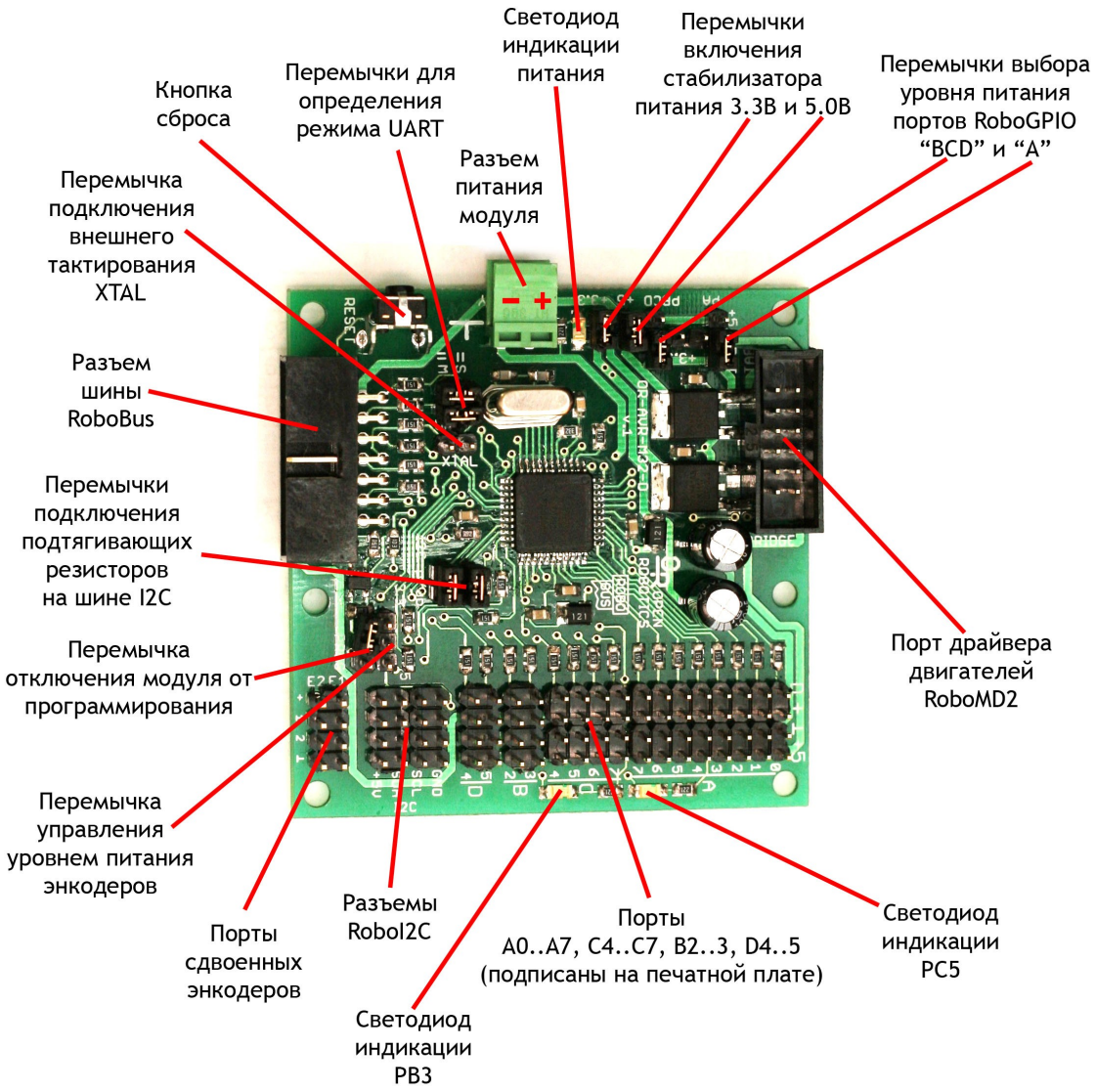
Дополнительная информация:

Главная страница проекта OpenRobotics: http://www.roboforum.ru/wiki/OpenRobotics
(Там же можно найти примеры применения контроллера и других модулей проекта)

Страница поддержки контроллера: http://roboforum.ru/viewtopic.php?f=69&t=5543

Примечание* *Допустимая нагрузка на линии питания модуля при использовании встроенных стабилизаторов.*
При питании VSS больше 6 вольт максимальная нагрузка по линии X вольт (X=3.3 или 5.0) не более (VSS-X)/2 ампер, но не выше 0.8 ампер.

Расположение и назначение разъемов, перемычек и светодиодов:



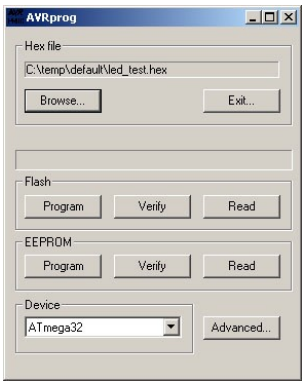
Загрузка .hex-прошивок с помощью программатора AVR910

Для загрузки готовых прошивок из прошивок в контроллер вам потребуется программатор. Можно использовать, например, AVR910 с адаптером под шину RoboBus.

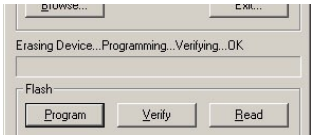
Сначала нужно установить ПО для загрузки прошивок, будем использовать AVRProg от фирмы ATMEL, который можно найти на сайте проекта OpenRobotics в разделе «Общие файлы».

После установки ПО подключите программатор через COM-порт к ПК, а через разъём RoboBus к контроллеру. Подайте питание на контроллер (программатор питается от контроллера), у вас должны загореться индикатор питания на контроллере и светодиод статуса на программаторе (сначала он горит красным, а при завершении загрузки - зелёным).

Запустите AVRProg, если она нашла ваш программатор, вы увидите вот такое окно (иначе см. раздел «Устранение неисправностей» внизу страницы):



В качестве файла через кнопку «Browse...» выберите тот .hex-файл который вы хотите загрузить (попробовать можно на файле bc_led_m32.hex, доступном на сайте OpenRobotics в разделе «Общие файлы»), после чего нажмите кнопку «Program» в секции «Flash». По окончании программирования у вас должна появиться надпись «Programing... Verifying... OK» (иначе см. раздел «Устранение неисправностей»):



Если всё сделано правильно, то мигают светодиоды подключенные к портам D6 и D7. Поздравляем!

Устранение неисправностей

Неисправность	Возможная причина	Способ устранения
При запуске AVRProg появляется окно «No supported board found!» и после нажатия «OK» программа закрывается.	Программатор не подключен к ПК или какой-то разъем воткнут не полностью.	Подключите программатор к ПК или подключите полностью соответствующий разъем.
	Программатор подключился к COM порту с номером >4	Перенастройте COM-порты так, чтобы программатор был на одном из портов — COM1-4
После программирования появляется сообщение «Verify... FAILED»	Не полностью воткнутый разъем RoboBus	Воткните соответствующий разъем полностью.
	Прошивка предназначена для другого контроллера	Выберите другую прошивку, подходящую для этого контроллера или используйте соответствующий контроллер.
	На шине RoboBus размещены другие модули которые мешают программированию нужного вам модуля	У всех других модулей выставьте перемычку RESET в состояние ISOLATED, если какие-то модули используют протокол SPI, тогда придётся у них выставить на время программирования эту перемычку в режим GND-LOCKED, а после вернуть на место. Либо можно просто снять с шины RoboBus мешающие модули.
	На программируемом контроллере не выставлена перемычка RESET	Выставить перемычку RESET в режим ROBOBUS.

Использование прошивки шлюз-контроллера для порта UART

Для управления через UART порт контроллера подключенными к нему устройствами разработана специальная готовая прошивка, которую можно загрузить в контроллер. Скачать прошивку можно на сайте проекта OpenRobotics в разделе «Готовые модули» \ «OR-AVR-M32-D».

В том числе реально вообще не заниматься программированием МК, а просто управлять роботом прямо с ПК или ноутбука или КПК / сотового телефона (через любой адаптер UART-порта, как то USB<=>RoboBus адаптер или Bluetooth-адаптер или напрямую через UART-порт, если его уровни соответствуют напряжению 3.3 В).

Формат команд

При работе через UART-порт используется формат команд на 100% совместимый с идеологией протокола i2c — это сделано для двух целей:

1. Простота адаптации полученной прошивки к получению команд по протоколу i2c, а значит можно будет на один UART-порт повесить сколько угодно однотипных модулей контроллера и управлять ими единым способом.
2. Простота передачи и обработки i2c запросов, которые могут быть отправлены любым i2c-устройствам на шине RoboBus, подключенным к шлюз-контроллеру.

При обмене данными головного устройства и шлюз-контроллера, головное устройство считается управляющим, а шлюз-контроллер управляемым устройством. Единственное сообщение отсылаемое по инициативе шлюз-контроллера - сообщение `"Ready!\n"` о готовности выполнять команды при включении, все остальные сообщения шлюз-контроллера являются ответами на команды.

Команды, отдаваемые шлюз-контроллеру через UART-порт все имеют одну и ту же форму **Q**`{AA}{RR}[{WW}*]`, в которой **{AA}** - адрес, **{RR}** - сколько байт хотим получить обратно, **{WW}*** - отсылаемые нами байты.

Формат ответа: **R**[{EE}*], где **{EE}*** - байты ответа, которые мы запрашивали. При ошибке возвращается **X**{EE}{Description}, где **{EE}** - код ошибки, а **{Description}** - её текстовое описание.

Номера портов

Порядок номеров портов 0x00..0x0F, использованный в прошивке шлюз-контроллера, совпадает с порядком разъемов GPIO на плате если считать от порта A0 до D4:

0x00..0x07 – порт A, линии 0..7

0x08..0x0B – порт C, линии 7..4

0x0C..0x0D – порт B, линии 2..3

0x0E..0x0F – порт D, линии 5..4

Список допустимых команд

Команда	Формат	Входные/выходные параметры
Отправить по i2c несколько байт и получить несколько байт в ответ	**Q**aarr{w}	• **aa** - адрес 00h..7Fh устройства на шине i2c • **rr** - сколько байт получить от устройства • **ww** - байты которые нужно передать устройству
Установить режим работы порта ввода-вывода	**QFF00**ppmm	• **pp** - номер порта 00h..0Fh увеличенный на 20h (например, для порта 0Fh это будет 2Fh) • **mm** - режим работы (0 - цифровой вход, 1 - цифровой выход, 2 - управление сервоприводом, 3 - аналоговый вход - последний режим будет работать только для портов в которых есть эта возможность)
Установить значение на выходе порта	**QFF00**ppvv	• **pp** - номер порта 00h..0Fh • **vv** - значение (для цифровых выходов - 0/1, для управления сервоприводом - 0x0D..0x53)
Получить значение со входа порта	**QFF01**pp	• **pp** - номер порта 00h..0Fh увеличенный на 80h (например, для порта 0Ch это будет 8Ch) обратно получим 1 байт - 0/1, если цифровой вход, либо 00h..FFh - если аналоговый (00h соответствует 0 В, FFh соответствует 3.3 В), либо если тип порта - выход - получим то, что туда отправляли.
Установить направление вращения двигателя	**QFF00**mmdd	• **mm** - номер двигателя 00h..01h умноженный на 2 и увеличенный на 10h (например, для двигателя 1 это будет 12h) • **dd** - направление вращения 00h - СТОП, 01h - ВПЕРЕД, 02h - НАЗАД
Установить скорость вращения двигателя	**QFF00**mmss	• **mm** - номер двигателя 00h..01h умноженный на 2 и увеличенный на 11h (например, для двигателя 1 это будет 13h) • **ss** - скорость от 00h до FFh, в точности соответствует скважности выдаваемого на двигатели ШИМ-сигнала.
	QFF0014[ttaabbccdd]*	• **tt** - время выполнения команды в 1/50 секунды • **aa** - начальная скорость+направление двигателя 0; • **bb** - начальная скорость+направление двигателя 1; • **cc** - конечная скорость+направление двигателя 0; • **dd** - конечная скорость+направление двигателя 1; Способ задания скорости с направлением - 0x00 - стоим, 0x01..0x7F - малый..полный вперед, 0xFF..0x80 - малый..полный назад. Максимально можно отправить одновременно 7 команд (иначе переполнится буфер приёма UART - легко расширить до 15-20 команд). Если нужно сбросить очередь команд - отправляем просто QFF0014 без команд.