Мотор-редуктор с управляющим контроллером, FLASH-I2C - Datasheet

Модуль - Мотор-редуктор с управляющим контроллером, I2C-flash.

Техническое описание: Данная страница содержит подробное техническое описание модуля - Мотор-редуктор с управляющим контроллером, I2C-flash и раскрывает работу с модулем через его регистры.
Ознакомиться с пользовательским описанием модуля и примерами работы с библиотекой iarduino_I2C_Motor можно на странице Wiki - Мотор-редуктор с управляющим контроллером, I2C-flash.

Назначение:

Модуль - Мотор-редуктор с управляющим контроллером, I2C-flash - является устройством состоящим из коллекторного двигателя с редуктором и платы управления, подключаемой к шине I2C.

К одной шине I2C можно подключить более 100 модулей. Адрес модуля на шине I2C (по умолчанию 0x09) назначается программно и хранится в его энергонезависимой памяти.

Описание:

Модуль построен на базе двигателя GM12-N20, редуктора, микроконтроллера STM32F030F4 и драйвера DRV8833, снабжен

многополюсным магнитным валом, датчиками Холла, и собственным стабилизатором напряжения. Модуль способен поддерживать заданную скорость и направление вращения вала, сверяясь с показаниями датчиков Холла. Модуль самостоятельно обрабатывает полученные данные сохраняя их в своих регистрах. Доступ к регистрам модуля осуществляется по шине I2C.

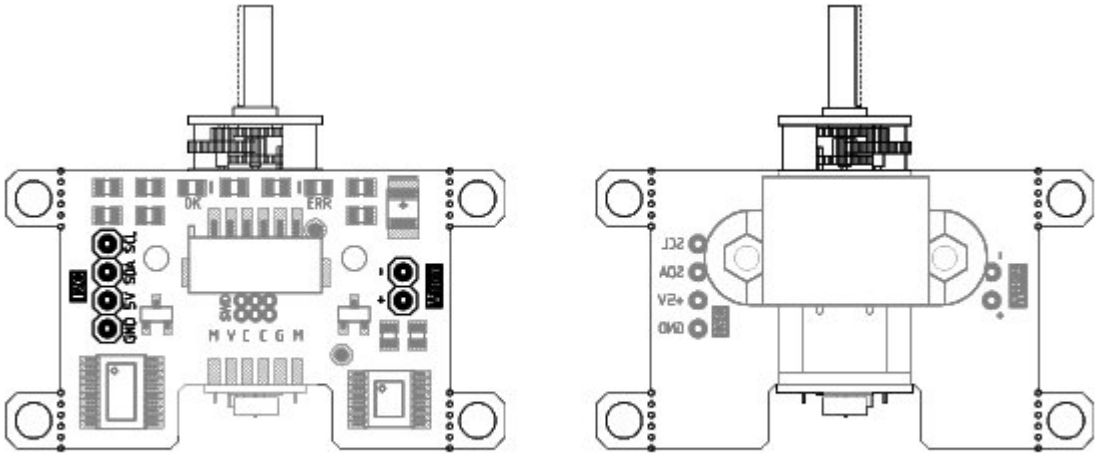
С помощью регистров модуля можно выполнять следующие действия:

- Изменить адрес данного модуля на шине I2C. При изменении адреса, можно указать, что новый адрес должен сохраниться в flash память модуля, а значит адрес сохранится и после отключения питания.
- Включить / отключить внутреннюю подтяжку линий шины I2C (по умолчанию включена). Состояние подтяжки линий шины I2C автоматически сохраняется в flash память модуля, а значит состояние линий сохранится после отключения и включения питания.
- Узнать версию прошивки модуля.
- Указать передаточное отношение редуктора мотора, от 0.01 до 167'772.15.
- Указать количество полюсов (одной полярности) магнитного вала, от 1 до 255.
- Указать наличие инверсии редуктора и тип подключения мотора.
- Задать направление вращения вала при положительной скорости, по/против ч.с.
- Задать скорость вращения вала указав коэффициент заполнения ШИМ, от 0 до ±4095.
  При отрицательных значениях, вал будет вращаться в противоположном направлении.
- Задать скорость вращения вала указав количество оборотов в минуту, от 0 до ±32'767.
  При отрицательных значениях, вал будет вращаться в противоположном направлении.
  Скорость будет поддерживаться модулем на основании показаний с датчиков Холла.
- Узнать каким образом задана скорость вращения вала (указанием ШИМ или количеством оборотов в минуту).
- Задать максимально допустимый % (от 0 до 100) отклонения реальной скорости от заданной.
  На плате модуля есть красный светодиод, который сигнализирует о том, что заданная скорость вращения вала отличается от реальной больше чем на указанный процент.
- Узнать отличается ли заданная скорость вращения вала от реальной больше допустимого %.
- Узнать текущую скорость вращения вала, от 0 до ±32'767 об.мин. (RPM).
  Скорость вращения вала определяется по показаниям с датчиков Холла, даже если мотор отключён, а вал вращается по средством внешних сил.
- Узнать количество совершённых полных оборотов вала, от 0.00 до 167'772.15, значение является беззнаковым, оно учитывает обороты,

как в прямом, так и в обратном направлении. Регистр обнуляется при переполнении.
Количество оборотов вала определяется по показаниям с датчиков Холла, даже если мотор отключён, а вал вращается по средством внешних сил.

- Остановить двигатель и/или указать тип его остановки. Двигатель может быть остановлен двумя способами: отключением мотора (свободный ход) или торможением (стопор).
  Заданный тип применяется ко всем последующим остановкам двигателя.
- Остановить двигатель через заданное количество полных оборотов вала, от 0 до 167'772.15, значение является беззнаковым, оно учитывает обороты, как в прямом, так и в обратном направлении. Значение регистра уменьшается по мере вращения вала, при достижении 0 мотор останавливается.
- Остановить двигатель через заданное количество миллисекунд, от 0 до 16'777'215. Значение регистра уменьшается с каждой миллисекундой, при достижении 0 мотор останавливается.
- Узнать остановлен ли мотор.
- Узнать есть ли у остановленного вала возможность свободного хода, или он застопорен.
- Узнать о наличии ошибки драйвера (перегрузка по току, перегрев, низкое напряжение).

Выводы модуля:



Модуль содержит 2 разъема расположенные по бокам платы.

С одной стороны платы расположен разъем **I2C** состоящий из 4 выводов, для подключения модуля к шине I2C:

- **SCL** - вход/выход линии тактирования шины I2C.
- **SDA** - вход/выход линии данных шины I2C.
- **5V** - вход питания логической части драйвера +5 В (номинально), или 3,3 В.
- **GND** - общий вывод питания.

С другой стороны расположен разъём $\mathbf{V_{МОТ}}$ состоящий из 2 выводов, для подключения питания мотора.

- $\mathbf{+V_{МОТ}}$ - вход питания мотора от +2,7 до +12 В постоянного тока.
- $\mathbf{-V_{МОТ}}$ - общий вывод питания мотора, внутрисхемно соединён с выводом GND шины I2C.

Характеристики:

- Напряжение питания логики: 5 В (номинально), или 3,3 В.
- Диапазон напряжений мотора поддерживаемый драйвером: 2,7 В ... 12 В.
- Максимальный ток мотора поддерживаемый драйвером: до 3 А (пиковый ток до 4 А).
- Драйвер оснащён защитой от перегрева и перегрузки по току.
- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: Vcc.
- Рабочая температура: от -20 до +70 °C.
- Габариты: 45 x 40 мм.
- Вес: 32 г.

Установка адреса:

Модуль - Мотор-редуктор с управляющим контроллером, I2C-flash относится к серии «Flash» модулей. Все модули данной серии позволяют назначать себе адрес для шины I2C, как временно (новый адрес действует пока есть питание), так и постоянно (новый адрес сохраняется в

энергонезависимую память и действует даже после отключения питания). По умолчанию все модули серии «Flash» поставляются с адресом 0x09.

Допускается указывать адреса в диапазоне: 7 < адрес < 127.

Установка адреса (без сохранения):

Если в регистр 0x06 «ADDRESS» записать значение из 7 бит адреса и младшим битом «SAVE_FLASH» равным 0, то указанный адрес станет адресом модуля на шине I2C, но он не сохранится во FLASH памяти, а значит после отключения питания или перезагрузки, установится прежний адрес модуля.

Установка адреса может быть заблокирована, если в регистре 0x01 «BITS_0» установлен бит «BLOCK_ADR». Этот бит по умолчанию сброшен, но он самостоятельно устанавливается при попытке записи данных в регистры предназначенные только для чтения. Бит «BLOCK_ADR» используется в модулях версии 5 и выше. Версия модуля хранится в регистре 0x05 «VERSION».

Установка адреса (с сохранением):

Для установки адреса с его сохранением в FLASH память модуля необходимо выполнить два действия:

- Установить бит «SAVE_ADR_EN» в регистре 0x01 «BITS_0» (при этом адрес модуля останется прежним).
- Записать в регистр 0x06 «ADDRESS» значение из 7 бит адреса и младшим битом «SAVE_FLASH» равным 1.

Если не выполнить первое действие (не установить бит «SAVE_ADR_EN»), то новый адрес будет проигнорирован и у модуля останется старый адрес. Бит «SAVE_ADR_EN» самостоятельно сбрасывается после сохранения адреса во FLASH память, а так же при обращении к любому регистру модуля (кроме записи в 0x01 «BITS_0» и 0x06 «ADDRESS»).

Установка адреса может быть заблокирована, если в регистре 0x01 «BITS_0» установлен бит «BLOCK_ADR». Этот бит по умолчанию сброшен, но он самостоятельно устанавливается при попытке записи данных в регистры предназначенные только для чтения. Бит «BLOCK_ADR» используется в модулях версии 5 и выше. Версия модуля хранится в регистре 0x05 «VERSION».

ВАЖНО: запись адреса занимает не менее 30 мс.

Регистры:

Карта регистров модуля:

адрес	7	6	5	4	3	2	1	0
0x00	FLG_RESET	FLG_SELF_TEST	-	-	-	FLG_I2C_UP	-	-
0x01	SET_RESET	SET_SELF_TEST	-	-	BLOCK_ADR	SET_I2C_UP	SAVE_ADR_EN	-
0x02 0x03	RESERVED							
0x04	MODEL[7-0]							
0x05	VERSION[7-0]							
0x06	ADDRESS[6-0]							SAVE_FLASH
0x07	CHIP_ID[7-0]							
0x08 0x09	FREQUENCY[7-0] FREQUENCY[15-8]							
0x0A	MAX_RPM_DEV[7-0]							
0x0B --- 0x0F	RESERVED							
адрес	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0x10	FLG_RPM_EN	-	FLG_RPM_ERR	FLG_DRV_ERR	-	-	FLG_STOP	FLG_NEUTRAL
0x11	MAGNET[7-0]							
0x12 0x13 0x14	REDUCER[7-0] REDUCER[15-8] REDUCER[23-16]							
0x15 0x16	SET_PWM[7-0] SET_PWM[15-8]							
0x17 0x18	SET_RPM[7-0] SET_RPM[15-8]							
0x19 0x1A	GET_RPM[7-0] GET_RPM[15-8]							
0x1B 0x1C 0x1D	GET_REV[7-0] GET_REV[15-8] GET_REV[23-16]							

адрес	7	6	5	4	3	2	1	0
0x1E 0x1F 0x20	STOP_REV[7-0] STOP_REV[15-8] STOP_REV[23-16]							
0x21 0x22 0x23	STOP_TMR[7-0] STOP_TMR[15-8] STOP_TMR[23-16]							
0x24	-	-	-	-	-	-	BIT_STOP	BIT_NEUTRAL
0x25	-	-	-	-	-	BIT_DIR_CKW	BIT_INV_RDR	BIT_INV_PIN
0x26	VOLTAGE[7-0]							
0x27 0x28	NOMINAL_RPM[7-0] NOMINAL_RPM[15-8]							

Регистры с адресами 0x02, 0x03 зарезервированы, их биты сброшены в 0. Попытка записи данных в эти регистры будет проигнорирована модулем.

Регистры с адресами 0x0B - 0x0F используются для программирования модуля изготовителем, не пытайтесь записывать данные в эти регистры.

Регистр 0x00 «FLAGS_0» - содержит флаги чтения состояния модуля:

Регистр только для чтения.

- **FLG_RESET** - Флаг указывает на факт выполнения успешной перезагрузки модуля. Флаг самостоятельно сбрасывается после чтения регистра 0x00 «FLAGS_0».
- **FLG_SELF_TEST** - Флаг указывает на результат выполнения самотестирования модуля (0-провал, 1-успех). Не поддерживается данным модулем.
- **FLG_I2C_UP** - Флаг указывает на то, что модуль позволяет управлять подтяжкой линий шины I2C при помощи бита «SET_I2C_UP» регистра 0x01 «BITS_0».

Регистр 0x01 «BITS_0» - содержит биты установки состояния модуля:

Регистр для записи и чтения.

- **SET_RESET** - Бит запускает программную перезагрузку модуля. О завершении перезагрузки свидетельствует установка флага «FLG_RESET» регистра 0x00 «FLAGS_0».
- **SET_SELF_TEST** - Бит запускает самотестирование модуля. При успешном завершении самотестирования устанавливается флаг «FLG_SELF_TEST » регистра 0x00 «FLAGS_0». Не поддерживается данным модулем.
- **BLOCK_ADR** - Бит блокирует смену и сохранение адреса для шины I2C. Бит устанавливается автоматически при попытке записи данных в регистры предназначенные только для чтения. Это защищает чип от ненамеренной смены адреса шумами на шине I2C, бит используется в модулях версии 5 и выше. Версия модуля хранится в регистре 0x05 «VERSION».
- **SET_I2C_UP** - Бит управляет внутрисхемной подтяжкой линий шины I2C. Значение бита сохраняется в FLASH память модуля. Установка бита в «1» приведёт к подтяжке линий SDA и SCL до уровня 3,3 В. На линии I2C допускается устанавливать внешние подтягивающие резисторы и иные модули с подтяжкой до уровня 3,3 В или 5 В, вне зависимости от состояния текущего бита. Если флаг «FLG_I2C_UP» регистра 0x00 «FLAGS_0» сброшен, значит управление подтяжкой не поддерживается модулем.
- **SAVE_ADR_EN** - Бит разрешает записать новый адрес модуля для шины I2C в FLASH память. Бит самостоятельно сбрасывается после сохранения адреса во FLASH память. Запись адреса выполняется следующим образом: нужно установить бит «SAVE_ADR_EN», после чего записать новый адрес в регистр 0x06 «ADDRESS» с установленным битом «SAVE_FLASH».

Регистр 0x04 «MODEL» - содержит идентификатор типа модуля:

Регистр только для чтения.

- **MODEL[7-0]** - Для модуля - Мотор-редуктор с управляющим контроллером, I2C-flash - идентификатор равен 0x14.

Регистр 0x05 «VERSION» - содержит версию прошивки модуля:

Регистр только для чтения.

- **VERSION[7-0]** - Версия прошивки (от 0x01 до 0xFF).

Регистр 0x06 «ADDRESS» - отвечает за чтение/установку адреса модуля на шине I2C:

Регистр для чтения и записи.

- **ADDRESS[6-0]** - 7 бит адреса модуля на шине I2C. При чтении возвращается текущий адрес модуля, при записи устанавливается

указанный адрес модулю. Допускается указывать адреса в диапазоне: 7 < адрес < 127.

- **SAVE_FLASH** - Флаг записи адреса в FLASH память модуля.
  Флаг имеет значение только при записи данных в регистр.
  Если флаг сброшен, то адрес в битах ADDRESS[6-0] будет установлен временно (до отключения питания, или сброса/записи нового адреса). Если флаг установлен, то адрес в битах ADDRESS[6-0] будет сохранён в FLASH память модуля (останется и после отключения питания), но только если в бите «SAVE_ADR_EN» регистра 0x01 «BITS_0» установлена логическая 1. Если флаг «SAVE_FLASH» установлен, а бит «SAVE_ADR_EN» сброшен, то адрес в битах ADDRESS[6-0] не будет установлен ни временно, ни постоянно.

Регистр 0x07 «CHIP_ID» - содержит идентификатор общий для всех модулей серии «Flash»:

Регистр только для чтения.

У всех модулей серии «Flash» в регистре «CHIP_ID» содержится значение 0x3C. Если требуется отличить модули серии «Flash» на шине I2C от сторонних модулей, то достаточно прочитать значение регистров 0x06 «ADDRESS» и 0x07 «CHIP_ID» всех модулей на шине I2C. Если 7 старших битов регистра 0x06 «ADDRESS» хранят адрес совпадающий с адресом модуля, а в регистре 0x07 «CHIP_ID» хранится значение 0x3C, то можно с большой долей вероятности утверждать, что данный модуль является модулем серии «Flash».

Регистры 0x08-0x09 «FREQUENCY» - содержат частоту ШИМ:

Регистры только для записи.

- **FREQUENCY[15-0]** - Частота ШИМ в диапазоне от 25 до 1'000 Гц. Значение по умолчанию 0x01F4 = 500 Гц. Частота записанная в регистры «FREQUENCY[15-8]», «FREQUENCY[7-0]» применяется после записи старшего байта «FREQUENCY[15-8]».
  Если записать значение < 0x0019 то в регистрах появится 0x0010 и частота будет 25 Гц.
  Если записать значение > 0x03E8 то в регистрах появится 0x03E8 и частота будет 1 кГц.
- Частота ШИМ не влияет на коэффициент заполнения ШИМ, а скорость мотора зависит от коэффициента заполнения ШИМ.
- Примечание: У модулей версии 6 и ниже, частота указывается в диапазоне от 1 (а не от 25) и записанное значение сохраняется даже после отключения питания.

Регистр 0x0A «MAX_RPM_DEV» - максимальный % отклонения скорости:

Регистр только для записи.

- **MAX_RPM_DEV[8-0]** - Число от 0 до 100 задаёт максимально допустимый процент отклонения реальной скорости от заданной, при котором устанавливается флаг ошибки скорости.
  Значение регистра не влияет на скорость, а определяет границу установки флага ошибки.
  Значение по умолчанию 0x0A = 10%.
  Если записать значение больше 100% (больше 0x64) то в регистре появится 0x64 = 100%.
- Если скорость вращения вала отличается от заданной регистрами 0x17-0x18 «SET_RPM» более чем на указанный процент, то в регистре 0x10 «FLG» устанавливается флаг ошибки скорости «FLG_RPM_ERR», а на плате модуля включается красный светодиод.
- Снижение отличия заданной и реальной скорости ниже указанного процента, приводит к сбросу флага «FLG_RPM_ERR» и отключению красного светодиода на плате модуля.
- Примечание: У модулей версии 6 и ниже, записанное значение сохраняется даже после отключения питания.

Регистр 0x10 «FLG» - содержит статусные флаги:

Регистр только для чтения.

- **FLG_RPM_EN** - Флаг указывает на то, что скорость вращения вала задана количеством оборотов в минуту 0x17-0x18 «SET_RPM». Если флаг сброшен, значит скорость вращения вала задана коэффициентом заполнения ШИМ 0x15-0x16 «SET_PWM».
- **FLG_RPM_ERR** - Флаг указывает на ошибку скорости. Флаг устанавливается если скорость вращения вала задана количеством оборотов в минуту 0x17-0x18 «SET_RPM» и эта скорость отличается от реальной скорости вращения вала более чем на 0x0A «MAX_RPM_DEV» %.
- **FLG_DRV_ERR** - Флаг указывает на ошибку драйвера. Флаг устанавливается при перегрузке по току мотора, перегреве драйвера или низком уровне напряжения питания мотора.
- **FLG_STOP** - Флаг информирует об остановке вала. Флаг устанавливается при установке бита «BIT_STOP» регистра 0x24 «STOP», при задании нулевой скорости, или при установке коэффициента заполнения ШИМ равного нулю.
- **FLG_NEUTRAL** - Флаг информирует о наличии свободного хода у остановленного вала.
  Флаг устанавливается при установке бита «BIT_NEUTRAL» регистра 0x24 «STOP».
  Если флаг установлен значит при остановке мотора, вал можно вращать.
  Если флаг сброшен, значит при остановке осуществляется торможение (вал застопорен).

Регистр 0x11 «MAGNET» - содержит количество полюсов кольцевого магнита:

Регистр для чтения и записи.

- **MAGNET[7-0]** - Количество полюсов магнитного вала (одной полярности), значение от 1 до 255. Данное значение можно изменить при смене магнитного вала модуля, который оказывает влияние на показания датчиков Холла.
- Запись нулевого значения означает отсутствие кольцевого магнита на роторе мотора или отсутствие датчиков Холла. При этом перестанут работать следующие регистры:
  - 0x17-0x18 «SET_RPM» - указание скорости через обороты в минуту.
  - 0x19-0x1A «GET_RPM» - получение скорости через обороты в минуту.
  - 0x1B-0x1D «GET_REV» - получение количества совершённых полных оборотов вала.
  - 0x1E-0x20 «STOP_REV» - количество оборотов вала оставшихся до остановки мотора.

Регистры 0x12-0x14 «REDUCER» - содержат передаточное отношение редуктора:

Регистры для чтения и записи.

- **REDUCER[23-0]** - Передаточное отношение редуктора в сотых долях, от 1 (что соответствует отношению 1:0.01) до 16'777'215 (что соответствует отношению 1:167'772.15). Передаточное отношение записанное в регистры «REDUCER[23-16]», «REDUCER[15-8]», «REDUCER[7-0]» применяется после записи старшего байта «REDUCER[23-16]». Данное значение можно изменить при смене редуктора мотора, на редуктор с другим передаточным отношением.
  Если записать значение 0x000000 то в регистрах появится 0x000064 (=100), что соответствует передаточному отношению 1:1.00 (редуктор отсутствует).
  Если записать значение от 0x000001 до 0x000064, значит редуктор повышающий.

Регистры 0x15-0x16 «SET_PWM» - задают скорость через коэффициент ШИМ:

Регистры для чтения и записи.

- **SET_PWM[15-0]** - Коэффициент заполнения ШИМ от 0 до ±4095. Коэффициент определяет скорость вращения вала, где 0 - остановка, а ±4095 максимальная скорость.
  При отрицательных значениях, вал будет вращаться в противоположном направлении.
  Коэффициент заполнения ШИМ записанный в регистры «SET_PWM[15-8]», «SET_PWM[7-0]» применяется после записи старшего байта «SET_PWM[15-8]».
- Запись любого значения приводит к обнулению регистров задания скорости количеством оборотов в минуту 0x17-0x18 «SET_RPM» и

сбросу флагов «FLG_RPM_EN», «FLG_RPM_ERR» регистра 0x10 «FLG».
Если записать значение выходящее за пределы диапазона ±4095 то в регистрах появится значение 4095 с сохранением записанного знака.

- Если записать значение 0, то будет установлен бит «BIT_STOP» регистра 0x24 «STOP», что приведёт к остановке вала и установке флага «FLG_STOP» регистра 0x10 «FLG».

Регистры 0x17-0x18 «SET_RPM» - задают скорость через обороты в минуту:

Регистры для чтения и записи.

- **SET_RPM[15-0]** - Скорость вращения вала заданная количеством оборотов в минуту, от 0 до ±32'767 об.мин. (RPM). При отрицательных значениях, вал будет вращаться в противоположном направлении.
  Скорость записанная в регистры «SET_RPM[15-8]», «SET_RPM[7-0]» применяется после записи старшего байта «SET_RPM[15-8]».
- Запись не нулевого значения приводит к установке флага «FLG_RPM_EN» регистра 0x10 «FLG», указывающего что скорость задана количеством оборотов в минуту.
- Заданная скорость поддерживается модулем на основании показаний с датчиков Холла.
  Если заданная скорость вращения вала отличается от реальной скорости вращения вала более чем на 0x0A «MAX_RPM_DEV» процентов, то в регистре 0x10 «FLG» устанавливается флаг ошибки скорости «FLG_RPM_ERR», а на плате модуля включается красный светодиод.
  В противном случае флаг «FLG_RPM_ERR» регистра 0x10 «FLG» сбрасывается, а красный светодиод на плате модуля отключается.
- Если записать значение 0, то будет сброшен флаг «FLG_RPM_EN» регистра 0x10 «FLG» и установлен бит «BIT_STOP» регистра 0x24 «STOP», что приведёт к остановке вала и установке флага «FLG_STOP» регистра 0x10 «FLG».

Регистры 0x19-0x1A «GET_RPM» - содержат количество оборотов в минуту:

Регистры только для чтения.

- **GET_RPM[15-0]** - Реальное количество оборотов вала в минуту, от 0 до ±32'767 об.мин. (RPM).
  Скорость вращения вала определяется по показаниям с датчиков Холла, даже если мотор отключён, а вал вращается по средством внешних сил. Отрицательные значения означают что вал вращается в противоположном направлении.

Регистры 0x1B-0x1D «GET_REV» - содержат количество совершённых оборотов:

Регистры только для чтения.

- **GET_REV[23-0]** - Количество совершённых оборотов вала в сотых долях полного оборота, от 0 (0.00 оборотов) до 16'777'215 (167'772,15 полных оборотов).
  Значение является беззнаковым, оно учитывает обороты, как в прямом, так и в обратном направлении (складывая их без учёта направления).
  Количество оборотов вала определяется по показаниям с датчиков Холла, даже если мотор отключён, а вал вращается по средством внешних сил.
- Обнулить количество совершённых оборотов (0x1B-0x1D «GET_REV») можно записав любое значение в регистры 0x1E-0x1F «STOP_REV».

Регистры 0x1E-0x20 «STOP_REV» - содержат количество оборотов до остановки:

Регистры для чтения и записи.

- **STOP_REV[23-0]** - Количество полных оборотов до остановки вала. Значение хранится в сотых долях полного оборота, от 0 (0.00 оборотов) до 16'777'215 (167'772,15 полных оборотов).
  Значение является беззнаковым, оно учитывает обороты, как в прямом, так и в обратном направлении.
  Количество полных оборотов записанное в регистры «STOP_REV[23-16]», «STOP_REV[15-8]», «STOP_REV[7-0]» применяется после записи старшего байта «STOP_REV[23-16]».
- Запись не нулевого значения включает отсчёт оборотов до остановки вала. При вращении вала, модуль уменьшает значение регистров (0x1E-0x20 «STOP_REV») на совершённое количество оборотов, а при достижении 0 устанавливает бит «BIT_STOP» регистра 0x24 «STOP», что приводит к остановке вала и установке флага «FLG_STOP» регистра 0x10 «FLG».
- Запись нулевого значения отключает отсчёт оборотов без установки бита «BIT_STOP» регистра 0x24 «STOP», что не приводит к остановке вала.
- Запись любого значения (в том числе и нулевого) приводит к обнулению регистров количества совершённых оборотов 0x1B-0x1D «GET_REV».
- Количество оборотов вала определяется по показаниям с датчиков Холла, даже если мотор отключён, а вал вращается по средством внешних сил.
- Записать новое значение в регистры (или переписать их) можно в любое время, как при вращающемся, так и при остановленном вале.
  Если модуль управляет колёсами, то зная их радиус, можно определить количество полных оборотов требуемое для движения на

заданное расстояние.
Если записать количество оборотов, при остановленном вале, а потом запустить вал (указав скорость), вы будите точно знать путь пройденный с момента запуска вала до его остановки.
Если записать количество оборотов при вращающемся вале, вы будите точно задать путь пройденный с момента записи количества оборотов до остановки вала.

Регистры 0x21-0x23 «STOP_TMR» - содержат время до остановки вала:

Регистры для чтения и записи.

- **STOP_TMR[23-0]** - Количество миллисекунд до остановки вала, от 0 до 16'777'215 (16'777,215 сек).
  Время записанное в регистры «STOP_TMR[23-16]», «STOP_TMR[15-8]», «STOP_TMR[7-0]» применяется после записи старшего байта «STOP_TMR[23-16]».
- Запись не нулевого значения включает отсчёт времени до остановки вала (даже если вал не вращается). Модуль уменьшает значение регистров (0x21-0x23 «STOP_TMR») на количество пройденных миллисекунд, а при достижении 0 устанавливает бит «BIT_STOP» регистра 0x24 «STOP», что приводит к остановке вала и установке флага «FLG_STOP» регистра 0x10 «FLG».
- Запись нулевого значения отключает отсчёт времени без установки бита «BIT_STOP» регистра 0x24 «STOP», что не приводит к остановке вала.
- Записать новое значение в регистры (или переписать их) можно в любое время, как при вращающемся, так и при остановленном вале.
  Если модуль управляет колёсами, то задав скорость оборотами в секунду, можно определить время требуемое для движения на заданное расстояние.

Регистр 0x24 «STOP» - содержит биты остановки вала:

Регистр только для записи.

- **BIT_STOP** - бит остановки вала. Сброс бита в 0 не приводит ни к каким действиям. Установка бита приводит к остановке вала, сбросу регистров задания скорости 0x15-0x16 «SET_PWM» и 0x17-0x18 «SET_RPM», сбросу флагов «FLG_RPM_EN» и «FLG_RPM_ERR» регистра 0x10 «FLG» и установке флага «FLG_STOP».
- **BIT_NEUTRAL** - бит определяет поведение вала при остановке. Изменение бита «BIT_NEUTRAL» приводит к аналогичному изменению флага «FLG_NEUTRAL» регистра 0x10 «FLG».

Если флаг установлен значит при остановке мотора, вал можно вращать.
Если флаг сброшен, значит при остановке осуществляется торможение (вал застопорен).

- Торможение (стопор) вала осуществляется путём замыкания обмоток двигателя, что приводит к сопротивлению поворота вала внешними силами.

Регистр 0x25 «BITS_2» - содержит биты инверсии направления вращения:

Регистр для чтения и записи.

- **BIT_DIR_CKW** - Бит вращения вала модуля по часовой стрелке, при положительной скорости. Данный бит позволяет установить два модуля по бокам устройства, например, по левому и правому борту машины. Для левого модуля бит сбрасывается, а для правого устанавливается. При указании положительной скорости для обоих модулей, колёса установленные на их валы будут вращаться в направлении по ходу машины. А при указании отрицательной скорости, оба колеса будут вращаться в обратном направлении. При этом не важно как указывалась скорость, при помощи ШИМ или указанием количества оборотов в минуту.
  По умолчанию бит установлен в 1.
- **BIT_INV_RDR** - Бит инверсии редуктора. Бит указывает на то, что вал редуктора вращается в сторону противоположную вращению ротора мотора.
  Значение по умолчанию зависит от используемого в модуле редуктора.
- **BIT_INV_PIN** - Бит инверсии подключения проводов мотора. Бит указывает на то, что ротор мотора вращается против часовой стрелки.
  Значение по умолчанию зависит от типа подключения мотора в модуле.

Регистр 0x26 «VOLTAGE» - содержит номинальное напряжение электродвигателя:

Регистр для чтения и записи.

- **VOLTAGE[7-0]** - Номинальное напряжение электродвигателя. Значение хранится в десятых долях Вольт, от 0 (0.0 В) до 255 (25,5 В).
  Значение является беззнаковым, оно носит исключительно информационный характер.

Регистры 0x27-0x28 «NOMINAL_RPM» - содержат номинальное количество оборотов в минуту:

Регистры для чтения и записи.

- **NOMINAL_RPM[15-0]** - Номинальное количество оборотов вала, от 0 до 65'535 об.мин. (RPM).

Значение является беззнаковым целым числом. Оно носит исключительно информационный характер, о скорости вращения вала редуктора при номинальном напряжении мотора и 100% коэффициенте заполнения ШИМ.

Доступ к данным регистров:

Каждый регистр модуля хранит 1 байт данных. Так как модуль использует интерфейс передачи данных I2C, то и доступ к данным охарактеризован им.

Обмен данными по шине I2C происходит по одному биту за один такт, после каждых переданных 8 бит (1 байта) принимающее устройство отвечает передающему одним битом: «ACK» в случае успешного приёма, или «NACK» в случае ошибки. Пакет приёма/передачи данных начинается сигналом «START» и завершается сигналом «STOP». Первый байт пакета всегда состоит из 7 бит адреса устройства и одного (младшего) бита R/W.

Сигналы интерфейса передачи данных I2C:

Для удобства восприятия сигналов они выполнены в следующих цветах:

- Зелёный - сигналы формируемые мастером.
- Красный - данные отправляемые мастером.
- Синий - данные отправляемые модулем.
- Фиолетовый - данные отправляемые мастером или модулем.

- «START» - отправляется мастером в начале пакета приема/передачи данных. Сигнал представляет переход уровня линии «SDA» из «1» в «0» при наличии «1» на линии «SCL».
- «STOP» - отправляется мастером в конце пакета приёма/передачи данных. Сигнал представляет переход уровня линии «SDA» из «0» в «1» при наличии «1» на линии «SCL».
- БИТ - значение бита считывается с линии «SDA» по фронту импульса на линии «SCL».
- «ACK» - бит равный 0, отправляется после успешного приёма байта данных.
- «NACK» - бит равный 1, отправляется после байта данных в случае ошибки.
- ПЕРВЫЙ БАЙТ - отправляется мастером, состоит из 7 бит адреса и бита «RW».
- «R/W» - младший бит первого байта данных указывает направление передачи данных пакета, 1 - прием (от модуля к мастеру), 0 - передача (от мастера в модуль).

- «RESTART» - повторный старт, отправляется мастером внутри пакета. Сигнал представляет из себя «START» отправленный не на свободной шине, а внутри пакета.

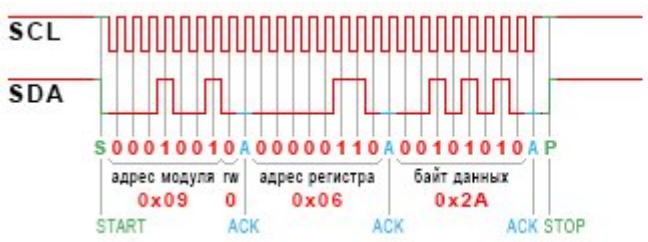
ВАЖНО: Все изменения на линии «SDA» должны происходить только при наличии «0» на линии «SCL» за исключением сигналов «START», «STOP» и «RESTART».

Запись данных в регистры:

- Отправляем сигнал «START».
- Отправляем первый байт: 7 бит адреса модуля и бит «R/W» равный 0 (запись).
  Получаем ответ от модуля в виде одного бита «ACK».
- Отправляем второй байт: адрес регистра в который будет произведена запись.
  Получаем ответ от модуля в виде одного бита «ACK».
- Отправляем третий байт: данные для записи в регистр.
  Получаем ответ от модуля в виде одного бита «ACK».
- Далее можно отправить четвёртый байт данных для записи в следующий по порядку регистр и т.д.
- Отправляем сигнал «STOP».

Пример записи в один регистр:

Запись значения 0x2A в регистр 0x06 модуля с адресом 0x09:



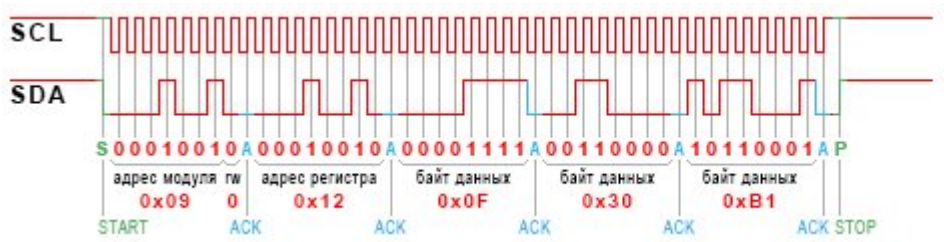

```
                              // Запись в регистр методами библиотеки Wire.h
Wire.beginTransmission(0x09);    // Инициируем передачу данных в устройство с адресом 0x09.
Wire.write(0x06);                // Записываем в буфер байт адреса регистра.
Wire.write(0x26);                // Записываем в буфер байт который будет записан в регистр.
```

```
Wire.endTransmission();          // Выполняем передачу адреса и байтов из буфера. Функция возвращает: 0-передача успешна / 1 -
```

Пример записи в несколько регистров подряд:

Запись в модуль с адресом 0x09 нескольких значений начиная с регистра 0x12:

В регистр 0x12 запишется значение 0x0F, в следующий по порядку регистр (0x13) запишется значение 0x30 и в следующий по порядку регистр (0x14) запишется значение 0xB1.




```
                                 // Запись в регистры методами библиотеки Wire.h
byte data[3] = {0x0F,0x30,0xB1}; // Определяем массив с данными для передачи.
Wire.beginTransmission(0x09);    // Инициируем передачу данных в устройство с адресом 0x09.
Wire.write(0x12);                // Записываем в буфер байт адреса первого регистра.
Wire.write(data, 3);             // Записываем в буфер 3 байта из массива data.
Wire.endTransmission();          // Выполняем передачу адреса и байт из буфера. Функция возвращает: 0-передача успешна / 1 - г
```

Чтение данных из регистров:

- При чтении пакет делится на 2 части: запись № регистра и чтение его данных.
- Отправляем сигнал «START».
- Отправляем первый байт: 7 бит адреса модуля и бит «R/W» равный 0 (запись).
  Получаем ответ от модуля в виде одного бита «ACK».
- Отправляем второй байт: адрес регистра из которого нужно прочитать данные.
  Получаем ответ от модуля в виде одного бита «ACK».

- Отправляем сигнал «RESTART».
- Отправляем первый байт после «RESTART»: 7 бит адреса и бит «R/W» равный 1 (чтение).
  Получаем ответ от модуля в виде одного бита «ACK».
- Получаем байт данных из регистра модуля.
  Отвечаем битом «ACK» если хотим прочитать следующий регистр, иначе отвечаем «NACK».
- Отправляем сигнал «STOP».

Пример чтения одного регистра:

Чтение из модуля с адресом 0x09 байта данных регистра 0x05:
(в примере модуль вернул значение 0x01).


```
                              // Чтение регистра методами библиотеки Wire.h
byte data;                    // Объявляем переменную для чтения байта данных.
Wire.beginTransmission(0x09); // Инициируем передачу данных в устройство с адресом 0x09.
Wire.write(0x05);             // Записываем в буфер байт адреса регистра.
Wire.endTransmission(false);  // Выполняем передачу без установки состояния STOP.
Wire.requestFrom(0x09, 1);    // Читаем 1 байт из устройства с адресом 0x09. Функция возвращает количество реально принятых
data=wire.read();             // Сохраняем прочитанный байт в переменную data.
```

Пример чтения нескольких регистров подряд:

Чтение из модуля с адресом 0x09 нескольких регистров начиная с регистра 0x05:
(в примере модуль вернул значения: 0x01 из рег. 0x05, 0x13 из рег. 0x06, 0xC3 из рег. 0x07).


```
                                // Чтение регистров методами библиотеки Wire.h
byte data[3];                   // Объявляем массив для чтения данных.
Wire.beginTransmission(0x09);   // Инициируем передачу данных в устройство с адресом 0x09.
Wire.write(0x05);               // Записываем в буфер байт адреса регистра.
Wire.endTransmission(false);    // Выполняем передачу без установки состояния STOP.
Wire.requestFrom(0x09, 3);      // Читаем 3 байта из устройства с адресом 0x09. Функция возвращает количество реально принять
int i=0;                        // Определяем счётчик номера прочитанного байта.
while( Wire.available() ){      // Выполняем цикл while пока есть что читать из буфера.
  if(i<3){                      // Лучше делать такую проверку, чтоб не записать данные за пределы массива data!
    data[i] = wire.read(); i++; // Читаем очередной байт из буфера в массив data.
  }                             //
}                               //
```

Примечание:

- Если на линии I2C только один мастер, то сигнал «RESTART» можно заменить на сигналы «STOP» и «START».
- Рекомендуется не выполнять чтение или запись данных чаще 200 раз в секунду.

Обратите внимание на сигналы «RESTART» и «STOP» в пакетах чтения данных:

- Между фронтом и спадом сигнала «RESTART» проходит фронт импульса на линии «SCL», что расценивается как передача бита равного 1.
- Между сигналом «NACK» и сигналом «STOP» проходит фронт импульса на линии «SCL», что расценивается как передача бита равного 0.
- Эти биты не сохраняются в модулях и не расцениваются как ошибки.

Модуль не поддерживает горячее подключение: Подключайте модуль только при отсутствии питания и данных на шине I2C. В противном случае потребуется отключить питание при уже подключённом модуле.

Пример запуска вращения (100rpm) и остановки вала:

Следующий скетч демонстрирует запуск вращения вала на скорости 100 оборотов в минуту и остановку вала на 2 секунды.

```
#include <Wire.h>                    // Подключаем библиотеку Wire для работы с шиной I2C.
const int ADDRESS = 0x09;            // Определяем адрес модуля.
const int REG_RPM = 0x17;            // Определяем адрес регистра задания скорости хранящего младший байт количества оборотов
const int REG_STP = 0x24;            // Определяем адрес регистра остановки вала.
const int VAL_RPM = 100;             // Определяем скорость вращения вала как 100 оборотов в минуту.
                                     //
void setup(){                        //
    Wire.setClock(100000L);          // Устанавливаем скорость передачи данных по шине I2C.
    Wire.begin();                    // Инициируем работу с шиной I2C в качестве мастера.
}                                    //
                                     //
void loop(){                         //
//  Запускаем мотор:                 //
    Wire.beginTransmission(ADDRESS); // Инициируем передачу данных по шине I2C к устройству с адресом ADDRESS и битом RW=0 (з
    Wire.write(REG_RPM);             // Функция write() помещает значение своего аргумента в буфер для передачи. В данном слу
    Wire.write( lowByte (VAL_RPM) ); // Функция write() помещает значение своего аргумента в буфер для передачи. В данном слу
    Wire.write( highByte(VAL_RPM) ); // Функция write() помещает значение своего аргумента в буфер для передачи. В данном слу
    Wire.endTransmission();          // Выполняем инициированную ранее передачу данных.
    delay(2000);                     // Добавляем задержку в две секунды.
//  Останавливаем мотор:             //
    Wire.beginTransmission(ADDRESS); // Инициируем передачу данных по шине I2C к устройству с адресом ADDRESS и битом RW=0 (з
    Wire.write(REG_STP);             // Функция write() помещает значение своего аргумента в буфер для передачи. В данном слу
    Wire.write(0b00000010);          // Функция write() помещает значение своего аргумента в буфер для передачи. В данном слу
//  Wire.write(0b00000011);          // Если место предыдущей строки отправить число с установленными битами BIT_STOP и BIT_N
```

```
    Wire.endTransmission();           // Выполняем инициированную ранее передачу данных.
    delay(2000);                      // Добавляем задержку в две секунды.
}                                     //
```

Габариты: