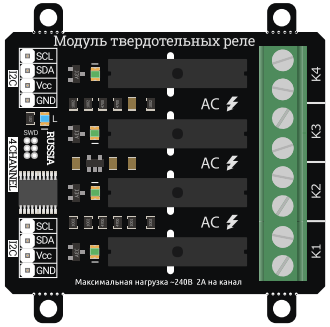
Модуль твердотельное реле, 4 канала, FLASH-I2C - Datasheet



Модуль твердотельных реле на 4 канала.

Техническое описание: Данная страница содержит подробное техническое описание модуля твердотельных реле на 4 канала, I2C, Flash и раскрывает работу с модулем через его регистры.
Ознакомиться с пользовательским описанием модуля и примерами работы с библиотекой iarduino_I2C_Relay можно на странице Wiki - Модуль твердотельных реле на 4 канала, I2C, Flash.

Назначение:

Модуль твердотельных реле на 4 канала, I2C, Flash - является устройством коммутации, которое позволяет подключать и отключать устройства к сети переменного тока от 120В до 240В. При этом устройства подключённые через выходные контакты модуля, не должны потреблять более 2А переменного тока (на каждый канал).

Управление модулем осуществляется по 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.

Модуль можно использовать в любых проектах где требуется управлять устройствами с напряжением питания от 120В до 240В и потреблением переменного тока до 2А.

Описание:

Модуль построен на базе микроконтроллера STM32F030F4, снабжен собственным стабилизатором напряжения и четырьмя твердотельными реле.

Силовые устройства подключаются к сети переменного тока (от 120В до 240В) через контакты разъемов «К1», «К2», «К3», «К4».

О состоянии реле можно судить по светодиодам расположенным рядом с реле.

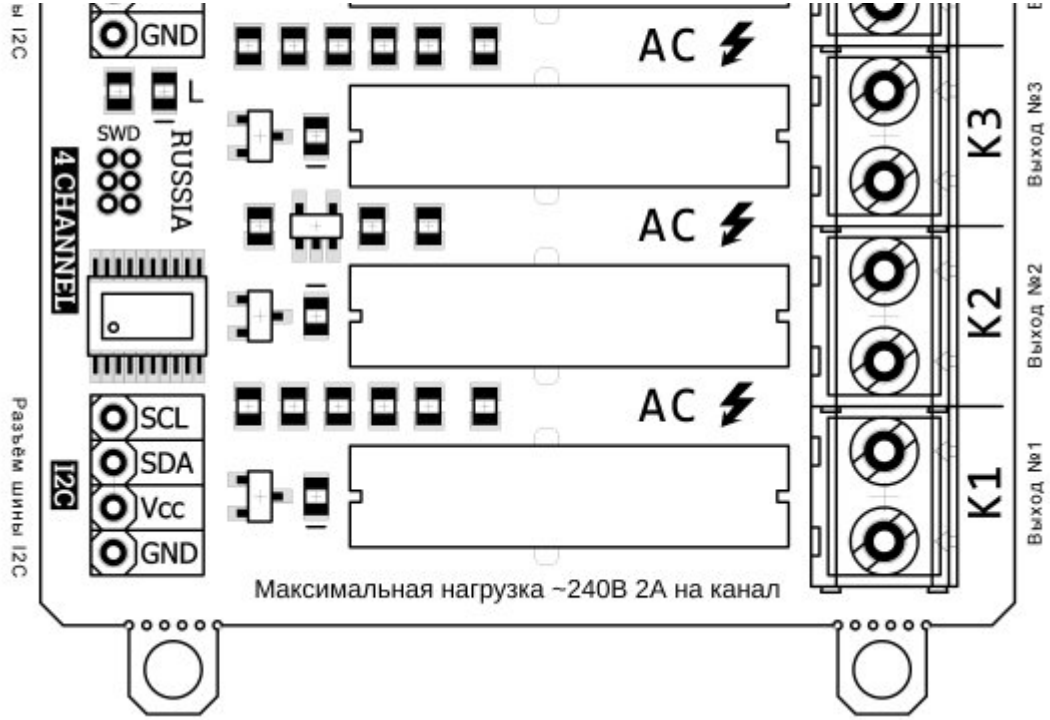
- Если светодиод выключен, значит реле отключено и его выводы разъединены.
- Если светодиод светится, значит реле включено и его выводы замкнуты.

Управление выходными контактами модуля осуществляется через его регистры. Доступ к регистрам модуля осуществляется по шине I2C.

С помощью регистров модуля можно:

- Изменить адрес данного модуля, временно (пока есть питание) или постоянно.
- Включить / отключить любое реле.

Выводы модуля:



В левой части платы расположены два разъема для подключения модуля к шине **I2C**. Шина подключается к любому разъему **I2C**, а второй разъем можно использовать для подключения следующего модуля твердотельных реле, или других устройств.

- **SCL** - вход/выход линии тактирования шины I2C.
- **SDA** - вход/выход линии данных шины I2C.
- **Vcc** - вход питания модуля 5В.
- **GND** - общий вывод питания.

В правой части платы расположены четыре разъема: **K1**, **K2**, **K3**, **K4**, это выходы, через контакты которых подключаются силовые устройства к сети переменного тока от 120В до 250В. Устройства не должны потреблять более 2А (на каждый канал).

- **K1** - разъём первого реле с нормально разомкнутыми «NO» (Normally Open) контактами.
- **K2** - разъём второго реле с нормально разомкнутыми «NO» (Normally Open) контактами.
- **K3** - разъём третьего реле с нормально разомкнутыми «NO» (Normally Open) контактами.

- **K4** - разъём четвёртого реле с нормально разомкнутыми «NO» (Normally Open) контактами.

Характеристики:

- Напряжение питания: 5 В (постоянного тока).
- Потребляемый ток: до 20 мА.
- Коммутируемое напряжение: от 120 В до 250 В (переменного тока).
- Коммутируемый ток: до 2 А (на каждый канал).
- Количество каналов: 4.
- Интерфейс: I2C.
- Скорость шины I2C: 100 кбит/с.
- Адрес на шине I2C: устанавливается программно (по умолчанию 0x09).
- Уровень логической 1 на линиях шины I2C: 3,3 В (толерантны к 5 В, подтянуты к 5 В).
- Рабочая температура: от -40 до +65 °C.
- Габариты с креплением: 55 x 55 мм.
- Габариты без креплений: 55 x 45 мм.
- Вес: 30 г.

Установка адреса:

Модуль твердотельных реле на 4 канала, I2C, Flash, I2C, Flash относится к линейке «Flash» модулей. Все модули данной линейки 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 линейки «Flash» поставляются с адресом 0x09.

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]							

адрес	7	6	5	4	3	2	1	0
0x06	ADDRESS[6-0]							SAVE_FLASH
0x07	CHIP_ID[7-0]							
0x08 ... 0x11	RESERVED							
0x12	-	-	-	-	DIGITAL-2	DIGITAL-3	DIGITAL-2	DIGITAL-1
0x13	WRITE_H-4	WRITE_H-3	WRITE_H-2	WRITE_H-1	WRITE_L-4	WRITE_L-3	WRITE_L-2	WRITE_L-1
0x14 ... 0x2F	RESERVED							
0x30	WDT[7-0]							

Регистры с адресами 0x02, 0x03, 0x08 - 0x11, 0x14 - 0x2F зарезервированы, их биты сброшены. Попытка записи данных в эти регистры будет проигнорирована модулем.

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 твердотельных реле на 4 канала - идентификатор равен 0x0B.

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.
- **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 всей линейки «Flash» модулей:

Регистр только для чтения.

У всех модулей линейки «Flash» в регистре «CHIP_ID» содержится значение 0x3C. Если требуется отличить модули линейки «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 линейки «Flash».

Регистр 0x12 «DIGITAL_ALL» - содержит биты управления реле:

Регистр для чтения и записи.

- **DIGITAL-1...4** - Биты определяют состояния выходов реле: «0» - разомкнуты, «1» - замкнуты.
  Пример: DIGITAL_ALL = $(XXXX0011)_2$ => выходы K1, K2 замкнуты, выходы K3, K4 - разомкнуты.
  Таким образом, можно управлять всеми выходами записав всего один байт в данный регистр.

Регистр 0x13 «DIGITAL_ONE» - содержит биты управления реле:

Регистр для чтения и записи.

- **WRITE_H-1...4** - Установка данных битов приводит к установке соответствующих битов «DIGITAL-1...4» регистра 0x12 «DIGITAL_ALL» и, как следствие, замыканию контактов соответствующих выходов.
  Биты сбрасываются самостоятельно.
- **WRITE_L-1...4** - Установка данных битов приводит к сбросу соответствующих битов «DIGITAL-1...4» регистра 0x12 «DIGITAL_ALL» и, как следствие, размыканию контактов соответствующих выходов.
  Биты сбрасываются самостоятельно.
- Данный регистр, в отличии от регистра 0x12 «DIGITAL_ALL», удобно использовать когда требуется изменить состояние выбранных

выходов, не меняя состояние остальных выходов.

Регистр 0x30 «WDT» - содержит время сторожевого таймера:

Регистр для чтения и записи.

Сторожевой таймер предназначен для безопасности Вашего устройства.

- **WDT[7-0]** - Значение от 0 до 254 указывает время (в сек.) оставшееся до перезагрузки модуля.
  Значение 255 (по умолчанию) означает, что сторожевой таймер отключён (не считает).
  Если записать число от 1 до 254, то каждую секунду, значение регистра будет уменьшаться на единицу, пока не достигнет 0. По достижении 0, модуль перезагрузится и все каналы, а так же таймер, будут отключены.
  ПРИМЕЧАНИЕ: Если сторожевой таймер досчитает до 0, то модуль перезагрузится и данные всех его регистров сбросятся в значения по умолчанию, по умолчанию нагрузки отключены.
  ПРИМЕР: Управляющее устройство (например, Arduino) постоянно (в цикле loop) отправляет в регистр «WDT» значение 10. Как только Arduino перестанет работать (отключится, зависнет), значение регистра «WDT» начнёт уменьшаться и через 10 секунд, нагрузки подключённые к модулю будут отключены.

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».

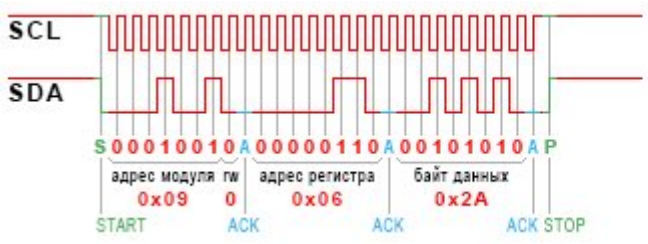
Запись данных в регистры:

- 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:

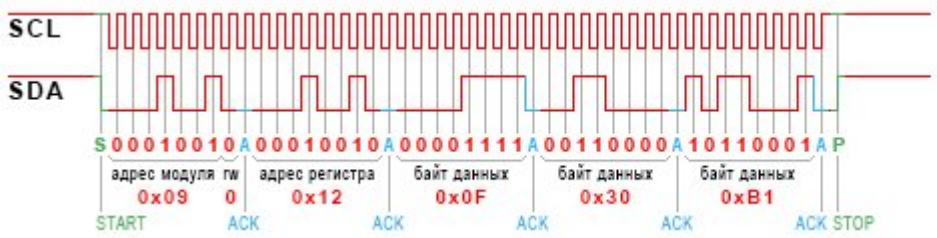



```
                                // Запись в регистр методами библиотеки Wire.h
Wire.beginTransmission(0x09);   // Инициируем передачу данных в устройство с адресом 0x09.
Wire.write(0x06);               // Записываем в буфер байт адреса регистра.
Wire.write(0x26);               // Записываем в буфер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

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 приняты
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.

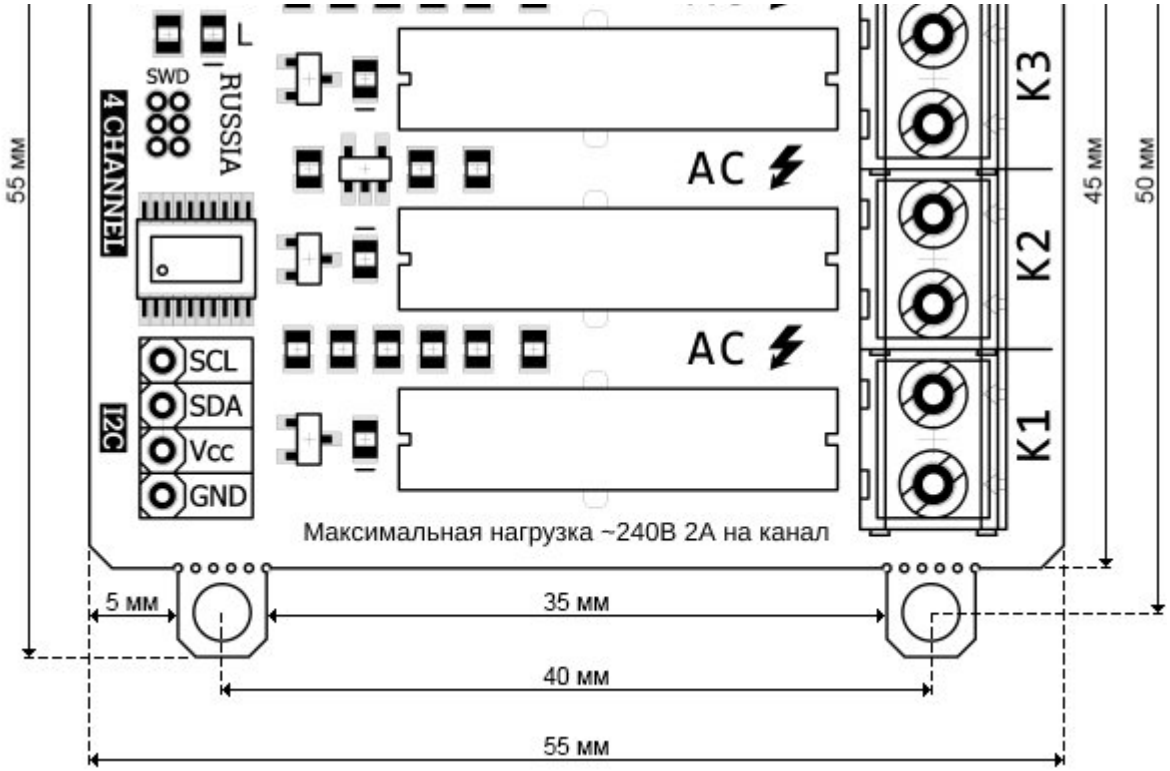
Пример включения и выключения 1 и 2 выхода модуля:

- В начале скетча определены константы с указанием адреса модуля и адреса регистра с помощью которого можно управлять одним выходом.
- В коде setup() была инициирована работа с шиной I2C.
- В коде loop с промежутками в 500 миллисекунд, в регистр «REG_DIGITAL_ONE» отправляется сначала байт данных со значением 0b00110000 (включение 1 и 2 канала), а потом со значением 0b00000011 (выключение 1 и 2 канала).

```
#include <Wire.h>                          // Подключаем библиотеку Wire для работы с шиной I2C.
const int ADDRESS         =  0x09;         // Определяем адрес модуля.
const int REG_DIGITAL_ONE =  0x13;         // Определяем адрес регистра DIGITAL_ONE для управления выходами.
                                           //
```

```
void setup(){                                   //
    Wire.setClock(100000L);                     // Устанавливаем скорость передачи данных по шине I2C.
    Wire.begin();                               // Инициируем работу с шиной I2C в качестве мастера.
    delay(500);                                 //
}                                               //
                                                //
void loop(){                                    //
//  Включаем 1 выход модуля:                    //
    Wire.beginTransmission(ADDRESS);            // Инициируем передачу данных по шине I2C к устройству с адресом ADDRESS и
    Wire.write(REG_DIGITAL_ONE);                // Функция write() помещает значение своего аргумента в буфер для передачи
    Wire.write(0b00110000);                     // Функция write() помещает значение своего аргумента в буфер для передачи
    Wire.endTransmission();                     // Выполняем инициированную ранее передачу данных.
    delay(500);                                 // Добавляем задержку в пол секунды.
//  Выключаем 1 выход модуля:                   //
    Wire.beginTransmission(ADDRESS);            // Инициируем передачу данных по шине I2C к устройству с адресом ADDRESS и
    Wire.write(REG_DIGITAL_ONE);                // Функция write() помещает значение своего аргумента в буфер для передачи
    Wire.write(0b00000011);                     // Функция write() помещает значение своего аргумента в буфер для передачи
    Wire.endTransmission();                     // Выполняем инициированную ранее передачу данных.
    delay(500);                                 // Добавляем задержку в пол секунды.
}                                               //
```

Габариты:


SWD
RUSSIA
4 CHANNEL
I2C
SCL
SDA
Vcc
GND
AC
AC
K3
K2
K1
Максимальная нагрузка ~240В 2А на канал
55 мм
45 мм
50 мм
5 мм
35 мм
40 мм
55 мм