**Библиотека для MIC74 под Arduino**

*Написание библиотеки Ардуино с помощью искусственного интеллекта*

**Краткая характеристика микросхемы MIC74.**

Чип MIC74 представляет собой полностью программируемый последовательно-параллельный преобразователь-расширитель ввода-вывода, совместимый с протоколом SMBus™. Он работает как клиент на шине, обеспечивающий восемь независимых линий ввода/вывода.

Каждый бит ввода/вывода может быть индивидуально запрограммирован как вход или как выход. Если какой-либо вывод запрограммирован как выход, то каждый такой вывод может быть запрограммирован как выход с открытым стоком или как двухтактный выход. При желании четыре выхода могут быть запрограммированы на реализацию регулировки скорости вращения вентилятора. Внутренний тактовый генератор и система состояния устраняют накладные расходы, необходимые для управления скоростью вращения вентилятора.

Выходные выводы способны напрямую управлять нагрузками, такими как светодиоды. Имеется возможность использования прерываний во время изменения состояния на входных выводах. Это исключает необходимость опрашивать устройство каждый раз для получения информации о состоянии выводов. Предоставляются три входа выбора адреса, позволяющих подключать к одной шине до восьми устройств и использовать их совместно, что обеспечивает в общей сложности 64 дополнительных ввода-вывода Вашему микроконтроллеру.

Чип MIC74 доступен в ультракомпактном 16-контактном корпусе QSOP. Низкий ток покоя, небольшая занимаемая площадь и малая высота корпуса делают MIC74 идеальным решением для портативных и настольных приложений:

*Тип корпуса и расположение выводов MIC74*

**Основные технические характеристики микросхемы MIC74 приведены в таблице**:

| **Параметр** | **Обозначение** | **Значение** | **Единица** | **Условия** |
| --- | --- | --- | --- | --- |
| Напряжение питания | VDD | +2.7 ... +3.6 | Вольт |  |
| Рабочий ток потребления | IDD | 2 - 6 | мкА | входы P[7:0] = VDD или GND;   /ALERT открыт; fCLK = 100 кГц |
| Ток потребления в режиме ожидания | ISTBY | 1 - 3 | мкА | /ALERT = открыт,  VCLK = VDATA = VDD;  P[3:0] = входы |
| Температура окружающей среды | TA | -40 ... +85 | °C |  |

*Таблица 1 - Основные технические характеристики MIC74*

Микросхема работает от низкого напряжения и имеет малый ток потребления, а остальные параметры можно посмотреть в прилагаемой документации.

**Назначение выводов микросхемы MIC74 приведено в отдельной таблице:**

| **Номер вывода** | **Обозначение** | **Описание** |
| --- | --- | --- |
| 1, 2, 3 | A0, A1, A2 | Установка адреса (входы) |
| 4, 5, 6, 7 | P0, P1, P2, P3 | Параллельный ввод/вывод (входы/выходы) |
| 8 | GND | Общий |
| 9, 10, 11, 12 | P4, P5, P6, P7 | Параллельный ввод/вывод (входы/выходы) |
| 13 | /ALERT | Прерывание (выход) |
| 14 | CLK | Тактовый генератор последовательной шины (вход) |
| 15 | DATA | Данные последовательной шины (вход/выход) |
| 16 | VDD | Источник питания (вход) |

*Таблица 2 - Назначение выводов MIC74*

Рассмотрим более подробное описание и функции некоторых выводов чипа MIC74:

* A0, A1, A2 - Входы установки адреса клиента, задаются три младших бита адреса чипа MIC74;
* P0, P1, P2, P3, P4, P5, P6, P7 - Контакты ввода/вывода общего назначения, направление и тип выхода программируются пользователем;
* /ALERT - Выход с активным низким уровнем и открытым стоком, выходной сигнал прерывания для хоста, при изменении логического уровня на входах микросхемы. Сигнал сбрасывается, когда хост шины опрашивает ARA (адрес ответа на оповещение = 0001100) или считывает статус;
* CLK - Хост обеспечивает синхронизирующие тактовые импульсы на этом входе;
* DATA - Последовательный ввод данных и последовательный вывод данных с открытым стоком.

Кроме этого выводы 9, 10, 11, 12 имеют альтернативную функцию управления вентилятором, при включении которой они обозначаются как /SHDN, /FS0, /FS1, /FS2. Когда включён режим управления вентилятором, то вывод /SHDN управляет запуском/остановкой вентилятора, а выводы /FS0, /FS1 и /FS2 - скоростью его вращения.

Адрес каждого отдельного чипа задаётся сочетанием логических уровней сигналов на трёх его входах A0, A1, A2 во время включения, и не может быть изменён во время работы:

| **Уровень A2** | **Уровень A1** | **Уровень A0** | **HEX Адрес** |
| --- | --- | --- | --- |
| 0 | 0 | 0 | 0x20 |
| 0 | 0 | 1 | 0x21 |
| 0 | 1 | 0 | 0x22 |
| 0 | 1 | 1 | 0x23 |
| 1 | 0 | 0 | 0x24 |
| 1 | 0 | 1 | 0x25 |
| 1 | 1 | 0 | 0x26 |
| 1 | 1 | 1 | 0x27 |

*Таблица 3 - Конфигурация адреса MIC74*

**Описание и назначение регистров микросхемы MIC74**

Для настройки и управления у микросхемы MIC74 имеется 7 регистров, каждый из которых состоит из 8 бит:

| **Название** | **Назначение** | **HEX адрес** | **Значение по умолчанию** |
| --- | --- | --- | --- |
| DEV\_CFG | Конфигурация устройства | 0x00 | 0b00000000 |
| DIR | Направление ввода/вывода | 0x01 | 0b00000000 |
| OUT\_CFG | Конфигурация выходов | 0x02 | 0b00000000 |
| STATUS | Состояние прерывания | 0x03 | 0b00000000 |
| INT\_MASK | Маска прерывания | 0x04 | 0b00000000 |
| DATA | Ввод/вывод общего назначения | 0x05 | 0b11111111 |
| FAN\_SPEED | Скорость вентилятора | 0x06 | 0b00000000 |

*Таблица 4 - Регистры MIC74*

Рассмотрим работу с регистрами более подробно:

* **DEV\_CFG** - Регистр конфигурации микросхемы, двумя младшими битами которого задаются режимы работы, такие как глобальное разрешение прерываний и управление вентилятором. Остальные биты зарезервированы и в них всегда должны быть записаны нули. После запуска чипа, по умолчанию все биты этого регистра имеют нулевое значение, означающее что прерывания и режим управления вентилятором отключены. Для разрешения прерываний нужно установить нулевой бит, а для управления вентилятором установить бит номер 1 в единицу, например записью в регистр значения 0b00000011. Регистр предназначен как для записи, так и для чтения.;
* **DIR** - Регистр настройки всех одновременно, или отдельных выводов на работу как входы, или как выходы. После включения регистр имеет значение по умолчанию 0b00000000 и все выводы настроены как входы. Соответственно для настройки отдельного вывода для работы на выход, соответствующий бит данного регистра нужно установить в единицу. Из регистра так же можно читать установленные значения. Если чип сконфигурирован на работу управления вентилятором, то выводы P[7:4] автоматически настраиваются как выходы с открытым стоком, и регистр DIR не влияет на режим работы этих выводов;
* **OUT\_CFG**- Регистр настройки всех одновременно, или отдельных выводов, сконфигурированных как выходы, на работу в двухтактном режиме, или режиме с открытым стоком. По умолчанию регистр имеет значение 0b00000000 и все выходные каскады сконфигурированы на работу в режиме с открытым стоком. Соответственно для настройки отдельного выхода для работы в двухтактном режиме, соответствующий бит данного регистра нужно установить в единицу. Из регистра так же можно читать установленные значения. Если чип сконфигурирован на работу управления вентилятором, то выводы P[7:4] автоматически настраиваются как выходы с открытым стоком, и регистр OUT\_CFG не влияет на режим работы этих выводов;
* **STATUS** - В битах этого регистра фиксируются события изменения логического уровня на соответствующих выводах микросхемы, сконфигурированных как входы. Бит устанавливается в единицу, когда происходит смена логического уровня на соответствующем входе, а если изменения не происходили, то бит сброшен в 0. Регистр предназначен только для чтения, и все его биты сбрасываются в 0 после выполнения этой операции. Если чип сконфигурирован на работу управления вентилятором, то выводы P[7:4] автоматически настраиваются как выходы с открытым стоком, и никакие прерывания не генерируются изменением логических уровней на этих выводах;
* **INT\_MASK** - Регистр разрешения прерываний на выводах, сконфигурированных как входы, и соответствующие биты которых в этом регистре установлены в единицу. Регистр используется для чтения/записи. Если чип сконфигурирован на работу управления вентилятором, то выводы P[7:4] автоматически настраиваются как выходы с открытым стоком, и никакие прерывания не генерируются изменением логических уровней на этих выводах;
* **DATA** - В регистре отображается текущее состояние любого вывода, сконфигурированного как вход, и последнее значение, применённое к выводу, сконфигурированному как выход. Запись значения в регистр DATA устанавливает состояние любого вывода, сконфигурированного как выход; запись в биты ввода-вывода, сконфигурированных как входы, игнорируется. По умолчанию все биты регистра установлены в единицу. Если чип сконфигурирован на работу управления вентилятором, то выводы P[7:4] автоматически настраиваются как выходы с открытым стоком, и их логические уровни управляются посредством регистра FAN\_SPEED, а регистр DATA не влияет на работу этих выводов;
* **FAN\_SPEED** - Регистр управления работой вентилятора, в котором задействованы три младших бита. В любое время, когда регистр содержит значение ноль, это означает, что вентилятор выключен, а если в него записывается ненулевое значение, то выходы /FS[2:0] и /SHDN примут состояние максимальной скорости вентилятора примерно на одну секунду (tSTART). После этого интервала состояние выходов управления скоростью вентилятора примет значение, указанное в содержимом регистра FAN\_SPEED. Это гарантирует, что вентилятор надежно запустится даже тогда, когда требуется работа на низкой скорости. Зависимость скорости вращения вентилятора от значения младших битов регистра приведена в следующей таблице, причём "Скорости 1" соответствует самая низкая, а "Скорости 7" самая высокая фактическая скорость вращения вентилятора:

| **Значение бит**  **регистра [2:0]** | **Выходной уровень**  **/FS[2:0]** | **Выходной уровень**  **/SHDN** | **Скорость вентилятора** |
| --- | --- | --- | --- |
| 000 | 111 | 0 | Выключен |
| 001 | 110 | 1 | Скорость 1 |
| 010 | 101 | 1 | Скорость 2 |
| 011 | 100 | 1 | Скорость 3 |
| 100 | 011 | 1 | Скорость 4 |
| 101 | 010 | 1 | Скорость 5 |
| 110 | 001 | 1 | Скорость 6 |
| 111 | 000 | 1 | Скорость 7 |

*Таблица 5 - Установка скорости вращения вентилятора*

К этому моменту времени мы уже рассмотрели все имеющиеся регистры чипа MIC74, и хотелось бы разделить их на две категории по принципу регулярности и частоты использования в основном цикле программного кода: "*Конфигурационные*" и "*Рабочие*":

| **Конфигурационные** | **Рабочие** |
| --- | --- |
| DEV\_CFG | STATUS |
| DIR | DATA |
| OUT\_CFG | FAN\_SPEED |
| INT\_MASK |  |

*Таблица 6 - Категории регистров*

Соответственно конфигурационные регистры используются в основном для настройки и инициализации устройства, а рабочие - для операционного управления им. По логике вещей обращение к категории "Рабочих" регистров происходит чаще и именно во время работы устройства. Хотя это определение тоже немного условно и относительно, так как в процессе работы устройства может потребоваться изменение направления ввода/вывода или режима работы выходного каскада, а так же маскировка определённых выводов для прерываний.

**Функционал библиотеки**

Согласно рассмотренным регистрам, определимся с необходимыми функциями, которые требуется реализовать для управления и конфигурации чипа MIC74. Для простоты и удобства будет лучше двигаться именно от управления и далее до конфигурации. Во всех цифровых устройствах управление подразумевает чтение или запись состояний всех или определённых выводов порта, а также чтение и запись сигналов дополнительных вспомогательных выводов, если таковые имеются.

В данном случае выводы могут быть настроены как выходы, так и входы, соответственно понадобится реализовать функции как для записи, так и для чтения указанных состояний. Дополнительно выводы, настроены как выходы могут работать в двухтактном режиме, или в режиме с открытым стоком, а выводы, настроенные как входы, могут использовать прерывания, или не использовать их.

Составим перечень функций согласно имеющимся возможностям управления и требующимся параметрам конфигурации.

*Запись логических уровней*

Взяв за основу стиль Ардуино можно определить функцию установки заданного логического уровня на требуемом выводе микросхемы:

**digitalWrite(***pin, value***);**

* **pin**: вывод порта от 0 до 7;
* **value**: записываемое значение HIGH или LOW.

Для примера digitalWrite(3, HIGH); установит высокий уровень на выводе P3 микросхемы, а digitalWrite(3, LOW); низкий логический уровень на том же выводе.

Дополнительно реализуем альтернативные функции **pinHigh(***pin***);** и **pinLow(***pin***);** для установки на указанном выводе чипа **pin** соответственно высокого и низкого логического уровня.

Для примера pinHigh(3); установит высокий логический уровень на выводе P3 микросхемы, а pinLow(3); установит низкий логический уровень на этом же выводе.

Также может оказаться очень полезной функция записи логических уровней на всех выводах одновременно, то есть запись значения в порт микросхемы. Назовём её так:

**portWrite(***value***);**

* **value**: записываемое значение, которое для наглядности удобно представлять в виде битового поля 0bxxxxxxxx.

Где в необходимых позициях вместо "x" записывается "0" или "1", например portWrite(0b11111111); одновременно установит на всех выводах чипа высокий логический уровень, а portWrite(0b00000000); соответственно низкий.

Можно подумать ещё о функциях отложенной записи значений в порт. В случае когда все, или большинство выводов настроены на работу как выходы, и разные выводы управляются отдельными функциями, в процессе работы программы разные её участки будут изменять разные биты порта ввода/вывода, и каждый раз физическое изменение состояний выводов может повлечь за собой повышенную нагрузку на шину и на управляющий микроконтроллер. В отдельных случаях можно оптимизировать работу системы и производить отправку значений в порт один раз за цикл, или обновлять значение порта через определённые интервалы времени, например по прерываниям. Для этого накопленные изменения будут храниться в буфере, а затем будет производиться физическая отправка значения буфера в порт микросхемы.

Для этих целей можно создать такие же функции установки уровней на выводах чипа с добавлением к их названиям слова "delayed", что означает отложенная запись. Ещё понадобиться дополнительная функция, которая будет отправлять значение буфера в порт по заданному графику:

**digitalWriteDelayed(***pin, value***);**

* **pin**: вывод порта от 0 до 7;
* **value**: записываемое значение HIGH или LOW.

**pinHighDelayed(***pin***);** и **pinLowDelayed(***pin***);**

**portWriteDelayed(***value***);**

* **value**: записываемое значение, которое для наглядности удобно представлять в виде битового поля 0bxxxxxxxx.

**updatePort();** - физическая запись значения буфера в регистр порта.

Главное подходить к использованию последних функций осознанно, так как могут произойти аномалии, когда один участок кода устанавливает логический уровень определённого вывода, от которого зависит логика работы дальнейшая, следующих за этим участком алгоритмов, а физическая запись значения в порт пока не произошла. Так же естественно не получится использовать отложенную запись там, где используются задержки в коде программы, если например мы хотим на секунду включить светодиод, а отправка буфера в порт будет ожидать истечение времени его свечения, то никакого физического зажигания светодиода не произойдёт вообще.

*Чтение логических уровней*

Данный функционал можно осуществить в двух режимах: оперативном, когда опрос состояния выводов микросхемы осуществляется периодически; и в так называемом ждущем режиме, при котором считывание уровней происходит только после их фактического изменения. Для второго способа нужно будет задействовать механизм прерывания, но он позволяет работать в энергосберегающем режиме, а так же значительно уменьшить нагрузку на шину и микроконтроллер. Для полноценной работы с микросхемой нужно предусмотреть оба режима и реализовать соответствующий функционал.

Для циклического считывания логических уровней разработаем функцию:

**digitalRead(***pin***);**

* **pin**: вывод порта от 0 до 7;
* **возвращаемое значение**: HIGH или LOW.

Для примера digitalRead(3); возвратит значение HIGH при высоком логическом уровне на выводе P3, и LOW при низком логическом уровне на этом выводе.

Также может понадобиться чтение всех значений порта одновременно, для чего будет полезно иметь функцию:

**portRead();**

* **возвращаемое значение**: число от 0 до 255, хранящее битовое поле логических уровней порта ввода/вывода..

Для примера после выполнения int val = portRead(); переменная val будет иметь значение 0b00001010 при высоких логических уровнях только на выводах P1 и P3 микросхемы, и низких логических уровнях на всех остальных её выводах.