Современные 32-разрядные ARM-микроконтроллеры серии STM32: порты общего назначения GPIO

Олег Вальпа

В статье приведено описание портов общего назначения GPIO, 32-разрядных ARM-микроконтроллеров серии STM32 компании STMicroelectronics. Рассмотрены архитектура портов, состав и назначение регистров конфигурирования, а также примеры программ инициализации.

ВВЕДЕНИЕ

Порты ввода-вывода являются основными средствами связи микроконтроллера с внешними устройствами. С их помощью можно опрашивать состояние кнопок, контактов реле, уровней сигнала и т.п., а также формировать сигналы управления различными устройствами, например светодиодами, реле, оптическими транзисторами и т.д.

Порты ввода-вывода имеются практически в любом микроконтроллере независимо от его архитектуры. Не являются исключением и микроконтроллеры STM32 компании STMicroelectronics [1], имеющие в зависимости от модели до семи портов.

ОПИСАНИЕ ПОРТОВ GPIO

Все порты ввода-вывода STM32 являются 16-разрядными и называются GPIO, от General Purpose Input/Output (входы/выходы общего назначения). Порты обозначаются буквами латинского алфавита от A до G , т.е. имеют названия PORTA, PORTB, PORTC, PORTD, PORTE, PORTF и PORTG.

Любой из единичных каналов этих портов может быть сконфигурирован как вход или как выход. Если канал порта настроен на ввод информации, он может функционировать как цифровой или аналоговый вход. В режиме цифрового входа канал порта подключается с помощью внутреннего подтягивающего резистора к плюсу источника питания или к нулевому потенциалу.

Настройка порта на вывод требует задания частоты тактирования, определяющей его максимальное быстродействие и тип выхода. Максимальное быстродействие может принимать значения: 2, 10 или 50 МГц. Ниже перечислены возможные типы выхода:

- двухтактный выход;
- выход с открытым стоком;
- двухтактный выход с альтернативной функцией;
- выход с открытым стоком и альтернативной функцией.

Последние два режима применяются для выводов, которые используют какой-либо функциональный блок периферийного устройства, например SPI, I²C, USART, DAC и т.п. Интересной и полезной особенностью портов является возможность чтения состояния порта, настроенного как выход с открытым стоком. Это позволяет программе осуществлять чтение данных с двунаправленной линии связи без переконфигурирования порта. Такая операция часто востребована для организации программного интерфейса I²C и 1Wire.

На рисунке 1 приведена структурная схема одного канала порта ввода-вывода, иллюстрирующая вышеизложенное описание. С помощью транзисторов PMOS и NMOS выход порта может принимать значения высокого потенциала в качестве логической единицы или нулевого потенциала в качестве логического нуля. Если порт сконфигурирован как выход с открытым стоком, то для управления состоянием линии используется только нижний транзистор NMOS, а верхний транзистор PMOS находится в отключенном состоянии.

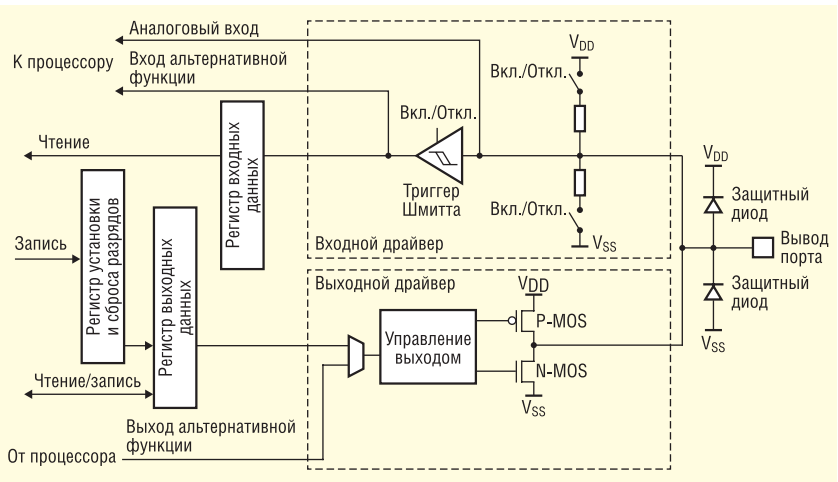

Рис. 1. Структурная схема одного канала порта ввода-вывода

С помощью внутренних резисторов можно подтянуть вывод, сконфигурированный как вход, к положительному потенциалу источника питания или к нулевому потенциалу.

Важными элементами порта являются защитные диоды, функция которых — защищать входы контроллера от перенапряжений. Поскольку микроконтроллер питается от номинального напряжения 3.3 В, на его входы подаются сигналы с уровнями, не превышающими это напряжение. Однако некоторые входы микроконтроллера можно подключать к сигналам с 5-В логикой. Такие входы называются толерантными и имеют буквенное обозначение FT. Для подключения сигнала с 5-В логикой к нетолерантному входу необходимо использовать согласующее устройство, которое в простейшем случае представляет собой резистор, имеющий сопротивление 1 кОм.

Отличительной особенностью портов микроконтроллера STM32 является возможность программной блокировки настройки порта. Такая блокировка позволяет запретить изменение конфигурации порта программным образом и может быть снята только после аппаратного сброса. Данная блокировка служит в качестве дополнительной защиты каналов портов и подключенных к нему устройств, предотвращая выход из строя при сбоях от помех. В результате такого сбоя может произойти самопроизвольное изменение конфигурации вывода, при котором порт настроится на выход и вступит в конфликт с подключенным к нему выходом внешнего устройства. Для исключения подобных случаев и используется программная блокировка.

РЕГИСТРЫ ПОРТОВ GPIO

Для конфигурирования портов и работы с ними микроконтроллер имеет по семь регистров для каждого порта. Эти регистры имеют следующие названия и назначение:

- CRL и CRH — регистры, задающие режим работы младшей и старшей половины порта;
- ODR — регистр для записи данных непосредственно в порт;
- IDR — регистр для чтения физического состояния выводов порта;
- BSRR — регистр установки и сброса разрядов порта;
- BRR — регистр сброса разрядов порта;
- LCKR — регистр для блокировки установленной конфигурации.

Рассмотрим подробнее каждый из этих регистров.

Регистры CRL и CRH представляют собой 32-разрядные регистры, задающие режим работы каждого вывода порта. Назначение всех разрядов для регистров CRL и CRH представлено в таблице 1 и таблице 2 соответственно.

Как видно из таблиц, для каждого из каналов порта имеется два двухразрядных поля CNFx[1:0] и MODEx[1:0]. Поле CNF определяет тип работы линии, а поле MODE — направление обмена по линии. Все биты этих полей доступны для чтения и записи.

Поле MODE[1:0] может принимать следующие значения:

- 00 — канал порта работает на ввод (такое состояние устанавливается после сброса);
- 01 — канал порта работает на выход с максимальной частотой переключения 10 МГц;
- 10 — канал порта работает на выход с максимальной частотой переключения 20 МГц;
- 11 — канал порта работает на выход с максимальной частотой переключения 50 МГц.

При работе на вход, когда MODE[1:0]=0, поле CNF[1:0] устанавливает следующие состояния каналов порта:

- 00 — аналоговый вход;
- 01 — дискретный вход в третьем состоянии (такое состояние устанавливается после сброса);
- 10 — дискретный вход с подтягивающим резистором к питанию или общей цепи в зависимости от состояния соответствующего разряда регистра BSRR;
- 11 — зарезервировано для будущих применений.

При работе канала порта на выход, когда поле MODE[1:0] отлично от 0, поле CNF[1:0] позволяет устанавливать следующие состояния каналов порта:

- 00 — цифровой симметричный выход;
- 01 — цифровой выход с открытым стоком;
- 10 — цифровой симметричный выход с альтернативной функцией;
- 11 — цифровой выход с альтернативной функцией и открытым стоком.

Регистр ODR представляют собой 32-разрядный регистр, предназначенный для записи данных непосредственно в порт. Однако в этом регистре используются только младшие 16 бит. Старшие биты, с 16-го по 31-й зарезервированы.

Назначение разрядов регистра ODR представлено в таблице 3.

При записи в регистр ODR какого-либо значения, оно устанавливается на выходах соответствующего порта. Биты этого регистра доступны как для записи, так и для чтения.

Регистр IDR также является 32-разрядным регистром и служит для чтения физического состояния каналов порта, настроенных на вход. Аналогично регистру вывода ODR, в регистре ввода IDR используется только 16 младших бит из 32. Назначение разрядов регистра IDR представлено в таблице 4.

При чтении регистра IDR можно узнать состояние всех каналов соответ-

Таблица 1. Назначение разрядов регистра CRL

Бит регистра	31	30	29	28	27	2	25	24	23	22	21	20	19	18	17	16
Назначение	CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
Канал порта	7				6				5				4			
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
Канал порта	3				2				1				0			

Таблица 2. Назначение разрядов регистра CRH

Бит регистра	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Назначение	CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
Канал порта	15				14				13				12			
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
Канал порта	11				10				9				8			

Таблица 3. Назначение разрядов регистра ODR

Бит регистра	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Назначение	Резерв															
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR 9	ODR 8	ODR 7	ODR 6	ODR 5	ODR 4	ODR 3	ODR 2	ODR 1	ODR 0

Таблица 4. Назначение разрядов регистра IDR

Бит регистра	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Назначение	Резерв															
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	IDR15	IDR14	IDR13	IDR12	IDR11	IDR10	IDR 9	IDR 8	IDR 7	IDR 6	IDR 5	IDR 4	IDR 3	IDR 2	IDR 1	IDR 0

ствующего порта. Биты данного регистра доступны только для чтения.

32-разрядный регистр BSRR предназначен для сброса и установки отдельных каналов порта, настроенных на вывод. Назначение разрядов регистра BSRR представлено в таблице 5.

Данный регистр позволяет менять состояние конкретных каналов порта без изменения остальных каналов. Запись единицы в один из старших разрядов этого регистра сбрасывает соответствующий выход канала порта, а запись единицы в младшие разряды устанавливает высокий уровень сигнала на соответствующем канале порта.

Запись в регистр производится в формате 32-разрядного слова, при этом нулевые биты никакого действия не оказывают. Все разряды данного регистра доступны только для записи. Для исключения неоднозначности при одновременной записи единиц в разряды BS и BR приоритетными будут разряды BS, т. е. выходы каналов установятся в высокий уровень.

В принципе для записи любого значения в порт достаточно одного регистра ODR, но для того чтобы изменить с его помощью состояние только некоторых каналов порта, потребуется выполнить несколько операций. Вначале нужно будет прочесть состояние регистра ODR, затем модифицировать его и потом записать обратно, чтобы сохранить состояние остальных каналов порта. Таким образом, понадобится три команды, которые потребуют программную память микроконтроллера и время на их выполнение. Использование регистра BSRR позволяет установить или сбросить любой канал порта всего лишь с помощью одной команды. Это экономит память и время микроконтроллера.

Кроме того, с помощью этого регистра можно задать, какой из подтягивающих резисторов будет использоваться в случае настройки канала порта на ввод. Единичное состояние соответствующего разряда регистра подключит вход канала через подтягивающий резистор к плюсовому источнику питания, а нулевое состояние — к общей цепи.

Регистр BRR представляет собой 32-разрядный регистр, предназначенный лишь для сброса в нулевое состояние каналов порта, настроенных на вывод. Аналогично регистру вывода ODR, в регистре ввода IDR используется только 16 младших бит из 32. Назначение разрядов регистра IDR представлено в таблице 6.

Таблица 5. Назначение разрядов регистра BSRR

Бит регистра	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Назначение	BR15	BR14	BR13	BR12	BR11	BR10	BR 9	BR 8	BR 7	BR 6	BR 5	BR 4	BR 3	BR 2	BR 1	BR 0
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	BS15	BS 14	BS 13	BS 12	BS 11	BS 10	BS 9	BS 8	BS 7	BS 6	BS 5	BS 4	BS 3	BS 2	BS 1	BS 0

Таблица 6. Назначение разрядов регистра BRR

Бит регистра	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Назначение	Резерв															
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	BR15	BS14	BS13	BR 12	BR 11	BR 10	BR 9	BR 8	BR 7	BR 6	BR 5	BR 4	BR 3	BR 2	BR 1	BR 0

Таблица 7. Назначение разрядов регистра LCKR

Бит регистра	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Назначение	Резерв															LCKK
Бит регистра	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Назначение	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0

Так же как и для регистра BSRR, запись в регистр BRR нулевых бит никакого действия не оказывает. Все разряды данного регистра доступны только для записи.

Данный регистр, в отличие от регистра BSRR, позволяет осуществлять сброс каналов порта без смещения на 16 разрядов. Во многих случаях это облегчает труд программистов.

Наконец, последний регистр LCKR, представляющий собой 32-разрядный регистр, предназначен для блокировки установленной конфигурации. В регистре блокировки LCKR используется 17 младших разрядов из 32, остальные разряды зарезервированы. Назначение всех разрядов регистра LCKR представлено в таблице 7.

Разряды от 0 до 15-го регистра LCKR определяют каналы порта, а разряд 16 с именем LCKK предназначен для осуществления самой операции блокировки конфигурации. Все задействованные биты данного регистра доступны для записи и чтения.

Процедура блокировки осуществляется следующим образом:

- установить бит LCK, соответствующий блокируемому каналу, в единичное состояние;
- выполнить последовательно запись в разряд LCKK значения 1, затем 0 и снова 1;
- прочесть регистр LCKR;
- повторно прочесть регистр LCKR.

Все операции должны быть выполнены последовательно друг за другом. После чего запись в конфигурационные регистры портов CRL и CRH будет заблокирована. Разблокировать ее можно будет только аппаратным сбросом.

Более подробное описание назначения регистров GPIO можно найти в источнике [2].

ПРОГРАММНАЯ ИНИЦИАЛИЗАЦИЯ

Перед началом работы любого порта GPIO STM32 его необходимо настроить. Для этого требуется разрешить тактирование порта, задать его максимальную тактовую частоту и установить режим работы. Включать тактирование порта необходимо, поскольку после сброса оно автоматически отключается для того, чтобы снизить потребление тока и предотвратить нагрев микроконтроллера. Кстати сказать, тактирование изначально отключено не только для всех портов, но и для многих других функциональных блоков.

Ниже приведен пример настройки порта GPIOA на языке Си в режим цифрового симметричного выхода:

```
// Разрешить тактирование порта GPIOA
RCC->APB2ENR |=RCC_APB2ENR_IOPAEN;
// Конфигурирование канала 0
порта GPIOA на вывод
GPIOA->CRL &= ~GPIO_CRL_MODE0; //
Очистить разряды MODE
GPIOA_>CRL |= GPIO_CRL_MODE0_0;
// Задать частоту 10 МГц
GPIOA->CRL &= ~GPIO_CRL_CNF0; //
Обнулить разряды CNF, установив
режим симметричного выхода
```

Для настройки данного канала порта в режим выхода с открытым стоком следует добавить операторы:

```
GPIOA->CRL |= GPIO_CRL_CNF0_0; //
Установить режим выхода с открытым
стоком
```

Поскольку микроконтроллеры ARM Cortex содержат большое количество функциональных блоков и, соответственно, регистров для их инициализации, разработчиками программного обеспечения были созданы специальные структуры данных и библиотеки, предназначенные для облегчения труда программистов. К их числу относится библиотека CMSIS (Cortex Microcontroller Software Interface Standard), которая представляет собой единый стандарт описаний ресурсов микроконтроллеров ARM Cortex.

Программы для ARM-микроконтроллеров, написанные на языке программирования Си, активно используют разнообразные структуры данных и определений из этих библиотек. Это позволяет легко переносить программы, написанные для одного типа микроконтроллера, на другой, поскольку вместо конкретных значений адресов регистров и числовых констант для инициализации используются символьные имена и указатели на них. Такие имена заранее определены в библиотечных файлах для конкретных моделей ARM-микроконтроллеров и эквивалентны конкретным числовым значениям.

В приведенном выше примере как раз используются такие элементы. Например, запись RCC–>APB2ENR представляет собой обращение к регистру APB2ENR из группы регистров тактирования и контроля RCC, а RCC_APB2ENR_IOPAEN является символьной записью значения управляющего бита порта PORTA в регистре APB2ENR. В числовом виде RCC_APB2ENR_IOPAEN выглядит как 0x00000004.

Символьные имена для всех портов STM32, включая приведенные выше, содержатся в файле stm32f10x.h среды разработки Keil.

Перед установкой конфигурации порта поля регистра для нужного нам канала очищаются с помощью обнуления всех разрядов этих полей. Для установки конфигурации порта используется оператор «|» логиче ской функции «ИЛИ», который позволяет установить нужные разряды в поле регистра или оператор «&» логической функции «И» для обнуления конкретных разрядов регистра.

При необходимости одновременной настройки нескольких каналов порта можно использовать объединение нескольких значений с помощью оператора «ИЛИ». Например, задать частоту 10 МГц для PORTA.0 и PORTA.1 можно так:

```
GPIOA->CRL |= GPIO_CRL_MODE0_0 |
GPIO_CRL_MODE1_0
```

После выполнения всех этих действий настройку порта можно считать законченной.

Описанную выше настройку также можно произвести с помощью следующих двух строк:

```
RCC->APB2ENR |= 0x00000004; //
Разрешить тактирование порта GPIOA
GPIOA->CRL = 0x00000001; //
PORTA.0 в режим симметричного
выхода с частотой 10 МГц
```

Однако такая запись труднее поддается анализу и в ней легче совершить ошибку.

Запись данных в регистр любого порта производится классическим способом, с помощью операции присвоения:

```
GPIOA->ODR=0x0001; // Записать
в PORTA значение 0x0001
```

Для установки или сброса нескольких каналов порта можно использовать регистр BSRR, например, так:

```
GPIOA->BSRR=0x00020001; // Установить
канал 0 и сбросить канал 1 порта A
```

Аналогичную процедуру можно выполнить с помощью предопределенных констант:

```
GPIOA->BSRR=GPIO_BSRR_BS0 |
GPIO_BSRR_BR1;
```

Для настройки этого же порта на аналоговый вход применяются следующие операторы:

```
// Конфигурирование канала 0
порта GPIOA как аналоговый вход
GPIOA->CRL &= ~GPIO_CRL_MODE0; //
Очистить разряды MODE регистра CRL
GPIOA->CRL &= ~GPIO_CRL_CNF0; //
Очистить разряды CNF регистра CRL
```

Чтобы настроить порт на дискретный вход, следует добавить оператор:

```
GPIOA->CRL |= GPIO_CRL_CNF0_0; //
Дискретный вход без подтяжки
к шине питания
```

Наконец, чтобы подтянуть этот вход к шине питания, используются операторы:

```
GPIOA->BSRR = GPIO_BSRR_BS0; //
Подтянуть вход к плюсовому потенциалу
```

или

```
GPIOA->BSRR = GPIO_BSRR_BR0; //
Подтянуть вход к нулевому потенциалу
```

Чтение данных порта или состояния конкретного канала производится с помощью регистра IDR. Вот несколько примеров чтения данных:

```
x= GPIOA->IDR; // Присвоить
переменной x данные из порта A
if (GPIOA->IDR & GPIO_IDR_IDR0)
x=1; // Если канал 0 порта A
не сброшен, то x=1
```

С помощью таких простых операторов можно построить любую сложную программу, которая сможет анализировать состояние внешних устройств и управлять ими.

Существуют и другие способы обращения к регистрам STM32, например с применением библиотек отладочного комплекта STM32VLDISCOVERY. Однако все эти способы в конце концов сводятся к простейшим операциям чтения и записи регистров. Естественно, каждый разработчик имеет право сам определить, какой способ ему больше нравится и что использовать.

Литература:

1. *https://www.st.com*
2. *http://www.st.com/web/en/resource/technical/document/reference_manual/CD00246267.pdf*

CNY

* Статья перепечатана из журнала «Современная электроника», № 7, 2013 г., с разрешения редакции, тел. +7 (495) 232-00-87, www.soel.ru