Как работать с периферией микроконтроллера ?

Цель занятия

- Разобраться что называется периферией, какая она бывает и как работает

Периферия микроконтроллера

Типы периферии

Внутренняя

- аппаратные блоки внутри микроконтроллера, которые выполняют специализированные функции вывода-вывода и обработки данных.

Внешняя

- аппаратные блоки, в центре которых находится сам микроконтроллер. Интерфейсами взаимодействия с такой периферией служат различные протоколы: SPI, I2C, UART и т.д. Это могут быть датчики, моторы, внешняя память и т.д.

Типы периферии (по функциям)

- Периферия ввода/вывода
- Периферия памяти
- Периферия времени
- Вычислительная периферия

Периферия ввода/вывода

- Модули, которые позволяют получать или передавать информацию внешним устройствам
- GPIO может быть подключён к кнопке или светодиоду, таким образом мы можем обмениваться информацией с человеком.
- АЦП (аналого-цифровой преобразователь) может быть подключён к аналоговым датчикам, а значит мы можем получать сигнал из некоторой физической системы (давление, температура и т. д.)
- Модули протоколов SPI, I2C, UART, USB служат для обмена данными с цифровыми устройствами: цифровыми датчиками, ЦАП и АЦП, другими микроконтроллерами и другой внешней периферией.

Периферия памяти

- аппаратные модули, которые расширяют объём памяти, доступный процессорным ядрам. Например, это модуль XIP, который даёт процессорному ядру доступ к внешней энергонезависимой памяти с прошивкой.
- модули, способные выполнять операции с памятью. Особым модулем периферии памяти в любом микроконтроллере является модуль DMA.

DMA

- DMA (Direct Memory Access, прямой доступ к памяти) — аппаратный модуль, подключённый к системной шине и способный выполнять операции чтения и записи вместо процессора.
- Этот модуль освобождает процессор от долгих операций копирования и позволяет использовать освободившееся время для более важных задач.
- Обычно он поддерживает несколько режимов:
  - Из периферии в память
  - Из памяти в периферию
  - Из памяти в память

Регистры периферии

Регистры периферии

- Регистр — аппаратная ячейка памяти фиксированной разрядности, способная хранить одно двоичное слово и допускающая операции записи и чтения.
- Двоичное слово — последовательность битов фиксированной длины, которую процессор или периферия обрабатывают как единое целое. Для RP2040 слово в 4 байта.
- Регистры используются для управления периферией

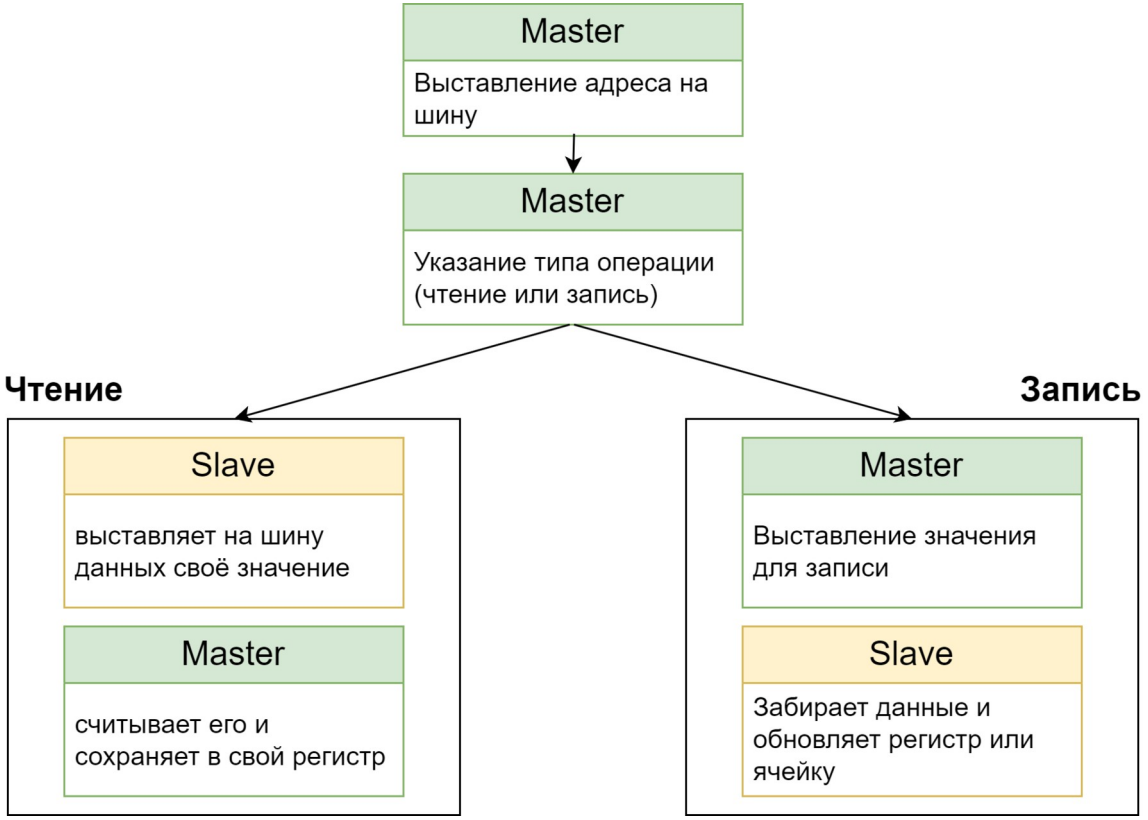
Шина памяти

- Шина (bus) — общий канал связи, к которому подключено несколько устройств и по которому передаётся информация по правилам одного протокола. В каждый момент времени обмен обычно ведут два участника: один передаёт, другой принимает (или один запрашивает, другой отвечает).
- Шина памяти (memory bus) — шина, по которой выполняются операции чтения и записи по адресу. По ней процессор (или модуль DMA) передаёт адрес и тип операции; устройство, «привязанное» к этому адресу (блок памяти или периферийный модуль), возвращает данные при чтении или принимает их при записи.
- Memory-mapped I/O (ввод-вывод, отображённый в память) — способ организации доступа к периферии, при котором регистры периферийных устройств размещены в общем адресном пространстве процессора наряду с оперативной и другой памятью. Обращение к периферии выполняется теми же командами чтения и записи по адресу, что и обращение к памяти; выбор «память или периферия» определяется только адресом и выполняется аппаратно.

Транзакция шины памяти

- Транзакция — одна операция чтения или записи по адресу на шине памяти
- Ведущее устройство (master) — инициатор транзакции
- Ведомое устройство (slave) — устройство, которое по адресу отвечает или принимает данные

Последовательность выполнения транзакции



Карта регистров

- Карта регистров (register map) — описание набора регистров периферийного модуля, отображённых в адресное пространство памяти: базовый адрес модуля, смещение каждого регистра относительно базового, имя, назначение (чтение/запись/только чтение) и смысл битовых полей. Карта регистров задаёт, по какому адресу и каким значением управлять модулем и откуда читать его состояние или данные.

The ADC registers start at a base address of `0x4004c000` (defined as ADC_BASE in SDK).

Offset	Name	Info
0x00	CS	ADC Control and Status
0x04	RESULT	Result of most recent ADC conversion
0x08	FCS	FIFO control and status
0x0c	FIFO	Conversion result FIFO
0x10	DIV	Clock divider. If non-zero, CS_START_MANY will start conversions at regular intervals rather than back-to-back. The divider is reset when either of these fields are written. Total period is 1 + INT + FRAC / 256
0x14	INTR	Raw Interrupts
0x18	INTE	Interrupt Enable
0x1c	INTF	Interrupt Force
0x20	INTS	Interrupt status after masking & forcing

Регистры на примере АЦП RP2040

АЦП RP2040

- АЦП (Analog-to-Digital Converter, аналого-цифровой преобразователь) — модуль, преобразующий аналоговое напряжение на входе в цифровой код фиксированной разрядности.

4.9.1. ADC controller

A digital controller manages the details of operating the RP2040 ADC, and provides additional functionality:

- One-shot or free-running capture mode
- Sample FIFO with DMA interface
- Pacing timer (16 integer bits, 8 fractional bits) for setting free-running sample rate
- Round-robin sampling of multiple channels in free-running capture mode
- Optional right-shift to 8 bits in free-running capture mode, so samples can be DMA'd to a byte buffer in system memory

Карта регистров RP2040

The ADC registers start at a base address of `0x4004c000` (defined as ADC_BASE in SDK).

Offset	Name	Info
0x00	CS	ADC Control and Status
0x04	RESULT	Result of most recent ADC conversion
0x08	FCS	FIFO control and status
0x0c	FIFO	Conversion result FIFO
0x10	DIV	Clock divider. If non-zero, CS_START_MANY will start conversions at regular intervals rather than back-to-back. The divider is reset when either of these fields are written. Total period is 1 + INT + FRAC / 256
0x14	INTR	Raw Interrupts
0x18	INTE	Interrupt Enable
0x1c	INTF	Interrupt Force
0x20	INTS	Interrupt status after masking & forcing

Схема проведения измерения

Включить питание

записать 1 в поле **EN** регистра CS. Если нужен температурный сенсор, дополнительно записать 1 в **TS_EN**.

↓

Выбрать канал

записать в **AINSEL** номер канала (0–4): внешний пин или температурный сенсор.

↓

Запустить преобразование

записать 1 в **START_ONCE** (бит сам сбросится)

↓

Дождаться готовности

опрашивать поле **READY**: пока там 0, идёт преобразование; при 1 можно читать результат.

↓

Прочитать результат

Считать значение из регистра результата АЦП (смещение и имя смотрите в карте регистров в даташите). Получим 12-битный код 0–4095.

Bits	Description	Type	Reset
14:12	**AINSEL**: Select analog mux input. Updated automatically in round-robin mode.	RW	0x0
11	Reserved.	-	-
10	**ERR_STICKY**: Some past ADC conversion encountered an error. Write 1 to clear.	WC	0x0
9	**ERR**: The most recent ADC conversion encountered an error; result is undefined or noisy.	RO	0x0
8	**READY**: 1 if the ADC is ready to start a new conversion. Implies any previous conversion has completed. 0 whilst conversion in progress.	RO	0x0
7:4	Reserved.	-	-
3	**START_MANY**: Continuously perform conversions whilst this bit is 1. A new conversion will start immediately after the previous finishes.	RW	0x0
2	**START_ONCE**: Start a single conversion. Self-clearing. Ignored if start_many is asserted.	SC	0x0
1	**TS_EN**: Power on temperature sensor. 1 - enabled. 0 - disabled.	RW	0x0
0	**EN**: Power on ADC and enable its clock. 1 - enabled. 0 - disabled.	RW	0x0

ant lab

От регистров к SDK

Описание регистра →

```
// Register    : ADC_CS
// Description : ADC Control and Status
#define ADC_CS_OFFSET _u(0x00000000)
#define ADC_CS_BITS   _u(0x001f770f)
#define ADC_CS_RESET  _u(0x00000000)
```

Описание поля регистра →

```
// Field       : ADC_CS_TS_EN
// Description : Power on temperature sensor. 1 - enabled. 0 - disabled.
#define ADC_CS_TS_EN_RESET  _u(0x0)
#define ADC_CS_TS_EN_BITS   _u(0x00000002)
#define ADC_CS_TS_EN_MSB    _u(1)
#define ADC_CS_TS_EN_LSB    _u(1)
#define ADC_CS_TS_EN_ACCESS "RW"
```

Функция работы с регистром →

```
static inline void adc_set_temp_sensor_enabled(bool enable) {
    if (enable)
        hw_set_bits(&adc_hw->cs, ADC_CS_TS_EN_BITS);
    else
        hw_clear_bits(&adc_hw->cs, ADC_CS_TS_EN_BITS);
}
```