Универсальный связной контроллер на базе ESP32-PICO-D4

Подключение радиочастотных и LoRa-модулей по шине SPI

В статье речь пойдет о конструировании универсального связного контроллера на SIP-модуле ESP32-PICO-D4, в частности о подключении радиочастотных и LoRa-модулей по шине SPI.

Дмитрий Доброхотов
support@nostris.com.ua

В настоящее время радиочастотные модули находят широкое применение в устройствах, где необходима передача данных по радиоканалу в разрешенных для гражданского применения диапазонах частот 433 и 868 МГц. Это беспроводные датчики для охранных систем, дистанционное управление различными механизмами, устройства телеметрии и т. д.

Компания Hoperf выпускает однокристальный трансивер CMT2300AW, способный работать в диапазоне частот 127–1020 МГц, а также готовые модули размером 16×16×1,9 мм на основе этого чипа — RFM300W (рис. 1). Примерно за $2 мы получаем готовый трансивер, который подключается по интерфейсу SPI, имеет выходную мощность 25 мВт на частоте 868 МГц, поддерживает такие типы модуляции, как OOK, FSK, MSK,GFSK, GMSK, и позволяет передавать данные со скоростью до 200 кбит/с.

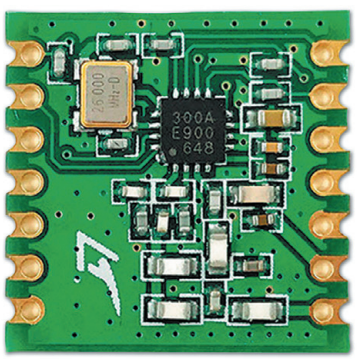

Рис. 1. Модуль RFM300W

Компания Hoperf создала и предоставляет проектировщикам готовую библиотеку для работы трансивера, а также графическую утилиту RFPDK для удобной настройки всех его параметров, которая выдает текстовый файл с настройками внутренних регистров трансивера, легко встраиваемый в программный код. Более подробно трансивер CMT2300AW и предоставляемое с ним программное обеспечение были рассмотрены в [1].

Сегодня активно развиваются и сети с использованием технологии LoRa, которая позволяет построить собственные корпоративные сети в разрешенных безлицензионных диапазонах частот и обеспечить надежную связь между устройствами на расстоянии до 15 км при сверхмалом энергопотреблении. Применение данной технологии предоставляет возможность устройствам с батарейным питанием функционировать автономно несколько лет. Технология LoRa уже получила широкое распространение в областях, где необходимо в режиме реального времени управлять различными устройствами, получать данные от датчиков, передавать небольшие объемы информации. LoRa-сети уже управляют уличным освещением, светофорами, следят за состоянием коммунальных водопроводных и канализационных сетей, высоковольтных линий электропередачи и т. д.

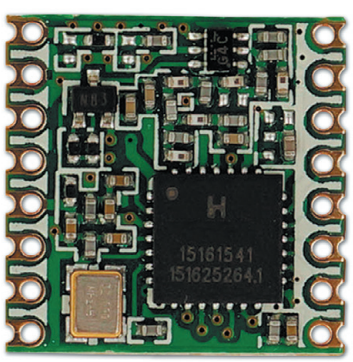

Рис. 2. LoRa-модуль RFM95W

Также компания Hoperf выпускает модуль RFM95W (рис. 2), функционирующий в диапазоне 868 МГц. Программно он полностью совместим с чипсетом SX1276 компании SEMTECH [2]. Модуль RFM95W имеет размеры 16×16×1,8 мм и тоже управляется по интерфейсу SPI.

Рассмотрим подключение данных модулей к модулю ESP32-PICO-D4 по шине SPI.

В своем составе модуль ESP32-PICO-D4 имеет четыре SPI-шины: SPI0, SPI1, HSPI и VSPI. Для работы с внутренней памятью применяется SPI0, а в ESP-IDF на данный момент пользователю доступны две SPI-шины — HSPI и VSPI, имеющие следующие технические характеристики:

- поддержка режимов master, slave, multi-master и multi-slave;
- поддержка полнодуплексного (full-duplex) и полудуплексного (half-duplex) режимов работы;
- поддержка режима QSPI;
- тактовая частота на выходе CLK до 80 МГц;
- четыре режима работы SPI MODE 0–3;
- поддержка 4-проводной версии SPI (4-wire, линии MOSI, MISO, CLK, CS);
- поддержка 3-проводной версии SPI (3-wire), линия MOSI используется для передачи и приема данных;
- до трех ведомых устройств CS0, CS1 и CS2 на каждой шине при работе в режиме master;
- поддержка до 64 байт FIFO;
- наличие встроенного DMA-контроллера.

Большинство встроенной периферии в ESP32 может напрямую подключаться к определенному GPIO, который называется IOMUX. Также существует возможность перенаправления периферийного сигнала на любой другой вывод GPIO, отличный от его вывода в IOMUX, при этом не стоит забывать, что выводы GPIO 34–39 могут работать только как входы. Номера выводов GPIO для IOMUX представлены в таблице.

При использовании IOMUX-выводов частота работы SPI-контроллеров может до-

Таблица. IOMUX-выводы для SPI-контроллеров HSPI и VSPI

Название линии	HSPI	VSPI
	Номер GPIO	
CS0	15	5
SCLK	14	18
MISO	12	19
MOSI	13	23
QUADWP	2	22
QUADHD	4	21

стигать 80 МГц. Если применяются другие GPIO, отличные от IOMUX, частота работы SPI-контроллеров не превышает 40 МГц.

Более подробную информацию о шине SPI модуля ESP32-PICO-D4, в том числе и описание внутренних регистров, можно найти в [3, 4].

Для работы в режиме master необходимо подключить библиотеку driver/spi_master.h, а в режиме slave — driver/spi_slave.h. Эти библиотеки входят в состав ESP-IDF.

Начинать работу с шиной SPI нужно с установки в структуре bus_config выводов MOSI, MISO, CLK и других параметров. Если поле не используется, следует установить ему значение, равное –1. Инициализация шины SPI происходит после вызова функции spi_bus_initialize ().

Далее требуется установить параметры подключаемого slave-устройства в структуре spi_device_interface_config_t и создать spi_device_handle_t (дескриптор) подключаемого устройства. Чтобы подключить slave-устройство, надо вызвать функцию spi_bus_add_device (). Пример кода инициализации представлен в листинге 1.

Листинг 1

```
#define SPI_MODE 0
#define SPI_CLOCK 4000000 // 4 MHz
#define host HSPI_HOST
#define PIN_NUM_MISO 34
#define PIN_NUM_MOSI 2
#define PIN_NUM_CLK 15
#define PIN_NUM_CS 5
.....
spi_bus_config_t buscfg={
.miso_io_num=PIN_NUM_MISO, // номер вывода MISO
или -1 при 3-wire
.mosi_io_num=PIN_NUM_MOSI, // номер вывода MOSI
.sclk_io_num=PIN_NUM_CLK, // номер вывода CLK
.quadwp_io_num=-1, // номер вывода QUADWP
.quadhd_io_num=-1, // номер вывода QUADHD
.max_transfer_sz=0 // максимальный размер данных,
0 — 4096 байт
};
spi_bus_initialize(host, &buscfg, 1);
spi_device_handle_t spi;
spi_device_interface_config_t devcfg={
.command_bits = 0, //0-16
.address_bits = 0, //0-64
.dummy_bits = 0, //пауза между address phase и data
phase
.mode = SPI_MODE,
.duty_cycle_pos = 128, //1-256, длительность положи-
тельного периода
//сигнала CLK
.cs_ena_pretrans = 0, // 0 not used
.cs_ena_posttrans = 0, // 0 not used
.clock_speed_hz = SPI_CLOCK,
.spics_io_num = PIN_NUM_CS,
.flags = 0, //или (SPI_DEVICE_3WIRE | SPI_DEVICE_
HALFDUPLEX)
  //для случая, когда MISO и MOSI на одном
  //выводе MOSI
.queue_size = 1, //размер очереди транзакций
.pre_cb = NULL, //вызов CALLBACK перед транзакцией
.post_cb = NULL; // вызов CALLBACK после транзакции
spi_bus_add_device(host, &devcfg, &spi);
```

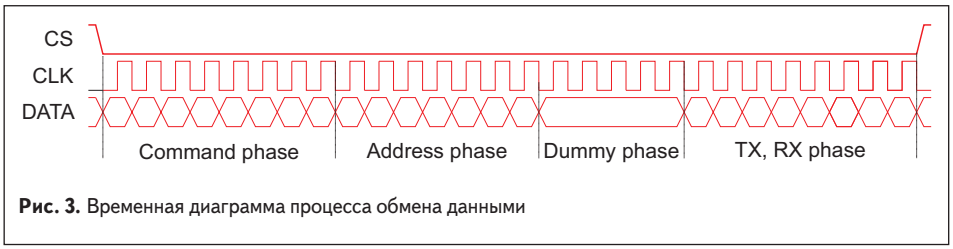

Рис. 3. Временная диаграмма процесса обмена данными

У модуля RFM300 прием и передача данных осуществляются по одному выводу SDIO [5]. Для подсоединения данного модуля к SPI-шине модуля ESP32-PICO-D4 необходимо включить прием и передачу данных по выводу MOSI. Для этого нужно произвести следующие настройки: выводу MISO присваиваем значение –1, а в структуре spi_device_interface_config_t в поле .flags устанавливаем два флага — SPI_DEVICE_3WIRE | SPI_DEVICE_HALFDUPLEX. С такими настройками можно подключать любое устройство, управляемое по 3-wire SPI.

На рис. 3 показана временная диаграмма процесса обмена данными по шине SPI в модуле ESP32-PICO-D4.

Процесс обмена данными состоит из нескольких фаз: Command phase, Address phase, Dummy phase, RX и TX phase. В течение Command phase и Address phase данные только передаются в шину, ничего не считывая взамен. Для выполнения транзакции надо заполнить одну или несколько структур spi_transaction_t, d и отправить их с использованием прерываний (interrupt transactions) или методом последовательной передачи (polling transactions).

С применением polling trans-actions транзакция происходит с помощью функций spi_device_polling_transmit () или spi_device_polling_start () и spi_device_polling_end ().

В листинге 2 представлен фрагмент программы для отправки и приема данных с помощью функции spi_device_polling_transmit ().

Листинг 2

```
esp_err_t SPI_write_Byte(spi_device_handle_t handle,
size_t length, uint8_t *data)
{
spi_transaction_t t = {
.flags = 0, //флаги
.cmd = 0, //команда, 0 если нет
.addr = 0, //адрес, 0 если нет
.length = length * 8, //длина передаваемых данных
cmd+addr+tx, bit
.rxlength = 0, //длина принимаемых данных, bit
.user = 0, //Можно использовать для сохранения
ID-транзакции
.tx_buffer = *data, //указатель на передаваемые данные,
NULL если нет
.rx_buffer = NULL, //указатель на передаваемые данные,
NULL если нет
};
esp_err_t err = spi_device_polling_transmit(handle, &t);
return err;
}
esp_err_t SPI_read_Byte(spi_device_handle_t handle,
size_t length, uint8_t *data)
{
if(length == 0) return ESP_ERR_INVALID_SIZE;
spi_transaction_t t = {
.flags = 0,
.cmd = 0,
.addr = 0,
.length = 0,
.rxlength = length * 8,
.user = 0,
.tx_buffer = NULL,
.rx_buffer = data,
};
esp_err_t err = spi_device_polling_transmit(handle, &t);
return err;
}
```

При interrupt transactions передача данных может происходить синхронно через функцию spi_device_transmit () или поочередно с помощью функций постановки в очередь spi_device_queue_tran () и проверки результата spi_device_get_trans_result ().

Если размер передаваемых данных 32 бит или менее, то они могут быть сохранены в самой структуре транзакции spi_transaction_t и нет необходимости выделять под них отдельный буфер. Для передачи данных следует использовать .tx_data [4] и установить флаг .f lags = SPI_USE_TXDATA, а для получения — .rx_data [4] и установить .f lags = SPI_USE_RXDATA. При этом tx_buffer и rx_buffer не стоит применять, поскольку они ссылаются на те же области памяти, что и tx_data и rx_data.

Более подробную информацию о работе с шиной SPI в модулях на основе ESP32 можно найти в документации на ESP-IDF в разделе API Reference [6]. ■

Литература

1. Гаевский С. Новый интегрированный трансивер CMT2300AW — универсальное решение для передачи данных на короткие дистанции // CHIP NEWS Украина. 2018. № 3.
2. Хаузер Ш., Колпаков А. 100% SiC или гибрид? // CHIP NEWS Украина. 2017. № 2.
3. www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf
4. www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
5. www.hoperf.com/data/upload/portal/20190307/CMT2300A%20Datasheet.pdf
6. www.docs.espressif.com/projects/esp-idf/en/latest/api-reference/peripherals/spi_master.html