что делает этот код?

unsigned char counter; for(counter=0; counter<8; counter++) { SPI\_SCK\_0; if((byte&0x80)==0) { SPI\_SDA\_0; } else SPI\_SDA\_1; byte=byte<<1; SPI\_SCK\_1; SPI\_SCK\_0; где: #define SPI\_SCK\_0 GPIOB->BRR=0X0020 #define SPI\_SCK\_1 GPIOB->BSRR=0X0020 #define SPI\_SDA\_0 GPIOB->BRR=0X0040 #define SPI\_SDA\_1 GPIOB->BSRR=0X0040

Этот код реализует **программную передачу данных по интерфейсу SPI** (Serial Peripheral Interface) с использованием GPIO-пинов микроконтроллера. Давайте разберем его пошагово.

**Основная задача кода**

Код передает один байт данных (byte) по SPI, используя два GPIO-пина:

* SPI\_SCK (Serial Clock) — тактовый сигнал SPI.
* SPI\_SDA (Serial Data) — линия данных SPI.

**Определения макросов**

Макросы используются для управления состоянием GPIO-пинов:

* SPI\_SCK\_0 и SPI\_SCK\_1 управляют состоянием пина SCK

(PB5, 5-й пин порта GPIOB).

* + SPI\_SCK\_0 устанавливает PB5 в низкий уровень (0).
  + SPI\_SCK\_1 устанавливает PB5 в высокий уровень (1).
* SPI\_SDA\_0 и SPI\_SDA\_1 управляют состоянием пина SDA

(PB6, 6-й пин порта GPIOB).

* + SPI\_SDA\_0 устанавливает PB6 в низкий уровень (0).
  + SPI\_SDA\_1 устанавливает PB6 в высокий уровень (1).

**Разбор кода**

1. **Инициализация переменной:**

unsigned char counter;

* + Создается переменная counter, которая будет использоваться для отслеживания количества переданных битов.

1. **Цикл передачи 8 бит:**

for(counter = 0; counter < 8; counter++)

* + Цикл выполняется 8 раз, чтобы передать все 8 бит байта.

1. **Установка тактового сигнала в низкий уровень:**

SPI\_SCK\_0;

* + Пин SCK (PB5) переводится в низкий уровень (0), что означает начало тактового импульса.

1. **Проверка старшего бита данных:**

if((byte & 0x80) == 0)

* + Проверяется значение старшего бита (7-го бита) байта byte:
    - Если старший бит равен 0, выполняется SPI\_SDA\_0 (установка PB6 в низкий уровень).
    - Если старший бит равен 1, выполняется SPI\_SDA\_1 (установка PB6 в высокий уровень).

1. **Сдвиг данных:**

byte = byte << 1;

* + Байт byte сдвигается влево на 1 бит, чтобы следующий бит стал старшим.

1. **Генерация тактового импульса:**

SPI\_SCK\_1;

SPI\_SCK\_0;

* + Пин SCK (PB5) переводится в высокий уровень (1), а затем снова в низкий уровень (0), чтобы сгенерировать тактовый импульс. Это сигнализирует устройству, что бит данных готов.

**Как работает передача данных**

* Код передает байт данных побитово, начиная со старшего бита (MSB — Most Significant Bit).
* Для каждого бита:
  1. Устанавливается значение на линии данных (SDA) в зависимости от текущего бита.
  2. Генерируется тактовый импульс (SCK), чтобы устройство-получатель могло прочитать бит.

**Пример работы**

Предположим, byte = 0b10101010:

1. На первой итерации:
   * Старший бит 1, поэтому SDA = 1.
   * Генерируется тактовый импульс.
2. На второй итерации:
   * Следующий бит 0, поэтому SDA = 0.
   * Генерируется тактовый импульс.
3. И так далее, пока не будут переданы все 8 бит.

**Итог**

Этот код реализует **программную эмуляцию SPI** для передачи одного байта данных. Он управляет состоянием GPIO-пинов (SCK и SDA) для генерации тактовых импульсов и передачи битов данных. Это полезно, если аппаратный модуль SPI недоступен или требуется более гибкое управление.

Что делает этот код?

void TFT\_SEND\_CMD(unsigned char o\_command)

{

SPI\_DC\_0;

SPI\_CS\_0;

SPI\_SendByte(o\_command);

SPI\_CS\_1;

}

Этот код предназначен для отправки команды на дисплей с интерфейсом SPI (например, TFT-дисплей). Давайте разберем его пошагово.

**Основная задача кода**

Функция TFT\_SEND\_CMD отправляет команду на TFT-дисплей, используя интерфейс SPI. Она управляет сигналами DC (Data/Command), CS (Chip Select) и вызывает функцию SPI\_SendByte для передачи команды.

**Разбор кода**

1. **Сигнал DC (Data/Command):**

SPI\_DC\_0;

* + Устанавливает сигнал DC в низкий уровень (0).
  + Сигнал DC используется для указания типа передаваемых данных:
    - DC = 0: Передается **команда**.
    - DC = 1: Передаются **данные**.
  + В данном случае, поскольку передается команда, DC устанавливается в 0.

1. **Сигнал CS (Chip Select):**

SPI\_CS\_0;

* + Устанавливает сигнал CS в низкий уровень (0).
  + Сигнал CS активирует устройство (TFT-дисплей), указывая, что с ним начинается обмен данными.

1. **Передача команды:**

SPI\_SendByte(o\_command);

* + Вызывается функция SPI\_SendByte, которая передает один байт данных (в данном случае команду) по интерфейсу SPI.
  + Параметр o\_command — это команда, которую нужно отправить на дисплей.

1. **Деактивация устройства:**

SPI\_CS\_1;

* + Устанавливает сигнал CS в высокий уровень (1).
  + Это деактивирует устройство, завершая обмен данными.

**Как это работает**

* Когда TFT-дисплей получает команду, он интерпретирует ее как инструкцию для выполнения определенного действия (например, настройка режима дисплея, очистка экрана и т.д.).
* Сигнал DC указывает, что передается именно команда, а не данные.
* Сигнал CS активирует дисплей на время передачи команды.

**Пример использования**

Предположим, вы хотите отправить команду 0x2A (например, для настройки адреса столбцов):

TFT\_SEND\_CMD(0x2A);

* Сигнал DC устанавливается в 0 (передача команды).
* Сигнал CS активируется (0).
* Команда 0x2A передается через SPI.
* Сигнал CS деактивируется (1).

**Итог**

Функция TFT\_SEND\_CMD отправляет команду на TFT-дисплей, используя интерфейс SPI. Она управляет сигналами DC и CS, чтобы указать тип передаваемых данных (команда) и активировать/деактивировать устройство. Это стандартный способ взаимодействия с дисплеями через SPI.

Что делает этот код?

void TFT\_SEND\_DATA(unsigned char o\_data)

{

SPI\_DC\_1;

SPI\_CS\_0;

SPI\_SendByte(o\_data);

SPI\_CS\_1;

}

Этот код отправляет данные на TFT-дисплей через интерфейс SPI (Serial Peripheral Interface). Давайте разберем его по шагам:

1. **SPI\_DC\_1;**  
   Эта строка устанавливает линию данных/команды (Data/Command) в состояние "1", что указывает на то, что следующие данные, передаваемые по SPI, будут интерпретироваться как данные для отображения на экране (а не как команда для управления дисплеем).
2. **SPI\_CS\_0;**  
   Эта строка активирует выбор устройства (Chip Select) на шине SPI, устанавливая линию CS (Chip Select) в состояние "0". Это указывает на то, что устройство (в данном случае TFT-дисплей) готово к приему данных.
3. **SPI\_SendByte(o\_data);**  
   Эта строка отправляет один байт данных (o\_data) на TFT-дисплей через интерфейс SPI. Функция SPI\_SendByte, вероятно, реализует низкоуровневую передачу данных по SPI.
4. **SPI\_CS\_1;**  
   Эта строка деактивирует выбор устройства, устанавливая линию CS в состояние "1". Это завершает передачу данных и освобождает шину SPI для других устройств.

### Итог:

Этот код отправляет один байт данных на TFT-дисплей, используя интерфейс SPI. Он управляет линиями данных/команды (DC) и выбора устройства (CS), чтобы корректно передать данные на дисплей.