Note

This documentation is not for the latest stable release version. The latest stable version is v5.3.1

SD SPI Host Driver

[中文]

Overview

The SD SPI host driver allows communication with one or more SD cards using the SPI Master driver, which utilizes the SPI host. Each card is accessed through an SD SPI device, represented by an SD SPI handle `sdspi_dev_handle_t`, which returns when the device is attached to an SPI bus by calling `sdspi_host_init_device()`. It is important to note that the SPI bus should be initialized beforehand by `spi_bus_initialize()`.

This driver's naming pattern was adopted from the SDMMC Host Driver due to their similarity. Likewise, the APIs of both drivers are also very similar.

SD SPI driver that accesses the SD card in SPI mode offers lower throughput but makes pin selection more flexible. With the help of the GPIO matrix, an SPI peripheral's signals can be routed to any ESP32 pin. Otherwise, if an SDMMC host driver is used (see SDMMC Host Driver) to access the card in SD 1-bit/4-bit mode, higher throughput can be reached while requiring routing the signals through their dedicated IO_MUX pins only.

With the help of SPI Master Driver the SD SPI host driver based on, the SPI bus can be shared among SD cards and other SPI devices. The SPI Master driver will handle exclusive access from different tasks.

The SD SPI driver uses software-controlled CS signal.

How to Use

Firstly, use the macro `SDSPI_DEVICE_CONFIG_DEFAULT` to initialize the structure `sdspi_device_config_t`, which is used to initialize an SD SPI device. This macro will also fill in the default pin mappings, which are the same as the pin mappings of the SDMMC host driver. Modify the host and pins of the structure to desired value. Then call `sdspi_host_init_device` to initialize the SD SPI device and attach to its bus.

Then use the `SDSPI_HOST_DEFAULT` macro to initialize the `sdmmc_host_t` structure, which is used to store the state and configurations of the upper layer (SD/SDIO/MMC driver). Modify the `slot` parameter of the structure to the SD SPI device SD SPI handle just returned from `sdspi_host_init_device`. Call `sdmmc_card_init` with the `sdmmc_host_t` to probe and initialize the SD card.

Now you can use SD/SDIO/MMC driver functions to access your card!

Other Details

Only the following driver's API functions are normally used by most applications:

- `sdspi_host_init()`
- `sdspi_host_init_device()`
- `sdspi_host_remove_device()`
- `sdspi_host_deinit()`

Other functions are mostly used by the protocol level SD/SDIO/MMC driver via function pointers in the `sdmmc_host_t` structure. For more details, see SD/SDIO/MMC Driver.

> **Note**
>
> SD over SPI does not support speeds above `SDMMC_FREQ_DEFAULT` due to the limitations of the SPI driver.

> **Warning**
>
> If you want to share the SPI bus among SD card and other SPI devices, there are some restrictions, see Sharing the SPI Bus Among SD Cards and Other SPI Devices.

Related Docs

- Sharing the SPI Bus Among SD Cards and Other SPI Devices

API Reference

Header File

- components/driver/spi/include/driver/sdspi_host.h
- This header file can be included with:

```
#include "driver/sdspi_host.h"
```

- This header file is a part of the API provided by the `driver` component. To declare that your component depends on `driver`, add the following to your CMakeLists.txt:

```
REQUIRES driver
```

or

```
PRIV_REQUIRES driver
```

Functions

esp_err_t **sdspi_host_init**(void)

Initialize SD SPI driver.

Note

This function is not thread safe

Returns:

- ESP_OK on success
- other error codes may be returned in future versions

esp_err_t **sdspi_host_init_device**(*const* sdspi_device_config_t *dev_config, sdspi_dev_handle_t *out_handle)

Attach and initialize an SD SPI device on the specific SPI bus.

Note

This function is not thread safe

Note

Initialize the SPI bus by `spi_bus_initialize()` before calling this function.

Note

The SDIO over sdspi needs an extra interrupt line. Call `gpio_install_isr_service()` before this function.

Parameters:
- **dev_config** -- pointer to device configuration structure
- **out_handle** -- Output of the handle to the sdspi device.

Returns:
- ESP_OK on success
- ESP_ERR_INVALID_ARG if sdspi_host_init_device has invalid arguments
- ESP_ERR_NO_MEM if memory can not be allocated
- other errors from the underlying spi_master and gpio drivers

esp_err_t **sdspi_host_remove_device**(sdspi_dev_handle_t handle)

Remove an SD SPI device.

Parameters: **handle** -- Handle of the SD SPI device

Returns: Always ESP_OK

esp_err_t **sdspi_host_do_transaction**(sdspi_dev_handle_t handle, sdmmc_command_t *cmdinfo)

Send command to the card and get response.

This function returns when command is sent and response is received, or data is transferred, or timeout occurs.

Note

This function is not thread safe w.r.t. init/deinit functions, and bus width/clock speed configuration functions. Multiple tasks can call sdspi_host_do_transaction as long as other sdspi_host_* functions are not called.

Parameters:
- **handle** -- Handle of the sdspi device
- **cmdinfo** -- pointer to structure describing command and data to transfer

Returns:
- ESP_OK on success
- ESP_ERR_TIMEOUT if response or data transfer has timed out
- ESP_ERR_INVALID_CRC if response or data transfer CRC check has failed
- ESP_ERR_INVALID_RESPONSE if the card has sent an invalid response

---

esp_err_t **sdspi_host_set_card_clk**(sdspi_dev_handle_t host, uint32_t freq_khz)

Set card clock frequency.

Currently only integer fractions of 40MHz clock can be used. For High Speed cards, 40MHz can be used. For Default Speed cards, 20MHz can be used.

Note

This function is not thread safe

Parameters:
- **host** -- Handle of the sdspi device
- **freq_khz** -- card clock frequency, in kHz

Returns:
- ESP_OK on success
- other error codes may be returned in the future

---

esp_err_t **sdspi_host_get_real_freq**(sdspi_dev_handle_t handle, int *real_freq_khz)

Calculate working frequency for specific device.

Parameters:
- **handle** -- SDSPI device handle
- **real_freq_khz** -- **[out]** output parameter to hold the calculated frequency (in kHz)

Returns:
- ESP_ERR_INVALID_ARG : `handle` is NULL or invalid or `real_freq_khz` parameter is NULL
- ESP_OK : Success

esp_err_t **sdspi_host_deinit**(void)

Release resources allocated using sdspi_host_init.

Note

This function is not thread safe

Returns:

- ESP_OK on success
- ESP_ERR_INVALID_STATE if sdspi_host_init function has not been called

esp_err_t **sdspi_host_io_int_enable**(sdspi_dev_handle_t handle)

Enable SDIO interrupt.

Parameters: **handle** -- Handle of the sdspi device

Returns:

- ESP_OK on success

esp_err_t **sdspi_host_io_int_wait**(sdspi_dev_handle_t handle, TickType_t timeout_ticks)

Wait for SDIO interrupt until timeout.

Parameters:

- **handle** -- Handle of the sdspi device
- **timeout_ticks** -- Ticks to wait before timeout.

Returns:

- ESP_OK on success

Structures

struct **sdspi_device_config_t**

Extra configuration for SD SPI device.

Public Members

spi_host_device_t **host_id**

SPI host to use, SPIx_HOST (see spi_types.h).

gpio_num_t **gpio_cs**

GPIO number of CS signal.

gpio_num_t **gpio_cd**

GPIO number of card detect signal.

gpio_num_t **gpio_wp**

GPIO number of write protect signal.

gpio_num_t **gpio_int**

GPIO number of interrupt line (input) for SDIO card.

bool **gpio_wp_polarity**

GPIO write protect polarity 0 means "active low", i.e. card is protected when the GPIO is low; 1 means "active high", i.e. card is protected when GPIO is high.

Macros

SDSPI_DEFAULT_HOST

SDSPI_DEFAULT_DMA

SDSPI_HOST_DEFAULT()

Default sdmmc_host_t structure initializer for SD over SPI driver.

Uses SPI mode and max frequency set to 20MHz

'slot' should be set to an sdspi device initialized by `sdspi_host_init_device()`.

SDSPI_SLOT_NO_CS

indicates that card select line is not used

SDSPI_SLOT_NO_CD

indicates that card detect line is not used

SDSPI_SLOT_NO_WP

indicates that write protect line is not used

SDSPI_SLOT_NO_INT

indicates that interrupt line is not used

SDSPI_IO_ACTIVE_LOW

SDSPI_DEVICE_CONFIG_DEFAULT()

Macro defining default configuration of SD SPI device.

Type Definitions

typedef int sdspi_dev_handle_t

Handle representing an SD SPI device.