Linux DM9051 Driver r2503_v3.9.1 user's guide

Configure to match the working kernel version

- For variated individual kernal version
    - Select one of the following pre-defined kernel versions, DM9051_KERNEL_5_10/ DM9051_KERNEL_5_15/ DM9051_KERNEL_6_1/ DM9051_KERNEL_6_6

  An example:

  ```
  #define KERNEL_BUILD_CONF DM9051_KERNEL_6_6
  ```

Optional settings by the structure definition

To manage various settings and parameters for the DM9051 SPI Fast Ethernet driver. Here is a detailed breakdown of the structure **driver_config** and its components:

```
struct driver_config
{
    const char *release_version;
    int interrupt;
    int mid;
    struct mod_config
    {
        char *test_info;
        int skb_wb_mode;
        int tx_mode;
        int checksuming;
        struct align_config
        {
            int burst_mode;
            size_t tx_blk;
            size_t rx_blk;
        } align;
    } mod[MODE_NUM];
};
```

Overview

The **struct driver_config** is a configuration structure in the DM9051 driver, likely used to store various settings and parameters related to the driver's operation. It includes fields for the release version, interrupt number, nested mod_config structure selection index, and an array of **mod_config** structures.

Members

1. **release_version**
    - Type: const char *
    - Description: A string representing the release version of the driver.
2. **interrupt**
    - Type: int
    - Description: An integer representing the interrupt mode. This could be one of several predefined modes such as **MODE_POLL**, **MODE_INTERRUPT**, or **MODE_INTERRUPT_CLKOUT**.
3. **mid**
    - Type: int
    - Description: An integer representing the mode index. This is used to select a specific configuration from the **mod** array.
4. **mod**
    - Type: struct mod_config[MODE_NUM]
    - Description: An array of **struct mod_config** structures, each representing a different configuration mode. The size of the array is defined by **MODE_NUM**.

Nested Structure:

The nested **struct mod_config** is a configuration structure for a specific designated platform.

1. **test_info**
    - Type: char *
    - Description: A string containing information about the test or configuration.
2. **int skb_wb_mode**

- Type: int

```
*   An integer representing the SKB (Socket Buffer) write-back mode.
```

3. **tx_mode**
    - Type: int
    - Description: An integer representing the transmission mode.
4. **checksuming**
    - Type: int
    - Description: An integer representing whether checksum offloading is enabled or not.
5. **align**
    - Type: struct
    - Description: A nested structure containing alignment settings for burst mode.

Nested Nested Structure: align_config

The nested nested anonymous structure containing alignment-related parameters.

1. **burst_mode**
   - Type: int
   - Description: An integer representing whether burst mode is enabled or not.
2. **tx_blk**
   - Type: size_t
   - Description: A size_t value representing the alignment block size for transmission.
3. **rx_blk**
   - Type: size_t
   - Description: A size_t value representing the alignment block size for reception.

Usage

This structure is used to configure various aspects of the DM9051 driver, including encryption settings, skb boundary mode, transmission modes, checksum offloading, and alignment settings for burst mode. The **mid** member is used to select a specific configuration from the **mod** array, allowing for different configurations to be easily switched between.

Example

Here is an example of how this structure might be initialized:

```
const struct driver_config confdata = {
    .release_version = "lnx_dm9051_kt6631_r2503_v3.9.1",
    .interrupt = MODE_POLL,
    .mid = MODE_A,
    .mod = {
        {
            .test_info = "Test in rpi5 bcm2712",
            .skb_wb_mode = SKB_WB_ON,
            .tx_mode = FORCE_TX_CONTI_OFF,
            .checksuming = DEFAULT_CHECKSUM_ON,
            .align = {
                .burst_mode = BURST_MODE_ALIGN,
                .tx_blk = 32,
                .rx_blk = 64
            }
        },
        // Additional configurations for other modes...
    }
};
```

This example initializes the **confdata** structure with specific settings for one mode, including encryption, skb boundary mode, transmission mode, checksum offloading, and alignment settings.