光纤采集卡驱动说明

2018-04-24

驱动加载

驱动文件 `ft_driver.ko` 、配置程序 `config` 和配置文件 `ft.conf` 放置在 `driver` 目录下，通过在 `root` 用户中运行该目录下的脚本 `load_driver.sh` 将驱动加载到系统中，并根据配置文件对设备进行初始化配置，如下所示：

```
[user@localhost driver] $ su
密码:
[root@localhost dirver] $ ./load_driver.sh
loading FIBRE_TEST driver.
Success to load driver
Starting configure FIBRE_TEST board.
parsing file: ft.conf
speed code: 1
syn length: 2
syn kcode: 2
syn value: bc75
idle length: 2
idle kcode: 3
idle value: bc3c
Setting...
Reset channels...
Succeed to configure board
[root@localhost driver] $
```

如果版本没有安装，驱动仍然可以加载到系统，但找不到对应的设备（这种情况是无法正常工作的），此时加载脚本会给出如下提示：

```
Unable to find the FIBRE_TEST device!
Did the driver correctly load?
```

注意：在使用 `root` 用户运行 `load_driver.sh` 的时候如果出现权限不足的情况，通常是由于在拷贝

过程中，该脚本失去了可运行的权限，可以通过以下命令添加权限：

```
[root@localhost dirver] $ chmod +x load_driver.sh
```

设备配置

设备在加载驱动后，会根据配置文件 `ft.conf` 进行初始化配置，配置文件中共定义了7个参数，具体含义如下。

- `speed_code` 速度编码，取值范围1-10，对应速率如下

速度编码	光纤速率Gbps
1	1.6
2	2.0
3	2.4
4	2.5
5	3.125
6	3.2
7	4.0
8	4.8
9	5
10	6.25

- `syn_length` 同步字头字节长度，可设置为2或者4，2为2字节，4为四字节。
- `syn_kcode` 同步字头各字节是否为 K 码标志，4bit 十六进制数，每个 bit 对应一个字节，0为数据，1为K码。
- `syn_value` 同步字头数据，32位十六进制数，当 `syn_length` 为2时，低16位有效， `syn_length` 为4时，32位都有效。例如若字头为 `0xbc75` ，其中 `bc` 为K码， `75` 为数据，则 `syn_length` 设置为2， `syn_kcode` 设置为 `0x2` ， `syn_value` 设置为 `0xbc75` 。

- `idle_length` 空闲码字节长度，可设置为2或者4，2为2字节，4为四字节。
- `idle_kcode` 空闲码各字节是否为 K 码标志，4bit 十六进制数，每个 bit 对应一个字节，0为数据，1为K码。
- `idle_value` 空闲码数据，32位十六进制数，当 `syn_length` 为2时，低16位有效， `syn_length` 为4时，32位都有效。例如若空闲码为 `0xbc3c` ，其中 `bc` 和 `3c` 均为K码，则 `idle_length` 设置为2， `idle_kcode` 设置为 `0x3` ， `idle_value` 设置为 `0xbc3c` 。

设备使用

设备驱动会把4路光纤分别驱动为4个文件，文件名为 `/dev/pcie_ft1` 到 `/dev/pcie_ft4` ，通过对这4个文件的标准读写操作就可以实现对应光纤的数据收发。

注意 这四个文件每一时刻都只有一个可以打开，如果同时打开两个，后一个文件打开操作会失败。

对于数据的接收，同时支持阻塞式和非阻塞式操作，可以在文件打开的时候通过是否使用选项 `O_NONBLOCK` 来控制，默认不使用该选项时为阻塞式接收。发送始终使用阻塞式。

可以通过标准的 Linux 命令 dd 来读取光纤数据并保持到本地文件，具体语句为：

```
dd if:/dev/pcie_ft1 of:test.dat bs:1024000 count:100
```

其中 if 参数为设备对应的文件名，of 参数为需要保存到的本地文件名，bs 参数为单次读操作的读取长度，count 为读取次数，如上命令就是从第一路光纤读取数据保持到 test.dat 文件中，每次读 1MB，读100次，共计 100MB 数据。

示例程序

此外还提供基于 C 语言的光纤数据读取（接收）示例，放置在 test 文件夹下，其中 test_new.c 为示例程序源代码。

在该目录下可以直接运行 make 命令进行编译，而后运行编译生成的 `./test_new` 运行示例程序。

设备控制

光纤采集卡驱动还提供基于标准文件操作 `ioctl` 的设备控制命令， `ioctl` 的定义如下：

```
#include <sys/ioctl.h>
int ioctl(int f_id, unsigned long request, ...);
```

其中， `f_id` 是文件描述符（ `open` 的返回值）， `request` 是控制的指令，光纤采集卡的控制指令定义在 `ft_macro.h` 头文件中：

```
//ioctl commands
enum FT_IOCTL_CMD {
    FT_RESET : 0x100,//reset FPGA
    FT_READ_BAR0_U32,//read an u32 data from bar0
    FT_WRITE_BAR0_U32,//write an u32 data to bar0
    FT_STOP_DMA_RX,//stop DMA receiving
    FT_TASK_MODE,//use real-task mode (no simulation data)
    FT_SIMU_MODE,//use simulation mode

    FT_IOCTL_CMD_END
};
```

`FT_RESET` 用于复位设备， `FT_STOP_DMA_RX` 用于停止接收， `FT_TASK_MODE` 用于使设备处在工作模式（默认）， `FT_SIMU_MODE` 用于使设备处在自测试模式（接收的数据为顺序的递增数），这几个命令都是没有参数的。

`FT_READ_BAR0_U32` 和 `FT_WRITE_BAR0_U32` 用于读写设备的寄存器，需要使用参数，参数类型为结构体 `Bar0Cmd_t` 的指针，同样定义在 `ft_macro.h` 文件夹内：

```
//struct for FT_READ_BAR0_U32 and FT_WRITE_BAR0_U32
struct Bar0Cmd_t {
    uint32_t addr;
    uint32_t *value;
};
```

其中， `addr` 为读写寄存器的地址， `value` 在寄存器值的指针，可以读写的寄存器地址定义在 `ft_macro.h` 文件中。**注意**寄存器操作有风险，请与硬件支持人员确认后操作。