

How to use Timer driver

1. Introduction

ADSP21535 has three Timers(Timer0, Timer1, Timer2) each of which has three modes:

- PWM(Pulse Width Modulation)
- CAP(Pulse Width Count and Capture)
- ECLK (External Event Counter)

The Timer device major/minor numbers are:

timer device	major	minor
Timer1	10	101
Timer2	10	102
Timer3	10	103

The timer0 device name is /dev/timer0, the timer1 device name is /dev/timer1, and the timer2 device name is /dev/timer2.

When the read function is called, the application is locked until the timer interrupt is generated, and the timer device information is returned.

2. Timer device information data structure

```
struct timer_info {
    unsigned short mode;
    unsigned short status;
    unsigned long width;
    unsigned long period;
}
```

mode: save the timer device work mode.

- ADSP_PWM_MODE = PWM(Pulse Width Modulation) mode
- ADSP_CAP_MODE = CAP(Pulse Width Count and Capture) mode
- ADSP_ECLK_MODE = ECLK(External Event Counter) mode

status: same as the timer configuration register, and the status are composed by the following bits.

- CF_PULSE_HI = Positive action pulse
- CF_PERIOD_CNT = Count to end of period
- CF_UART_SEL = UARTy RX selected
- CF_INT_EN = Enable interrupt request

width: same as the device width register.

period: same as the device period register.

3. system call

The timer device driver is designed as a standard character driver.

The following system calls are supported.

The device can be opened by only one process(thread) at a time.

3.1 open: The standard open function call.

```
int open("/dev/timerx", int oflag, /* mode_t mode */...);
```

The open function is used to establish the connection between the Timer device with a file descriptor.

```
- oflag:
  O_RDONLY   Open for reading only
  O_WRONLY   Open for writing only
  O_RDWR    Open for reading and writing
```

USAGE(Timer1):

```
-----
```

```
int fd;

fd = open("/dev/timer1", O_RDONLY, 0);
...
close(fd);
```

3.2 close: The standard open function call.

```
int close(int file_handler);
```

The close function is used to disconnect the Timer device with the relevant file descriptor.

USAGE(Timer1):

```
-----
```

```
int fd;

fd = open("/dev/timer1", O_RDONLY, 0);
...
close(fd);
```

3.3 ioctl: the standard ioctl function call(refer to section 3).

```
int ioctl(int file_handler, int request, /* arg */...);
```

The ioctl command is used to configure the Timer device.

USAGE(Timer1):

```
-----
```

```
int fd;
int ret;
struct timer_info tinfo;

fd = open("/dev/timer1", O_RDONLY, 0);
.....
// the ioctl command CMD_TIMER_SETMODE is used to configure the timer1 device.
// about the detail ioctl command, refer to section 4.
ret = ioctl(fd, CMD_TIMER_SETMODE, &tinfo);
...
close(fd);
```

3.4 read: the standard read function call.

```
ssize_t read(int file_handler, void *buf, size_t nbytes);
```

The timer device read functions only support the timer information

read operations. When the read function is called, the timer device information (struct timer_info) is returned.

USAGE(Timer1):

```
int fd;
int ret;
struct timer_info tinfo;
```

```
fd = open("/dev/timer1", O_RDONLY, 0);
// call the read function to get the Timer parameters.
ret = read(fd, &tinfo, sizeof(tinfo));
.....
close(fd);
```

4. Timer device ioctl

CMD_TIMER_ENABLE: This ioctl does not need an argument, and it can be used to enable the timer device.

CMD_TIMER_DISABLE: This ioctl does not need an argument, and it can be used to disable the timer device.

CMD_TIMER_RESET: This ioctl does not need an argument, and it can be used to disable the timer device and clear all timer current information.

CMD_TIMER_SETMODE: This ioctl needs one argument (struct timer_info *), which can be used to set the timer device.

CMD_TIMER_GETINFO: This ioctl needs one argument (struct timer_info *), which can be used to get the timer device information.

5. Caution
