Driver I/O Library

5 декабря 2005 г.

Оглавление

Глава 1

Библиотека drvclient

Данная библиотека предоставляет интерфейс связи с драйверами Устройств Связи с Объектами (УСО) и позволяет читать/изменять состояния каналов, а также получать информацио о конфигурации драйвера.

1.1 drv_open()

Открытие соединения с драйвером.

Синтаксис:

```
drv_t drv_open(char* filename);
```

Аргументы:

- filename — имя драйвера устройства;

Библиотека:
libdrvio

Описание:
Открывает соединение с драйвером, для последующего использования.

Возвращаемые значение:

- NULL — Невозможно открыть драйвер, при этом в глобальную переменную errno заносится код ошибки
- иначе — Объект драйвера

Примеры:

```
#include <stdio.h>
#include <errno.h>
#include <S_libdrvio.h>

int main(int argc, char* argv[])
{
    char *file = "/dev/PCL-818H";
    drv_t drv;

    ...
    /* open driver */
```

```
    if ((drv = drv_open(file)) == NULL) {
        fprintf(stderr,"Can't open driver (%s)\n", strerror(errno));
        exit(1);
    }
    ...
    drv_close(drv);

    return (0);
}
```

Смотри также:
drv_close()

1.2 drv_close()

Закрытие соединения с драйвером.

Синтаксис:

```
int drv_close(drv_t drv);
```

Аргументы:

- drv — драйвер;

Библиотека:
libdrvio
Описание:
Закрывает соединение с драйвером, открытым drv_open().

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Невозможно закрыть драйвер, при этом в глобальную переменную *errno* заносится код ошибки;

Примеры:
Смотри drv_open()

1.3 drv_read_info()

Получение конфигурации драйвера.

Синтаксис:

```
int drv_read_info(drv_t drv, drv_info_t *drv_info)
```

Аргументы:

- drv — драйвер;
- drv_info — указатель на область данных, где функция сохранит конфигурацию драйвера.

Библиотека:
libdrvio

Описание:
Эта функция производит чтение конфигурации драйвера и сохраняет ее в структуре *drv_info_t*. Структура содержит такие поля:

- num_di — кол-во дискретных входных каналов;
- num_do — кол-во дискретных выходных каналов;
- num_ai — кол-во аналоговых входных каналов;
- num_ao — кол-во аналоговых выходных каналов;
- num_ci — кол-во счетчиков;
- ios — массив адресов портов ввода/вывода;
- ios_size – массив размеров областей ввода/вывода;
- ios_number — кол-во элементов в массивах ios и ios_size;
- dma — номер DMA канала;
- irq — номер прерывания;
- VendorId, DeviceId — идентификатор производителя и устройства (только для PCI плат)
- pci_dev_idx — номер платы;
- type — тип устройства (pci/isa/other).

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка при получении данных, при этом в глобальную переменную *errno* заносится код ошибки;

1.4 drv_digit_read(), drv_digit_read_s()

Чтение дискретных каналов.

Синтаксис:

```
int drv_digit_read(drv_t drv, int ch1, int ch2, uint32_t *buff);
int drv_digit_read_s(drv_t drv, int ch1, int ch2, uint32_t *buff);
```

Аргументы:

- drv — объект драйвера;
- ch1 — номер первого канала;
- ch2 — номер последнего канала;
- buff — указатель на буфер где будут сохраняться прочитанные данные;

Библиотека:
libdrvio

Описание:
Данная функция производит чтение группы дискретных каналов. Каналы считаются с первого по последний включительно. Если вы хотите прочитать один канал, тогда номер последнего канала должен равняться номеру первого канала.
Функция *drv_ digit_ read_ s()* производит чтение данных через разделяемую память. Эту функцию возможно использовать только, если клиент и драйвер работают на одном узле.

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Ошибка распределения памяти/Нет доступа к разделяемой памяти;
  - EINVAL — Неправильный номер канала;

Смотри также:
drv_digit_write_s(),

Примеры:

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <inttypes.h>
#include <S_libdrvio.h>

int main(int argc, char* argv[])
{
    drv_t   drv;
    char    *file = "/dev/PCL-818H";
    uint32_t   buff[16];
```

```
    int     i, j;

    if ((drv = drv_open(file)) == NULL) {
        fprintf(stderr,"Can't open driver (%s)\n", strerror(errno));
        exit(1);
    }

    for (i = 0; i < 100; i++)
    {
        if (drv_digit_read(drv, 0, 15, buff) < 0)
            break;

        for (j = 0; j <= 15; j++) printf ("%u ", buff[j]);

        printf("\n");

        /* ждать 100 мс */
        usleep(100000);
    }

    drv_close(drv);

    return (0);
}
```

1.5 drv_analog_read(), drv_analog_read_s()

Чтение аналоговых каналов.

Синтаксис:

```
int drv_analog_read(drv_t drv, int ch1, int ch2, float *buff);
int drv_analog_read_s(drv_t drv, int ch1, int ch2, float *buff);
```

Аргументы:

- drv — объект драйвера;
- ch1 — номер первого канала;
- ch2 — номер последнего канала;
- buff — указатель на буфер где будут сохраняться прочитанные данные;

Библиотека:
libdrvio

Описание:
Данная функция производит чтение группы аналоговых каналов. Каналы считаются с первого по последний включительно. Если вы хотите прочитать один канал, тогда номер последнего канала должен равняться номеру первого канала.
Функция *drv_analog_read_s()* производит чтение данных через разделяемую память. Эту функцию возможно использовать только, если клиент и драйвер работают на одном узле.

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Ошибка распределения памяти/Нет доступа к разделяемой памяти;
  - EINVAL — Неправильный номер канала;

Смотри также:
drv_analog_write()

1.6 drv_counter_read(), drv_counter_read_s()

Чтение каналов счетчиков.

Синтаксис:

```
int drv_counter_read(drv_t drv, int ch1, int ch2, uint32_t *buff);
int drv_counter_read_s(drv_t drv, int ch1, int ch2, uint32_t *buff);
```

Аргументы:

- drv — объект драйвера;
- ch1 — номер первого канала;
- ch2 — номер последнего канала;
- buff — указатель на буфер где будут сохраняться прочитанные данные;

Библиотека:
libdrvio

Описание:
Данная функция производит чтение группы каналов счетчиков. Каналы считаются с первого по последний включительно. Если вы хотите прочитать один канал, тогда номер последнего канала должен равняться номеру первого канала.
Функция *drv_counter_read_s()* производит чтение данных через разделяемую память. Эту функцию возможно использовать только, если клиент и драйвер работают на одном узле.

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Ошибка распределения памяти/Нет доступа к разделяемой памяти;
  - EINVAL — Неправильный номер канала;

1.7 drv_digit_write(), drv_digit_write_s()

Запись дискретных каналов.

Синтаксис:

```
int drv_digit_write(drv_t drv, int ch1, int ch2, uint32_t *buff);
int drv_digit_write_s(drv_t drv, int ch1, int ch2, uint32_t *buff);
```

Аргументы:

- drv — объект драйвера;
- ch1 — номер первого канала;
- ch2 — номер последнего канала;
- buff — указатель на буфер где будут сохраняться прочитанные данные;

Библиотека:
libdrvio

Описание:
Данная функция производит запись группы дискретных каналов. Каналы считаются с первого по последний включительно. Если вы хотите прочитать один канал, тогда номер последнего канала должен равняться номеру первого канала.
Функция *drv_digit_write_s()* производит запись данных через разделяемую память. Эту функцию возможно использовать только, если клиент и драйвер работают на одном узле.

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Ошибка распределения памяти/Нет доступа к разделяемой памяти;
  - EINVAL — Неправильный номер канала;

Смотри также:
drv_digit_read()

1.8 drv_analog_write(), drv_analog_write_s()

Запись аналоговых каналов.

Синтаксис:

```
int drv_analog_write(drv_t drv, int ch1, int ch2, float *buff);
int drv_analog_write_s(drv_t drv, int ch1, int ch2, float *buff);
```

Аргументы:

- drv — объект драйвера;
- ch1 — номер первого канала;
- ch2 — номер последнего канала;
- buff — указатель на буфер где будут сохраняться прочитанные данные;

Библиотека:
libdrvio

Описание:
Данная функция производит запись группы аналоговых каналов. Каналы считаются с первого по последний включительно. Если вы хотите прочитать один канал, тогда номер последнего канала должен равняться номеру первого канала.
Функция *drv_analog_write_s()* производит запись данных через разделяемую память. Эту функцию возможно использовать только, если клиент и драйвер работают на одном узле.

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Ошибка распределения памяти/Нет доступа к разделяемой памяти;
  - EINVAL — Неправильный номер канала;

Смотри также:
drv_analog_read()

1.9 drv_shared_status()

Определяет возможность использования функций *_s() для доступа к драйверу.

Синтаксис:

```
int drv_shared_status(drv_t drv)
```

Аргументы:

- drv — объект драйвера;

Библиотека:
libdrvio

Описание:
Эта функция определяет возможность использования функций *_s() для доступа к драйверу.

Возвращаемые значение:

- 0 — доступ к драйверу через разделяемую памят возможен;
- -1 — доступ к драйверу через разделяемую памят не возможен.

Смотри также:
drv_digit_read_s(), drv_analog_read_s(), drv_counter_read_s(), drv_digit_write_s(), drv_analog_write_s(), drv_wait_data_s(), drv_timedwait_data_s()

1.10 drv_wait_data_s()

Ожидание поступления новых данных.

Синтаксис:

```
int drv_wait_data_s(drv_t drv)
```

Аргументы:

- drv — объект драйвера;

Библиотека:
libdrvio

Описание:
Эта функция ожидает поступления новых данных в драйвер. Если данных нет, тогда вызывающий поток блокируется до поступления новых данных. Для чтения данных необходимо использовать функции drv_*_read_s().

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Доступ к разделяемой памяти невозможен;
  - EINVAL — Неправильный параметр drv;

Смотри также:
drv_digit_read_s(), drv_analog_read_s(), drv_counter_read_s(), drv_timedwait_data_s()

1.11 drv_timedwait_data_s()

Ожидание поступления новых данных с ограничение по времени

Синтаксис:

```
int drv_timedwait_data_s(drv_t drv, uint32_t timeout)
```

Аргументы:

- drv — объект драйвера;
- timeout — время ожидания в миллисекундах.

Библиотека:
libdrvio

Описание:
Эта функция ожидает поступления новых данных в драйвер с ограничением по времени. Если данных нет, тогда вызывающий поток блокируется до поступления новых данных и ожидает их в течении *timeout*, после чего разблокируется. Для чтения данных необходимо использовать функции *drv_*_read_s()*.

Возвращаемые значение:

- 0 — Успешное завершение;
- -1 — Ошибка, при этом в глобальную переменную *errno* заносится код ошибки:
  - ENOMEM — Нет доступа к разделяемой памяти;
  - EINVAL — Неправильный параметр drv;
  - ETIMEDOUT – Вышло указанное время ожидания.

Смотри также:
drv_digit_read_s(), drv_analog_read_s(), drv_counter_read_s(), drv_wait_data_s()