

Библиотека функции
для модуля
RS-PC104

Руководство пользователя.
v1.0

Оглавление

Назначение	3
Условия работы	3
Введение	4
Функции	5
RSPC104DIO_Open [QNX 4.25]	5
RSPC104DIO_Close [QNX 4.25]	5
RSPC104DIO_SetData [QNX 4.25]	6
RSPC104DIO_GetData [QNX 4.25]	6
RSPC104DIO_SetMask [QNX 4.25]	7
RSPC104DIO_DefEvent [QNX 4.25]	7
Обработка прерываний при работе в операционной системе QNX 4.25	8

Назначение

Библиотека функций предназначена для предоставления доступа к ресурсам цифрового ввода/вывода плат серии RS-PC104.

Условия работы

Установленное оборудование:

IBM PC совместимый ПК 80386 и выше

Программное обеспечение:

QNX 4.25: компилятор Watcom C++ версии 10.6

Поддерживаемые операционные системы:

QNX 4.25

Поддерживаемые платы:

RS-PC104

Введение

Плата RS-PC104 реализует:

- Шесть COM-портов (USART 16C550C), из которых:
 - Четыре COM-порта с интерфейсом RS232;
 - Два COM-порта конфигурируются для работы с интерфейсом RS232 или интерфейсом RS422/485 (полный дуплекс или полудуплекс);
- Независимая гальваническая развязка каждого COM-порта;
- Скорость передачи, по каждому COM-порту – до 115200 бит/с;
- Полная программная совместимость с USART 16C450;
- Занимает в адресном пространстве ввода/вывода шины PC104 64 адреса;
- Гибко настраиваемый базовый адрес в диапазоне 000h-3C0h с шагом 40h;
- Блок разовых команд, состоящий из 24 разовых команд с индивидуальным программированием каждой команды на вход или на выход;
- Генерация от четырех разовых команд прерывания на шине PC104;

Данная библиотека предназначена для работы с блоком разовых команд. Для работы с COM-портами используется стандартный драйвер (Dev.ser).

Функции

RSPC104DIO_Open [QNX 4.25]

Описание:

Получение доступа к драйверу устройства.

Прототип:

```
int RSPC104DIO_Open(char *name)
```

Входные параметры:

Имя устройства в системе

Возвращаемое значение:

> 0 - доступ к драйверу устройства получен

-1- ошибка открытия драйвера (код ошибки в переменной errno)

Включаемые файлы:

QNX 4.25: #include "rs-pc104-dio_api.h"

RSPC104DIO_Close [QNX 4.25]

Описание:

Закрытие доступа к драйверу устройства

Прототип:

```
void RSPC104DIO_Close(int handle)
```

Входные параметры:

Результат возвращенный функцией RSPC104DIO_Open

Возвращаемое значение:

нет

Включаемые файлы:

QNX 4.25: #include "rs-pc104-dio_api.h"

RSPC104DIO_SetData [QNX 4.25]

Описание:

Выдача данных

Прототип:

int RSPC104DIO_SetData(int handle, unsigned long data)

Входные параметры:

handle – результат возвращенный функцией RSPC104DIO_Open.
data – выдаваемые данные.

Возвращаемое значение:

> 0 – успешное завершение
1 – ошибка

Включаемые файлы:

QNX 4.25: #include "rs-pc104-dio_api.h"

RSPC104DIO_GetData [QNX 4.25]

Описание:

Получение данных

Прототип:

int RSPC104DIO_GetData(int handle, unsigned long *data)

Входные параметры:

handle – результат возвращенный функцией RSPC104DIO_Open.
data – указатель на переменную в которую будут помещены считанные данные.

Возвращаемое значение:

> 0 – успешное завершение
1 – ошибка

Включаемые файлы:

QNX 4.25: #include "rs-pc104-dio_api.h"

RSPC104DIO_SetMask [QNX 4.25]

Описание:

Установка маски определяющей режим работы канала (вход/выход).

Прототип:

```
int RSPC104DIO_SetMask(int handle, unsigned long mask)
```

Входные параметры:

handle – результат возвращенный функцией RSPC104DIO_Open.
mask – маска.

Возвращаемое значение:

> 0 – успешное завершение
1 – ошибка

Включаемые файлы:

QNX 4.25: #include "rs-pc104-dio_api.h"

RSPC104DIO_DefEvent [QNX 4.25]

Описание:

Установка прокси для информирования клиента о прерывании.

Прототип:

```
int RSPC104DIO_DefEvent(int handle, pid_t proxy)
```

Входные параметры:

handle – результат возвращенный функцией RSPC104DIO_Open.
proxy – идентификатор прокси.

Возвращаемое значение:

> 0 – успешное завершение
1 – ошибка

Включаемые файлы:

QNX 4.25: #include "rs-pc104-dio_api.h"

Обработка прерываний при работе в операционной системе QNX 4.25

Пример:

```
#include "rs-pc104-dio_api.h"

pid_t proxy;
int handle;
unsigned char stack[10000];

//-----
void server(void)
{
    //Обработка прерывания
    unsigned msg;

    //Подключаем прокси
    proxy = qnx_proxy_attach(0, NULL, 0, -1);
    if( proxy == -1)
    {
        printf("Ошибка подключения прокси\n");
        exit(-1);
    }
    RSPC104DIO_DefEvent(handle, proxy);

    while(1)
    {
        //Вечный цикл ожидания прерываний
        Receive(proxy, &msg, sizeof(msg));
        RSPC104DIO_GetData(handle, &interrupt_data);
        //
        // .....
        //
    }
}

//-----
int main(int argc, char *argv[])
{
    pid_t receiver;

    /* Открываем модуль */
    handle = RSPC104DIO_Open(argv[1]);
    if( handle == -1 )
    {
        printf("Error open device %s\n", argv[1]);
        return -1;
    }
}
```

```
//Поток для приема прерываний
receiver = tfork(&stack, sizeof(stack), &server, NULL, 0);

//
//Настройка платы для генерации прерываний
//

//
// .....
//
}
```