**Описание драйвера модуля DD64-PCI для Windows.**

Оглавление

1.Введение 4

2.Открытие и закрытие устройства 4

2.1.Открытие устройства по серийному номеру. 4

2.2.Открытие устройства по порядковому номеру. 4

2.3.Закрытие устройства. 5

3.Функции драйвера 5

3.1.Работа с регистрами. 5

3.1.1.Чтение регистра прерываний RI. 5

3.1.2.Чтение регистра таймера TIMER. 5

3.1.3. Запись в регистр таймера TIMER. 6

3.1.4. Запись в регистр адреса RA. 6

3.1.5.Чтение регистра данных RD. 6

3.1.6.Запись в регистр данных RD. 7

3.1.7.Чтение регистра косвенной адресации. 7

3.1.8. Запись в регистр косвенной адресации. 8

3.2.Работа с разовыми командами. 8

3.2.1.Конфигурация и режимы РК. 8

3.2.1.1.Конфигурация выходных РК. 8

3.2.1.2.Конфигурация входных РК. 9

3.2.1.3.Характеристики использования конкретной РК. 9

3.2.1.4.Определение способа управления выходными РК. 10

3.2.1.5.Задание способа управления выходными РК. 10

3.2.2.Управление с использованием регистра выходных РК . 11

3.2.2.1.Установка выходной РК. 11

3.2.2.2.Сброс выходной РК 11

3.2.2.3.Изменение состояния группы выходных РК 12

3.2.3.Управление РК с использованием активной матрицы. 13

3.2.3.1.Определение номера матрицы, заданного аппаратно . 13

3.2.3.2.Установка активной матрицы. 14

3.2.3.3.Чтение номера активной матрицы. 14

3.2.4.Состояние входных и выходных РК. 15

3.2.4.1.Чтение состояния РК 15

3.2.4.2.Чтение состояния группы РК 15

3.2.5.Фильтры выходных РК . 17

3.2.5.1.Создание фильтров. 17

3.2.5.2.Состояние фильтров. 17

3.2.5.3.Конфигурирование портов OHF\_sm подключения внешних OH-фильтров. 18

3.2.5.4.Состояние портов внешних фильтров. 19

3.3.Таймер. 19

3.3.1.Чтение/запись в регистр таймера. 19

3.3.2.Запись в регистр делителя частоты. 19

3.3.3.Чтение регистра делителя частоты. 20

3.3.4.Запись в регистр сравнения таймера. 20

3.4.Работа с ЦАП. 21

3.4.1.Запись в регистр данных ЦАП. 21

3.4.2.Запись в регистр адреса ЦАП. 21

3.4.3.Запись в регистр управления ЦАП. 22

3.4.4.Чтение регистра статуса ЦАП. 22

3.4.5.Чтение регистра конфигурации ЦАП. 23

3.5.Работа с АЦП. 23

3.5.1.Запись/чтение регистров АЦП. 23

3.5.1.1.Запись в регистр данных АЦП1. 23

3.5.1.2.Чтение регистра данных АЦП1. 24

3.5.1.3.Запись в регистр управления АЦП1. 24

3.5.1.4.Чтение регистра состояния АЦП1. 25

3.5.1.5.Запись в регистр данных АЦП2. 25

3.5.1.6.Чтение регистра данных АЦП2. 25

3.5.1.7.Запись в регистр управления АЦП2. 26

3.5.1.8.Чтение регистра состояния АЦП2. 26

3.5.2.Комплексные функции АЦП1. 27

3.5.2.1.Задание диапазонов входных каналов 0..3 АЦП1. 27

3.5.2.2.Задание диапазонов входных каналов 4..7 АЦП1. 28

3.5.2.3.Задание диапазона одного входного канала АЦП1. 28

3.5.2.4.Установка номера и режима работы канала АЦП1 для преобразования. 29

3.5.2.5.Запуск одиночного преобразования АЦП1. 30

3.5.3.Комплексные функции АЦП2. 30

3.5.3.1.Задание диапазонов входных каналов 0..3 АЦП2. 31

3.5.3.2.Задание диапазонов входных каналов 4..7 АЦП2. 32

3.5.3.3.Задание диапазона одного входного канала АЦП2. 32

3.5.3.4.Установка номера и режима работы канала для преобразования АЦП2. 33

3.5.3.5.Запуск одиночного преобразования АЦП2. 33

3.5.4.Чтение регистра конфигурации АЦП. 34

3.6. Прочие функции 35

3.6.1. Чтение параметров модуля. 35

3.6.2. Чтение серийного номера адаптера. 35

3.7.Обслуживание прерываний. 36

# 1.Введение

Драйвер dd64\_PCI.sys предназначен для работы 64-канального модуля дискретного ввода-вывода, ЦАП и АЦП DD64-PCI производства ЗАО «ЭЛКУС» в операционной системе Windows XP. Установка драйвера производится стандартными средствами ОС. (Панель управления – Установка оборудования).

# 2.Открытие и закрытие устройства

В программе пользователя должна быть зарезервирована переменная ***hDD64PCI*** типа ***HANDLE*  -** дескриптор драйвера.

Для открытия устройства можно воспользоваться одним из 2-х предлагаемых способов:

- по серийному номеру;

- по порядковому номеру.

Оба способа открытия требуют включения в проект файла ***intrfaceDD64PCI.h***.

## 2.1.Открытие устройства по серийному номеру.

В комплект поставки ПО входит файл ***OpenDD64bySN.cpp***, содержащий программу открытия устройства по серийному номеру. ( Требуется включение в проект файла ***ioctlDD64.h***). Вызов программы:

***hDD64PCI*** ***=*** ***OpenDeviceBySN******( SerialNumber, &Error);***

где:

***USHORT SerialNumber*** – серийный номер устройства

***DWORD Error*** - возвращаемый код ошибки.

## 2.2.Открытие устройства по порядковому номеру.

В состав поставки драйвера входит файл ***OpenDD64byIndex.cpp***, содержащий программу открытия драйвера по порядковому номеру. Вызов программы:

***hDD64PCI*** ***=*** ***OpenDeviceByIndex ( unit, &Error);***

где:

***DWORD unit***  - номер устройства ( 0, 1, 2..);

***DWORD Error***  - возвращаемый код ошибки.

В случае ошибки эти функции возвращают значение ***hDD64PCI*** , равное INVALID\_HANDLE\_VALUE (0xFFFFFFFF). При успешном завершении в ***hDD64PCI*** возвращается дескриптор драйвера.

## 2.3.Закрытие устройства.

***CloseHandle(hDD64PCI*** ***);***

# 3.Функции драйвера

Коды функций содержатся в файле ***ioctlDD64.h***

## 3.1.Работа с регистрами.

### 3.1.1.Чтение регистра прерываний RI.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_RI,

NULL,

0,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра прерываний.

### 3.1.2.Чтение регистра таймера TIMER.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_TIMER,

NULL,

0,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра таймера.

### 3.1.3. Запись в регистр таймера TIMER.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_TIMER,

&bufInput,

2 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция записывает в регистр таймера код ***USHORT bufInput***;

### 3.1.4. Запись в регистр адреса RA.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_RA,

&bufInput,

2 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция записывает в регистр адреса код ***USHORT bufInput***;

### 3.1.5.Чтение регистра данных RD.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_RD,

NULL,

0,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра данных.

### 3.1.6.Запись в регистр данных RD.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_RD,

&bufInput, .

2 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция записывает в регистр данных код ***USHORT bufInput***;

### 3.1.7.Чтение регистра косвенной адресации.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_INDIRECT\_REGISTR,

&bufInput,

2 ,

&bufOutput,

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра, имеющего адрес, заданный в ***USHORT bufInput***.

Структура зоны косвенной адресации приведена в ТО на модуль.

### 3.1.8. Запись в регистр косвенной адресации.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_INDIRECT\_REGISTR,

&bufInput,

4,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput[2];*** //

Описание

Функция записывает код ***USHORT bufInput*** ***[1]*** в регистр, имеющий в зоне косвенной адресации адрес, заданный в ***USHORT bufInput[0]***.

Структура зоны косвенной адресации приведена в ТО на модуль.

## 3.2.Работа с разовыми командами.

Примечание

1. Все входные и выходные разовые команды (РК) имеют сквозную нумерацию (от 1 до 64). Программным путем направленность конкретной РК (входная или выходная) можно определить с помощью функций раздела 3.2.1.

2. Управление выходными РК может осуществляться с использованием регистра выходных РК (см.п.3.2.2.) или активной матрицы выдачи (см.п.3.2.3.).

3. Задержка времени между записью кода выходной РК и реальным изменением сигнала РК может составлять до 400 мкс.

### 3.2.1.Конфигурация и режимы РК.

#### 3.2.1.1.Конфигурация выходных РК.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_CONFIG\_DO,

NULL,

0,

&bufOutput,

64 ,

&nOutput ,

NULL );

Описание

Функция возвращает информацию о всех выходных РК, задействованных в плате.

Выходные параметры

***UCHAR bufOutput[64];*** // - элемент выходного массива принимает значение 1, если в системе существует выходная РК с номером, равным ***индекс*** ***элемента массива +1***. Например, ***bufOutput[6]=1;*** означает, что в системе существует выходная РК с номером 7. Если ***bufOutput[8]=0;***  , РК номер 9 в системе не существует или она сконфигурирована как входная.

#### 3.2.1.2.Конфигурация входных РК.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_CONFIG\_DI,

NULL,

0,

&bufOutput,

64 ,

&nOutput ,

NULL );

Описание

Функция возвращает информацию о всех входных РК, задействованных в плате.

Выходные параметры

***UCHAR bufOutput[64];*** // - элемент выходного массива принимает значение 1, если в системе существует входная РК с номером, равным ***индекс*** ***элемента массива +1***. Например, ***bufOutput[5]=1;*** означает, что в системе существует входная РК с номером 6. Если ***bufOutput[25]=0;*** РК номер 26 в системе не существует или она сконфигурирована как выходная.

#### 3.2.1.3.Характеристики использования конкретной РК.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_CONFIG\_DS,

&bufInput,

1 ,

&bufOutput,

1 ,

&nOutput ,

NULL );

Описание

Функция определяет, установлена ли РК с заданным номером на плате и если установлена, является ли она входной или выходной.

Входные параметры

***UCHAR bufInput;*** // - номер РК (1-64)

Выходные параметры

***UCHAR bufOutput;*** // 0 – команда не установлена;

// 1 – входная;

// 2 – выходная;

#### 3.2.1.4.Определение способа управления выходными РК.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_MODE\_DO,

NULL ,

0 ,

&bufOutput,

1 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***UCHAR bufOutput*** способ управления выходными РК ***= 1;***  - от активной матрицы (см.п.3.2.3.).

***= 0;***  - от регистра выходных РК (см.п.3.2.2.).

#### 3.2.1.5.Задание способа управления выходными РК.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_MODE\_DO,

&bufInput,

1 ,

NULL , .

0 ,

&nOutput ,

NULL );

Описание

Функция задает способ управления выходными РК:

***UCHAR bufInput = 1;***  - от активной матрицы (см.п.3.2.3.).

***= 0;***  - от регистра выходных РК (см.п.3.2.2.).

### 3.2.2.Управление с использованием регистра выходных РК .

#### 3.2.2.1.Установка выходной РК.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_DO,

&bufInput,

1 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция устанавливает заданную РК в 1.

Входные параметры

***UCHAR bufInput;*** // - номер РК (1-64)

#### 3.2.2.2.Сброс выходной РК

DeviceIoControl (

hDD64PCI ,

DD64PCI\_CLEAR\_DO,

&bufInput,

1 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция переводит заданную РК в 0.

Входные параметры

***UCHAR bufInput;*** // - номер РК (1-64)

#### 3.2.2.3.Изменение состояния группы выходных РК

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_DO\_GROUP,

&bufInput,

9 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция дает возможность одновременно изменять состояние до 8-ми разовых команд, входящих в одну группу.

В таблице указано соответствие номера РК (1..64), номера группы (1..8) и индекса во входном буфере (1..8).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № группы | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Индекс массива |  | | | | | | | |
| 1 | 1 | 9 | 17 | 25 | 33 | 41 | 49 | 57 |
| 2 | 2 | 10 | 18 | 26 | 34 | 42 | 50 | 58 |
| 3 | 3 | 11 | 19 | 27 | 35 | 43 | 51 | 59 |
| 4 | 4 | 12 | 20 | 28 | 36 | 44 | 52 | 60 |
| 5 | 5 | 13 | 21 | 29 | 37 | 45 | 53 | 61 |
| 6 | 6 | 14 | 22 | 30 | 38 | 46 | 54 | 62 |
| 7 | 7 | 15 | 23 | 31 | 39 | 47 | 55 | 63 |
| 8 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 |

Входные параметры

***UCHAR bufInput[9] ;***

Где:

***bufInput[0]***  - номер группы (1-8);

***bufInput[1]***  .. ***bufInput[8]***  - состояние РК после выполнения функции (0/1/0xff); если задан код 0xff , состояние не изменяется.

Пример.

Необходимо изменить состояние 18-й (в 0), 21-й (в 0) и 23-й (в 1) команды. Входной буфер должен иметь вид:

***bufInput[0] = 3;***

***bufInput[1] =*** 0xff***;***

***bufInput[2] =*** 0***;***

***bufInput[3] =*** 0xff***;***

***bufInput[4] =*** 0xff***;***

***bufInput[5] =*** 0***;***

***bufInput[6] =*** 0xff***;***

***bufInput[7] =*** 1***;***

***bufInput[8] =*** 0xff***;***

### 3.2.3.Управление РК с использованием активной матрицы.

Примечание. В функциях матрицы выдачи нумеруются от 1 до 8.

#### 3.2.3.1.Определение номера матрицы, заданного аппаратно .

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_MATRIX\_HD\_NUMBER,

NULL ,

0 ,

&bufOutput,

1,

&nOutput ,

NULL );

Описание

Функция возвращает в ***UCHAR bufOutput***  конфигурацию джамперов X11,X22 (см. ТО).

#### 3.2.3.2.Установка активной матрицы.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_SET\_ACTIVE\_MATRIX,

&bufInput, .

1 ,

NULL,

0,

&nOutput ,

NULL );

Описание

Функция переводитв активное состояние матрицу с номером, заданным в ***UCHAR bufInput (1..8)*** .

Если ***UCHAR bufInput !***= (1..8) – в активное состояние переводится матрица, заданная аппаратно (с помощью джамперов).

#### 3.2.3.3.Чтение номера активной матрицы.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_ACTIVE\_MATRIX,

NULL,

0,

&bufOutput,

1 ,

&nOutput ,

NULL );

Описание

Функция возвращает в ***UCHAR bufOutput*** номер активной матрицы, управляемой программно или 0xff, если активная матрица установлена с помощью джамперов.

### 3.2.4.Состояние входных и выходных РК.

#### 3.2.4.1.Чтение состояния РК

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_DS,

&bufInput,

1 ,

&bufOutput,

1 ,

&nOutput ,

NULL );

Описание

Функция возвращает состояние РК.

Входные параметры

***UCHAR bufInput;*** // - номер РК (1-64)

Выходные параметры

***UCHAR bufOutput;***  // Состояние РК (0/1)

Примечание. Пользователю необходимо учитывать, что для выходных РК функция чтения состояния выходной РК отражает лишь состояние соответствующего бита в регистре выходных разовых команд и не отображает реального состояния сигнала. Например, при установке выходной РК в активное состояние, как описано в п.3.2.2.1. значение состояния также будет активным, но сигнал на выходе каскада РК появится лишь спустя некоторое время, зависящее от типа РК и для гальванически развязанных выходных РК составляющее порядка 400 мкс.

#### 3.2.4.2.Чтение состояния группы РК

DeviceIoControl (

hDD64PCI ,

DD64PCI\_GET\_DS\_GROUP,

&bufInput,

1,

&bufOutput,

16,

&nOutput ,

NULL );

Описание

Функция дает возможность одновременно определить состояние 16-ти разовых команд, входящих в одну группу.

В таблице указано соответствие номера РК (1..64), номера группы (1..4) и индекса в выходном буфере (0..15).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № группы | 1 | 2 | 3 | 4 |
| Индекс массива |  | | | |
| 0 | 1 | 17 | 33 | 49 |
| 1 | 2 | 18 | 34 | 50 |
| 2 | 3 | 19 | 35 | 51 |
| 3 | 4 | 20 | 36 | 52 |
| 4 | 5 | 21 | 37 | 53 |
| 5 | 6 | 22 | 38 | 54 |
| 6 | 7 | 23 | 39 | 55 |
| 7 | 8 | 24 | 40 | 56 |
| 8 | 9 | 25 | 41 | 57 |
| 9 | 10 | 26 | 42 | 58 |
| 10 | 11 | 27 | 43 | 59 |
| 11 | 12 | 28 | 44 | 60 |
| 12 | 13 | 29 | 45 | 61 |
| 13 | 14 | 30 | 46 | 62 |
| 14 | 15 | 31 | 47 | 63 |
| 15 | 16 | 32 | 48 | 64 |

Входные параметры

***UCHAR bufInput;*** // - номер группы (1..4)

Выходные параметры

***UCHAR bufOutput[16];***  //- cостояние РК (0/1)

См. примечание к п. 3.2.4.1.

### 3.2.5.Фильтры выходных РК .

#### 3.2.5.1.Создание фильтров.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_CREATE\_FILTER1,

&bufInput,

8,

NULL ,

0 ,

&nOutput ,

NULL );

Описание

Функция выполняет настройку фильтра №1.

Входные параметры

***UCHAR bufInput [8];***  // - в таблице приведено соответствие номеров РК индексу массива и номеру бита.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № бита | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Индекс массива |  | | | | | | | |
| 0 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 1 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 |
| 2 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 |
| 3 | 32 | 31 | 30 | 29 | 28 | 27 | 26 | 25 |
| 4 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 |
| 5 | 48 | 47 | 46 | 45 | 44 | 43 | 42 | 41 |
| 6 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 |
| 7 | 64 | 63 | 62 | 61 | 60 | 59 | 58 | 57 |

Для включения РК в фильтр №1 следует установить соответствующий бит в 1.

Аналогичные функции для создания фильтров №2 и №3 имеют соответственно коды ***DD64PCI\_CREATE\_FILTER2*** и ***DD64PCI\_CREATE\_FILTER3***.

#### 3.2.5.2.Состояние фильтров.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_CONFIG\_FILTER1,

NULL,

0,

& bufOutput, .

8 ,

&nOutput ,

NULL );

Описание

Функция возвращает информацию о текущей конфигурации “One Hot”-фильтра №1.

Выходные параметры

***UCHAR bufOutput[8];*** // - каждый бит выходного массива соответствует РК как указано в таблице п.3.2.5.1. Если бит = 1, РК включена в фильтр.

Аналогичные функции для чтения состояния фильтров №2 и №3 имеют соответственно коды ***DD64PCI\_READ\_CONFIG\_FILTER2*** и ***DD64PCI\_READ\_CONFIG\_FILTER3***.

#### 3.2.5.3.Конфигурирование портов OHF\_sm подключения внешних OH-фильтров.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_CREATE\_EXTERNAL\_FILTER1,

&bufInput,

1,

NULL ,

0 ,

&nOutput ,

NULL );

Описание

Функция выполняет настройку порта подключения OH\_sm1 внешнего фильтра №1.

Входные параметры

***UCHAR bufInput;***  // = 0 – линия отключена;

***//*** = 1 режим MASTER.

***//*** = 2 режим SLAVE.

Если код задан неверно, никаких действий не производится.

Аналогичные функции для фильтров №2 и №3 имеют соответственно коды ***DD64PCI\_CREATE\_EXTERNAL\_FILTER2*** и ***DD64PCI\_CREATE\_EXTERNAL\_FILTER3.***

#### 3.2.5.4.Состояние портов внешних фильтров.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_CONFIG\_EXTERNAL\_FILTER1,

NULL,

0,

& bufOutput, .

1 ,

&nOutput ,

NULL );

Описание

Функция возвращает информацию о состоянии порта OH\_sm1 внешнего фильтра №1.

Выходные параметры

***UCHAR bufOutput;*** // ***;***  // = 0 – линия отключена;

***//*** = 1 режим MASTER.

***//*** = 2 режим SLAVE.

Аналогичные функции для фильтров №2 и №3 имеют соответственно коды ***DD64PCI\_READ\_CONFIG\_EXTERNAL\_FILTER2 и DD64PCI\_READ\_CONFIG\_EXTERNAL\_FILTER3.***

## 3.3.Таймер.

### 3.3.1.Чтение/запись в регистр таймера.

См. п.п. 3.1.2 и 3.1.3.

### 3.3.2.Запись в регистр делителя частоты.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_RDIVT,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код ***bufInput*** , определяющий частоту тактирования таймера.

Частота тактирования определяется по формуле:

***F = 1 000 000 / (bufInput +1) Гц***.

### 3.3.3.Чтение регистра делителя частоты.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_RDIVT,

NULL ,

0 ,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра делителя частоты.

Частота тактирования равна:

***F = 1 000 000 / (bufOutput+1) Гц.***

### 3.3.4.Запись в регистр сравнения таймера.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_TMRCMP,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр сравнения таймера.

## 3.4.Работа с ЦАП.

### 3.4.1.Запись в регистр данных ЦАП.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_DACDATA,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр данных.

### 3.4.2.Запись в регистр адреса ЦАП.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_DACADR,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр адреса.

### 3.4.3.Запись в регистр управления ЦАП.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_DACCTRL,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр управления.

### 3.4.4.Чтение регистра статуса ЦАП.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_DACST,

NULL ,

0 ,

&bufOutput,

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра статуса.

### 3.4.5.Чтение регистра конфигурации ЦАП.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_***DACCFG***,

NULL ,

0 ,

&bufOutput,

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра конфигурации.

## 3.5.Работа с АЦП.

### 3.5.1.Запись/чтение регистров АЦП.

#### 3.5.1.1.Запись в регистр данных АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_ADCDATA1,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр данных.

#### 3.5.1.2.Чтение регистра данных АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_ADCDATA1,

NULL ,

0 ,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра данных.

#### 3.5.1.3.Запись в регистр управления АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_ADCCTRL1,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр управления.

#### 3.5.1.4.Чтение регистра состояния АЦП1.

DeviceIoControl (

hDD64PCI ,

***DD64PCI\_READ\_ADCSTATE1***,

NULL ,

0 ,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра состояния.

#### 3.5.1.5.Запись в регистр данных АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_ADCDATA2,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр данных.

#### 3.5.1.6.Чтение регистра данных АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_ADCDATA2,

NULL ,

0 ,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра данных.

#### 3.5.1.7.Запись в регистр управления АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_WRITE\_ADCCTRL2,

&bufInput,

2,

NULL,

0,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput;*** //

Описание

Функция записывает код  ***bufInput***  в регистр управления.

#### 3.5.1.8.Чтение регистра состояния АЦП2.

DeviceIoControl (

hDD64PCI ,

***DD64PCI\_READ\_ADCSTATE2***,

NULL ,

0 ,

&bufOutput, .

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра состояния.

### 3.5.2.Комплексные функции АЦП1.

Функции данного раздела позволяют выполнять завершенные операции с АЦП1.

***Важные замечания.***

1.Для настройки диапазонов каналов АЦП1 пользователь должен использовать либо функции 3.5.2.1.-3.5.2.3. , либо функции обращения к регистрам из раздела 3.5.1. Смешивание этих функций может привести к непредсказуемым результатам.

2.Функции 3.5.2.1.-3.5.2.5 возвращают состояние АЦП1 после выполнения заданной операции ***USHORT adcState***. Если АЦП1 занято, для ожидания его освобождения в дальнейшем используется функция опроса регистра состояния (п.3.5.1.4.).См.рис 1.

Функции 3.5.2.1.-3.5.2.5

Опрос состояния АЦП1 (п.3.5.1.4.)

АЦП1 свободен?

Рис. 1

#### 3.5.2.1.Задание диапазонов входных каналов 0..3 АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC1\_RANGE\_CHANNELS\_0\_3,

&bufInput,

8,

&adcState,

2,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput[4];*** //

Описание

Функция задает диапазон преобразования для каждого из каналов 0..3 и опрашивает регистр статуса ADC1STATE.

Каждый элемент массива ***bufInput[4]*** задает диапазон канала, номер которого равен индексу элемента массива;

допустимые значения ***bufInput[]*** :

= 0: - 10V..+ 10V

= 1: - 5V..+ 5V

= 2: -2.5V..+2.5V

= 3: 0V.. 10V

#### 3.5.2.2.Задание диапазонов входных каналов 4..7 АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC1\_RANGE\_CHANNELS\_4\_7,

&bufInput,

8,

&adcState,

2,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput[4];*** //

Описание

Функция задает диапазон преобразования для каждого из каналов 4..7 и опрашивает регистр статуса ADC1STATE.

Каждый элемент массива ***bufInput[4]*** задает диапазон канала, номер которого равен (индекс элемента массива+4);

допустимые значения ***bufInput[]*** :

= 0: - 10V..+ 10V

= 1: - 5V..+ 5V

= 2: -2.5V..+2.5V

= 3: 0V.. 10V

#### 3.5.2.3.Задание диапазона одного входного канала АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC1\_RANGE\_ONE\_CHANNEL,

&bufInput,

4,

&adcState,

2,

&nOutput ,

NULL );

Описание

Функция задает диапазон преобразования для одного канала АЦП1 и опрашивает регистр статуса ADC1STATE.

Входные параметры

***USHORT bufInput[0]***- номер канала (0..7)

***USHORT bufInput[1]***- диапазон канала;

допустимые значения ***bufInput[1]*** :

= 0: - 10V..+ 10V

= 1: - 5V..+ 5V

= 2: -2.5V..+2.5V

= 3: 0V.. 10V

#### 3.5.2.4.Установка номера и режима работы канала АЦП1 для преобразования.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC1\_CHANNELS\_NUMBER\_AND\_MODE,

&bufInput,

6,

&adcState,

2,

&nOutput ,

NULL );

Описание

Функция служит для выбора канала и его режима преобразования. Если одиночное преобразование осуществляется для одного канала несколько раз подряд, достаточно одного вызова этой функции перед началом первого преобразования.

Перед завершением выполнении функция опрашивает регистр статуса ADC1STATE

Входные параметры

***USHORT bufInput[0]***- номер канала (0..7)

***USHORT bufInput[1]***- режим :

= 0: 8 одиночных аналоговых входов

= 1: 4 псевдо-дифференциальных входа

= 2: 4 дифференциальных входа

= 3: 7 псевдо-дифференциальных входов

***USHORT bufInput[2]***– тип выходного кодирования

= 0: двоичное дополнение

= 1: прямая двоичная логика

#### 3.5.2.5.Запуск одиночного преобразования АЦП1.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC1\_CHANNEL\_PUSK,

NULL ,

0,

&adcState,

2,

&nOutput ,

NULL );

Описание

Функция выполняет запуск одиночного преобразования и опрашивает регистр статуса ADC1STATE. После запуска одиночного преобразования следует дождаться перехода АЦП1 в свободное состояние и прочитать результаты преобразования в регистре DATA (п.3.5.1.2.)

.

запуск одиночного преобразования(п. 3.5.2.5.)

АЦП свободен?

Опрос состояния АЦП (п. 3.5.1.4.)

чтение результатов одиночного преобразования(п.п. 3.5.1.2.)

Рис.2

### 3.5.3.Комплексные функции АЦП2.

Функции данного раздела позволяют выполнять завершенные операции с АЦП2.

***Важные замечания.***

1.Для настройки диапазонов каналов АЦП2 пользователь должен использовать либо функции 3.5.3.1.-3.5.3.3. , либо функции обращения к регистрам из раздела 3.5.1. Смешивание этих функций может привести к непредсказуемым результатам.

2.Функции 3.5.3.1.-3.5.3.5 возвращают состояние АЦП2 после выполнения заданной операции ***USHORT adcState***. Если АЦП2 занято, для ожидания его освобождения в дальнейшем используется функция опроса регистра состояния (п.3.5.1.8.).См.рис 3.

Функции 3.5.3.1.-3.5.3.5

Опрос состояния АЦП2 (п.3.5.1.8.)

АЦП2 свободен?

Рис. 3

#### 3.5.3.1.Задание диапазонов входных каналов 0..3 АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC2\_RANGE\_CHANNELS\_0\_3,

&bufInput,

8,

&adcState,

2,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput[4];*** //

Описание

Функция задает диапазон преобразования для каждого из каналов 0..3 и опрашивает регистр статуса ADC2STATE.

Каждый элемент массива ***bufInput[4]*** задает диапазон канала, номер которого равен индексу элемента массива;

допустимые значения ***bufInput[]*** :

= 0: - 10V..+ 10V

= 1: - 5V..+ 5V

= 2: -2.5V..+2.5V

= 3: 0V.. 10V

#### 3.5.3.2.Задание диапазонов входных каналов 4..7 АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC2\_RANGE\_CHANNELS\_4\_7,

&bufInput,

8,

&adcState,

2,

&nOutput ,

NULL );

Входные параметры

***USHORT bufInput[4];*** //

Описание

Функция задает диапазон преобразования для каждого из каналов 4..7 и опрашивает регистр статуса ADC2STATE.

Каждый элемент массива ***bufInput[4]*** задает диапазон канала, номер которого равен (индекс элемента массива+4);

допустимые значения ***bufInput[]*** :

= 0: - 10V..+ 10V

= 1: - 5V..+ 5V

= 2: -2.5V..+2.5V

= 3: 0V.. 10V

#### 3.5.3.3.Задание диапазона одного входного канала АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC2\_RANGE\_ONE\_CHANNEL,

&bufInput,

4,

&adcState,

2,

&nOutput ,

NULL );

Описание

Функция задает диапазон преобразования для одного канала АЦП2 и опрашивает регистр статуса ADC2STATE.

Входные параметры

***USHORT bufInput[0]***- номер канала (0..7)

***USHORT bufInput[1]***- диапазон канала;

допустимые значения ***bufInput[1]*** :

= 0: - 10V..+ 10V

= 1: - 5V..+ 5V

= 2: -2.5V..+2.5V

= 3: 0V.. 10V

#### 3.5.3.4.Установка номера и режима работы канала для преобразования АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC2\_CHANNELS\_NUMBER\_AND\_MODE,

&bufInput,

6,

&adcState,

2,

&nOutput ,

NULL );

Описание

Функция служит для выбора канала и его режима преобразования. Если одиночное преобразование осуществляется для одного канала несколько раз подряд, достаточно одного вызова этой функции перед началом первого преобразования.

Перед завершением выполнении функция опрашивает регистр статуса ADC2STATE

Входные параметры

***USHORT bufInput[0]***- номер канала (0..7)

***USHORT bufInput[1]***- режим :

= 0: 8 одиночных аналоговых входов

= 1: 4 псевдо-дифференциальных входа

= 2: 4 дифференциальных входа

= 3: 7 псевдо-дифференциальных входов

***USHORT bufInput[2]***– тип выходного кодирования

= 0: двоичное дополнение

= 1: прямая двоичная логика

#### 3.5.3.5.Запуск одиночного преобразования АЦП2.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_ADC2\_CHANNEL\_PUSK,

NULL ,

0,

&adcState,

2,

&nOutput ,

NULL );

Описание

Функция выполняет запуск одиночного преобразования и опрашивает регистр статуса ADC2STATE. После запуска одиночного преобразования следует дождаться перехода АЦП2 в свободное состояние и прочитать результаты преобразования в регистре DATA (п.3.5.1.6.)

.

запуск одиночного преобразования(п. 3.5.3.5.)

АЦП свободен?

Опрос состояния АЦП (п. 3.5.1.8.)

чтение результатов одиночного преобразования(п.п. 3.5.1.6.)

Рис.4

### 3.5.4.Чтение регистра конфигурации АЦП.

DeviceIoControl (

hDD64PCI ,

DD64PCI\_READ\_***ADCCFG***,

NULL ,

0 ,

&bufOutput,

2 ,

&nOutput ,

NULL );

Описание

Функция возвращает в переменной ***USHORT bufOutput*** содержимое регистра конфигурации.

## 3.6. Прочие функции

### 3.6.1. Чтение параметров модуля.

Функция возвращает начальные значения адресов диапазонов ввода-вывода и IRQ.

DeviceIoControl (

hARINC ,

DD64PCI\_GET\_PARAM,

NULL,

0,

&bufOutput,

6,

&nOutput ,

NULL);

Выходные параметры:

struct {

USHORT adrPLX; //нач. адрес диапазона ввода-вывода PLX

USHORT adr; //нач. адрес диапазона ввода-вывода модуля

USHORT nIRQ; //номер IRQ

} bufOutput;

### 3.6.2. Чтение серийного номера адаптера.

DeviceIoControl (

hARINC ,

DD64PCI\_GET\_SN,

NULL,

0,

&bufOutput,

2,

&nOutput ,

NULL);

Выходные параметры:

USHORT bufOutput; // переменная, в которой драйвер возвращает серийный номер модуля

## 3.7.Обслуживание прерываний.

Обслуживание прерываний модуля производится в 2 этапа. При возникновении прерывания операционная система (ОС) передает управление обработчику прерывания драйвера, который выполняет минимальные действия, определяемые требованиями ОС, и запускает пользовательский обработчик. Последний должен определить источник прерывания модуля и выполнить все дальнейшие действия по обработке прерывания.

Для обслуживания прерываний в приложении создаются:

- дополнительный поток, являющийся пользовательским обработчиком прерываний;

- синхронизирующее событие, которое в исходном состоянии находится в занятом состоянии, освобождается обработчиком прерывания драйвера после возникновения аппаратного прерывания, активизирует поток (пользовательский обработчик прерывания) и вновь переходит в занятое состояние до прихода следующего прерывания.

Далее перечислены дополнения, которые рекомендуется внести в программу для обеспечения возможности работы с прерыванием и краткие пояснения к ним.

Имена переменных и функций произвольны.

Область данных.

***HANDLE hEvent, hThread*** дескрипторы события и потока

***DWORD dwThreadID*** переменная, в которую система внесет идентификатор потока

***DWORD WINAPI f\_INT(LPVOID)*** прототип асинхронной функции, которая будет активизироваться из обработчика прерывания, включенного в драйвер

Область кодов.

***hEvent=CreateEvent(NULL, TRUE,FALSE, "SignalEvent");***  создаем событие, *SignalEvent* – имя события (должно быть уникальным в системе)

***hThread = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE) f\_INT,***

***NULL, 0, &dwThreadID);*** создаем поток, где 3-й параметр вызова - адрес функции, которая начнет выполняться, после того как поток будет активизирован.

***DeviceIoControl (hDD64PCI, DD64PCI\_START\_INT, &hEvent, 4 ,***  ***NULL****,*  ***0, &*** ***nOutput, NULL );*** передаем в драйвер дескриптор события для синхронизации прерывания

***DeviceIoControl (hDD64PCI, DD64PCI\_STOP\_INT, NULL, 0 ,***  ***NULL****,*  ***0, &*** ***nOutput, NULL );*** Функция завершения работы с прерываниями.

Область функции-обработчика прерывания.

***WaitForSingleObject(hEvent, INFINITE);*** для того, чтобы функция активизировалась только в случае прихода в драйвер аппаратного прерывания, она должна содержать вызов функции ожидания

***ResetEvent(hEvent);*** сброс события

*Способ обработки прерывания зависит от его источника (см.ТО).*