АНАЛИЗАТОР СПЕКТРА

Руководство программиста

ЗТМС.00068-01 33

Описание функций доступа к драйверам Zet через библиотеку Zadc.dll

Работа с устройствами фирмы Zet.

Пользователь может работать с устройствами фирмы Zet с помощью вызовов функций библиотеки Zadc.dll. Данное руководство приведено для библиотеки Zadc.dll с версией 5.0 и старше.

Библиотека написана на C++ и имеет интерфейс WINAPI (Microsoft® Windows® application programming interface (API)). Все объявления в настоящем руководстве приведены на языке C/C++. Все функции библиотеки Zadc.dll возвращают код ошибки. Значение 0 – говорит о выполнении функции без ошибок.

Функции Zadc.dll имеют вид ZXXX(…). Если функция влияет на работу только АЦП, то она заканчивается на ADC, если функция влияет на работу только ЦАП – она заканчивается на DAC. Общие функции не имеют специфичных обозначений.

Функции делятся на две категории: информационные и управляющие, информационные функции, возвращают в программу пользователя различные параметры и не меняют режим работы драйвера, сигнального процессора и модулей АЦП и ЦАП. Управляющие программы меняют режим работы драйвера и устройств, подключенных к драйверу. При этом в драйвере устанавливается признак изменения режима работы (*modify*). Если несколько программ одновременно работают с одним драйвером, то каждая программа должна при каждом обращении к драйверу считывать этот признак и должна менять свой режим работы: обработки сигналов, отображения и пр.

Структурная схема управления устройством на примере KADSP/PCI представлена на рисунке 1.

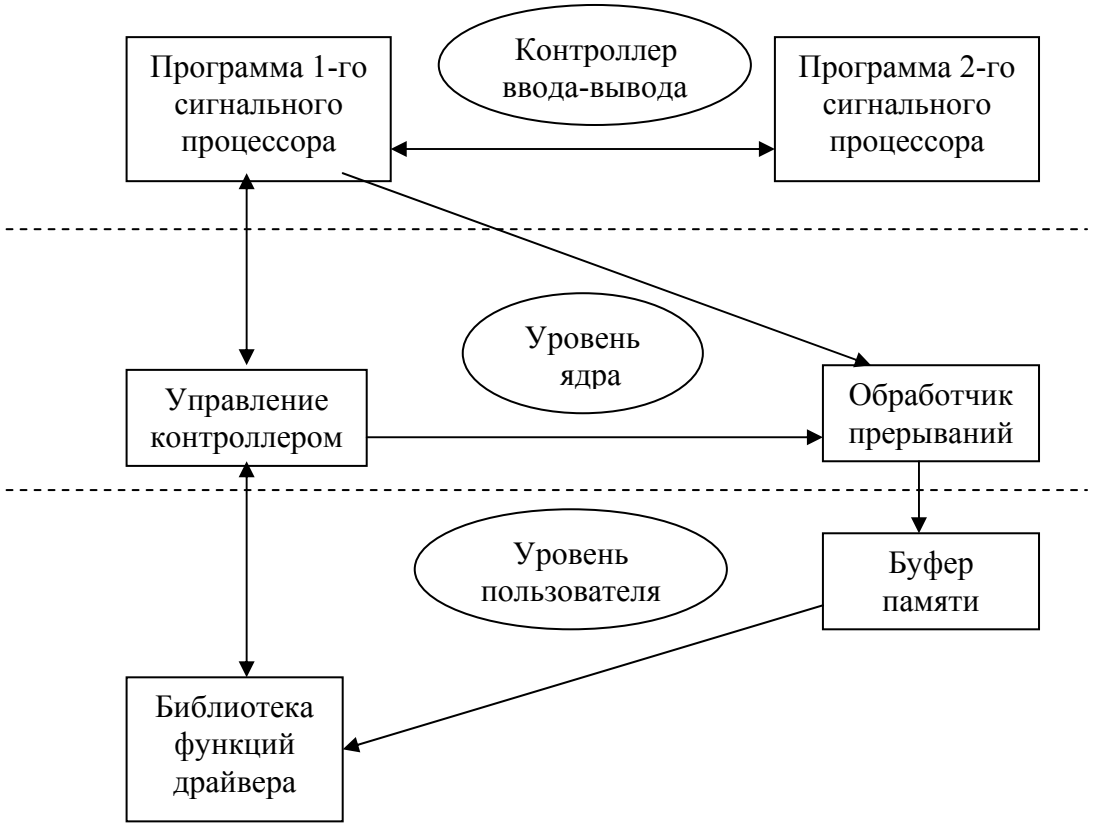

Рисунок 1.

Сигнальный процессор выполняет следующие функции:

- устанавливает коэффициенты усиления на модуле АЦП и коэффициент затухания на модуле ЦАП;
- устанавливает частоту дискретизации на модулях АЦП и ЦАП;
- обеспечивает синхронизацию нескольких сигнальных процессоров;
- по завершению преобразования производит опрос выбранных каналов АЦП и формирует массив данных во внутренней памяти сигнального процессора;
- проводит первичную цифровую фильтрацию и прореживание оцифрованных входных сигналов;
- проводит каскадную декадную фильтрацию и прореживание
- устанавливает и снимает сигнал прерывания для центрального процессора при заполнении внутреннего буфера памяти;
- обеспечивает перекачку данных из внутренней памяти в память центрального процессора порции данных, определяемой драйвером как размер буфера для перекачки.

Драйвер на уровне ядра обеспечивает обмен управляющими данными с сигнальным процессором. Программа обработчика прерываний от сигнального процессора обеспечивает прием оцифрованных сигналов и размещение их в буфере памяти. Для каждого сигнального процессора создается свой буфер данных.

На уровне пользователя функционирует библиотека подпрограмм для связи пользовательских программ с частью драйвера, расположенного на уровне ядра. Драйвер поддерживается операционными системами Windows 2000 и Windows XP, Windows 2003.

Индексы контроллеров PCI, как правило, распределяются последовательно в направлении от разъема AGP к краю системной платы. Индексы контроллеров USB зависят от порядка подключения и от номера порта USB.

В системе может быть установлено несколько контроллеров KADSP/PCI или KADSP/PDP, к каждому сигнальному процессору модуля может быть подключен только один модуль АЦП или ЦАП. Дополнительно, к любому из сигнальных процессоров, управляющих модулем АЦП, можно подключить усилитель заряда ПУ 8/10 и управлять его программируемым коэффициентом усиления. Два сигнальных процессора на плате KADSP/PCI соединены по схеме ведущий/ведомый. Оба процессора работают на одной тактовой частоте от одного тактового генератора. Ведущим процессором на плате является процессор, расположенный справа и ближе к краю платы (см. рисунок в РЭ). Соответственно справа расположен и 50-контактный разъем для подключения модулей АЦП и ЦАП. Ведомым процессором на плате является процессор, расположенный слева и ближе к разъемам «лемо». Соответственно слева расположен и 50-контактный разъем для подключения модулей АЦП и ЦАП.

Для удобства программирования реализована сквозная адресация сигнальных процессоров и подключенных к ним модулей. Ключевым параметром во всех функциях является номер сигнального процессора (numberDSP).

Подключение к драйверу и отключение.

Все программы, которые используют функции Zadc.dll, должны начинаться и заканчиваться процедурами из этого раздела. При этом никаких действий связанных с модулями АЦП, ЦАП и сигнальными процессорами не происходит. Это позволяет подключаться одновременно нескольким программам к одним модулям и сигнальным процессорам. Все текущие настройки сигнального процессора и модулей АЦП и ЦАП сохраняются во внутренних структурах драйвера. Для того чтобы определить, сколько установлено сигнальных процессоров в системе, необходимо выполнить 10 попыток подключения к драйверу с номерами сигнальных процессоров от 0 до 9 для соответствующего типа устройства. Количество удачных попыток подключения будет информировать о количестве установленных сигнальных процессоров в системе.

Подключение к драйверу
long ZOpen(long typeDevice, long numberDSP) *(Zadc.dll)*
функция для подключения к драйверу. При работе с платами АЦП и ЦАП – это первая функция, к которой необходимо произвести обращение.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800, 0x0900, 0x0A00, 0x0B00 – ошибка открытия драйвера,
0x0C00 – неподдерживаемая версия драйвера.
Функция не меняет признак *modify* в драйвере.

Отключение от драйвера
long ZClose(long typeDevice, long numberDSP) *(Zadc.dll)*
функция для отключения от сигнального процессора. Эта функция – последнее обращение к драйверу перед выходом из программы.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка закрытия драйвера,
Функция не меняет признак *modify* в драйвере.

Сброс и инициализация.

Сброс и останов сигнального процессора
long ZResetDSP(long typeDevice, long numberDSP) *(Zadc.dll)*
Функция сбрасывает все сигнальные процессоры одного устройства. После вызова данной функции для нормальной работы с сигнальными процессорами нужно их проинициализировать с помощью ZInitDSP(). Например, плата KADSP/PCI содержит два процессора, поэтому для ее инициализации нужно вызвать один раз ZResetDSP() (для любого из двух процессоров), а затем вызвать два раза ZInitDSP() для каждого процессора.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
Функция меняет признак *modify* в драйвере.

Инициализация сигнальный процессора
*long ZInitDSP(long typeDevice, long numberDSP, char *fileName)* *(Zadc.dll)*
Функция инициализирует и загружает в сигнальный процессор исполняемую программу (если устройство не содержит прошитую программу). При выключении питания программа в памяти сигнального процессора стирается. Для функционирования программ достаточно один раз загрузить программу после включения питания.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
fileName – строка с именем файла для загрузки. Если строка пустая, то загружается исполняемая программа по умолчанию (файл *.*ios*, поставляемый с драйвером). Если строка не пустая, то загружается файл, указанный в строке *fileName* из системной папки Windows, где находятся драйвера.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0200 – *filename* задан неправильно,
0x0800 – ошибка открытия драйвера,
0x0900…0x09FF – ошибка чтения записи контроллера.
Функция меняет признак *modify* в драйвере.

Пример программы инициализации сигнального процессора.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "zadc_int.h"        // Интерфейс библиотеки Zadc.dll

int main(void)
{
        long err;
        long typeDevice = KDU1616;      // Здесь используется ADC 1616 Sigma USB
        long numberDSP = 0;             // Первое найденное устройство
```

```
        err = ZOpen(typeDevice, numberDSP);
        if (err == 0)
                printf("Devices found, handle open.\n\r");
        else
        {
                printf("ERROR opening device: (%0x) \n\r", err);
                return 0;
        }
        err = ZResetDSP(typeDevice, numberDSP);
        if (err == 0)
                printf("DSP was reset\n\r");
        else
        {
                printf("ERROR device: (%0x) returned from ZResetDSP\n\r", err);
                err = ZClose(typeDevice, numberDSP);
                return 0;
        }
        err = ZInitDSP(typeDevice, numberDSP , "");
        if (err == 0)
                printf("DSP programs was initialized\n");
        else
                printf("ERROR device: (%0x) returned from ZInitDSP\n", err);

        err = ZClose(typeDevice, numberDSP);
        printf("Press any key for exit...\n");
        getch();

        return 0;
}
```

Сервисные функции

Опрос версии программ и драйвера

long ZGetVersion(long typeDevice, long numberDSP, char verDSP, char* verDRV, char* verLIB)*
(Zadc.dll)

опрос версии прошивки сигнального процессора, драйвера и библиотеки доступа к драйверу.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
verDSP – строковый массив, резервируемый в памяти пользовательской программы длиной не менее 100 байт. После обращения к подпрограмме возвращает строку с указанием версии прошивки сигнального процессора. Если прошивка не загружена в сигнальный процессор, то возвращается строка нулевой длины.
verDRV– строковый массив, резервируемый в памяти пользовательской программы длиной не менее 100 байт. После обращения к подпрограмме возвращает строку с указанием версии драйвера.
verLIB– строковый массив, резервируемый в памяти пользовательской программы длиной не менее 100 байт. После обращения к подпрограмме возвращает строку с указанием версии динамической библиотеки.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,

Функция не меняет признак *modify* в драйвере.

В данном примере считывается версия прошивки DSP, драйвера и библиотеки.

```
long PrintVersion(void)
{
	char dsp[100],drv[100],lib[100];
	long type = KD1610;		// Здесь тип устройства ADC 16/200
	long number = 0;		// Первый DSP

	err = ZOpen(type, numberDSP);
	if (err == 0)
		printf("Devices found, handle open.\n\r");
	else
	{
		printf("ERROR opening device: (%0x) \n\r", err);
		return err;
	}

	err = ZGetVersion(type, number, dsp, drv, lib);
	printf(“%s\n”, dsp);
	printf(“%s\n”, drv);
	printf(“%s\n”, lib);

	err = ZClose(type, number);
	printf("Press any key for exit...\n");
	getch();
	return 0;
}
```

В данном примере на Visual Basic считывается версия прошивки DSP.

```
Public Sub GetVerDSP()
   Dim typeDevice As Long  ' Код ошибки
   Dim numberDSP As Long   ' Длина строки
   Dim Err As Long         ' Код ошибки
   Dim Length As Long      ' Длина строки
   Dim strVer As String
   Dim strVerDrv As String, strVerDSP As String, strVerLib As String

   typeDevice = KDU1616 ' Здесь выбрано устройство ADC 1616 Sigma USB
   numberDSP = 0            ' Первое устройство

   Err = ZOpen(typeDevice, numberDSP)
   If Err <> 0 Then MsgBox ("Device not find, Error = " + CStr(Err))

   strVerDrv = Space$(100)
   strVerDSP = Space$(100)
   strVerLib = Space$(100)
   Err = ZGetVersion(typeDevice, numberDSP, strVerDrv, strVerDSP, strVerLib)
   strVer = Trim(strVerDSP)  ' Отрезаем пробелы
   Length = Len(strVer)
   If strLength > 0 Then strVer = Left(strVer, strLength - 1) ' Отрезаем нулевой символ
   MsgBox (strVer)
   Err = ZClose(typeDevice, numberDSP)
End Sub
```

Чтение кода ошибки

*long ZGetError(long typeDevice, long numberDSP, long *error)* *(Zadc.dll)*

Функция возвращает код последней внутренней ошибки драйвера (обработчика прерываний)

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

error – код ошибки.

Возвращаемое значение:

0 – нормальное выполнение функции,

0x0100 – *numberDSP* задан неправильно,

0x0800 – ошибка открытия драйвера,

Функция не меняет признак *modify* в драйвере.

Опрос изменения режима работы сигнального процессора

*long ZGetModify (long typeDevice, long numberDSP, long *modify)* *(Zadc.dll)*

Функция возвращает параметр изменения режима работы сигнального процессора. Драйвер позволяет одновременно работать нескольким программам. В этом случае каждая программа должна следить за параметром *modify* и в случае его изменения опрашивать необходимые параметры и соответственно менять свои параметры работы.

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

modify – признак изменения работы драйвера. При каждом существенном изменении параметров работы драйвера происходит инкремент признака *modify*. Программа пользователя должна считывать признак и запоминать текущее значение признака и при изменении этого значения производить необходимые действия.

Возвращаемое значение:

0 – нормальное выполнение функции,

0x0100 – *numberDSP* задан неправильно,

0x0800 – ошибка открытия драйвера,

Функция не меняет признак *modify* в драйвере.

Установка режима работы сигнального процессора.

В устройствах ADC 16/200, ADC 16/500, ADC 16/500P (конфигурация KADSP/PCI - АЦП16/200 - ЦАП16/2000; KADSP/PCI - АЦП16/1000 - ЦАП16/2000; KADSP/PDP - АЦП16/1000 - ЦАП16/2000) отсутствует режим автоопределения подключенных устройств. Т.е изначально в системе неизвестно к какому сигнальному процессору подключен модуль АЦП, а к какому модуль ЦАП. Необходимо, в соответствии со схемой подключения устройств, программно установить режимы работы сигнальных процессоров. По умолчанию, после загрузки программы в сигнальный процессор, установлен режим работы с модулем АЦП.

Установка работы с модулем АЦП
long ZSetTypeADC(long typeDevice, long numberDSP) *(Zadc.dll)*
установка сигнального процессора в режиме работы с модулем АЦП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Установка работы с модулем ЦАП
long ZSetTypeDAC(long typeDevice, long numberDSP) *(Zadc.dll)*
установка сигнального процессора в режиме работы с модулем ЦАП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос основных характеристик модулей АЦП и ЦАП.

Все драйверы для модулей АЦП и ЦАП построены по единому принципу. Программа пользователя не должна знать параметры модулей и их количество. Все параметры должны определяться по функциям опроса характеристик.

Опрос возможности работы сигнального процессора с модулем АЦП.
*long ZGetEnableADC(long typeDevice, long numberDSP, long *enable)* *(Zadc.dll)*
опрос возможности работы сигнального процессора с модулем АЦП. В одних устройствах сигнальный процессор может поддерживать режим работы только с АЦП или только с ЦАП, в других - может работать одновременно с АЦП и ЦАП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – возможность работы с модулем АЦП. 0 – не поддерживается , 1 – поддерживается.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
Функция не меняет признак *modify* в драйвере.

Опрос возможности работы сигнального процессора с модулем ЦАП.
*long ZGetEnableDAC(long typeDevice, long numberDSP, long *enable)* *(Zadc.dll)*
опрос возможности работы сигнального процессора с модулем ЦАП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – возможность работы с модулем ЦАП. 0 – не поддерживается, 1 – поддерживается.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
Функция не меняет признак *modify* в драйвере.

Опрос максимального количества каналов модуля АЦП/ЦАП.
*long ZGetQuantityChannelADC (long typeDevice, long numberDSP, long *quantityChannel)*
*long ZGetQuantityChannelDAC (long typeDevice, long numberDSP, long *quantityChannel)*
(Zadc.dll)
опрос максимального количества каналов модуля АЦП/ЦАП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
quantityChannel - максимальное количество каналов модуля АЦП/ЦАП
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Опрос веса младшего разряда АЦП/ЦАП
*long ZGetDigitalResolutionADC (long typeDevice, long numberDSP, double *digitalResolution)*
*long ZGetDigitalResolutionDAC (long typeDevice, long numberDSP, double *digitalResolution)*

опрос веса младшего разряда АЦП/ЦАП. Вес определяется как напряжение, измеренное в Вольтах для одного младшего разряда АЦП/ЦАП. Эта функция необходима, для преобразования двоично-дополнительного кода поступающего с АЦП (в ЦАП) в напряжение в Вольтах установленное на входе АЦП (на выходе ЦАП). Оцифрованные двоичные данные хранятся в буфере памяти (рисунок 1). Для того чтобы преобразовать двоичные данные в напряжения на входе модуля АЦП, необходимо умножить двоичное число на вес младшего разряда АЦП и разделить на коэффициент усиления выбранного канала. Для того чтобы преобразовать двоичные данные в напряжения на выходе модуля ЦАП, необходимо умножить двоичное число на вес младшего разряда ЦАП и умножить на коэффициент затухания (аттенюатора).
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9),
digitalResolution – вес младшего разряда в Вольтах.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Опрос количества двоичных разрядов АЦП/ЦАП
*long ZGetBitsADC (long typeDevice, long numberDSP, long *numberBits)* *(Zadc.dll)*
*long ZGetBitsDAC (long typeDevice, long numberDSP, long *numberBits)*
опрос разрядности. Эта информация полезна для оценки максимального входного напряжения. Чтобы получить максимальный размах напряжения, необходимо 2 возвести в степень количества двоичных разрядов АЦП/ЦАП и умножить на вес младшего разряда АЦП/ЦАП. Максимальная амплитуда напряжение определяется как половина размаха. Например, для модуля АЦП 16/200 максимальная амплитуда входного сигнала равна 10.14 Вольт при единичном коэффициенте усиления.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberBits – количество двоичных разрядов АЦП/ЦАП.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Опрос размера каждого отсчета АЦП/ЦАП в 16-разрядных словах,
*long ZGetWordsADC (long typeDevice, long numberDSP, long *numberWords)* *(Zadc.dll)*
*long ZGetWordsDAC (long typeDevice, long numberDSP, long *numberWords)*
опрос размера каждого отсчета АЦП/ЦАП в 16-разрядных словах. Этот размер важен для того, чтобы задать тип данных при работе с оцифрованными значениями. Если количество 16-разрядных слов равно 1, то тип данных должен быть:

short	в C++,
integer	в Visual Basic.
smallint	в Delphi

Если количество 16-разрядных слов равно 2, то тип данных должен быть:

long	в C++,
long	в Visual Basic.

integer в Delphi
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberWords – количество 16-разрядных слов, занимаемое каждым отсчетом.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Установка частоты дискретизации и режима синхронизации АЦП/ЦАП

Частота дискретизации сигнала F_d зависит от частоты опорного генератора F_o и коэффициента деления частоты опорного генератора K_d. Коэффициент - целое положительное число.

$$F_d = \frac{F_o}{K_d}$$

Некоторые типы устройств позволяют подключение внешнего опорного генератора. Для удобства программирования существует два метода установки частоты дискретизации.

- В драйвере существует список возможных частот дискретизации. При помощи функции *ZGetListFreqADC() (ZGetListFreqDAC())* можно получить список возможных частот дискретизации. А при помощи функции *ZSetNextFreqADC() (ZSetNextFreqDAC())* можно поменять текущее значение частоты дискретизации в большую или меньшую сторону на один шаг.
- Задается требуемая частота дискретизации, драйвер через функцию *ZSetFreqADC() (ZSetFreqDAC())* устанавливает ближайшую возможную частоту дискретизации и возвращает точное значение установленной частоты дискретизации.

Получение списка возможных частот дискретизации АЦП/ЦАП
*long ZGetListFreqADC (long typeDevice, long numberDSP, long next, double *freq) (Zadc.dll)*
*long ZGetListFreqDAC (long typeDevice, long numberDSP, long next, double *freq)*
получение списка возможных частот дискретизации. При многократном обращении к этой функции, она возвращает возможные частоты дискретизации.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
next – параметр для получения списка. При первом обращении этот параметр должен быть равен 0, при последующих обращениях этот параметр не должен быть равен 0. После чтения последнего элемента функция возвращает код ошибки 0x0200 и значение частоты равное 0.
freq – следующее возвращаемое значение частоты дискретизации в Гц.
Возвращаемое значение (код ошибки):

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – next задан неправильно (конец списка)
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

В данном примере подпрограмма выдает на экран список возможных частот

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "zadc_int.h"        // Интерфейс библиотеки Zadc.dll

int main(void)
{
        long err;
```

```
        long typeDevice;
        long numberDSP;
        long next;
        double freq;

        typeDevice = KD1610;      // Тип устройства - ADC 16/200
        numberDSP = 0;            // Первый сигнальный процессор

        err = ZOpen(typeDevice, numberDSP); //открывается доступ к драйверу
        if(err != 0) return err;            //возврат из программы в случае неудачи
        next = 0;
        while(1)                            //бесконечный цикл
         {
           err=ZGetListFreqADC(typeDevice, numberDSP, next, &freq); //получить
                                            //значение частоты
           if(err != 0) break;              //если ошибка, то выход из цикла
           printf("%g Hz\n",freq);          //распечатать значение частоты
           next=1;                          //при следующем обращении, будет
                                            //считываться следующее значение частоты
        }
        err=ZClose(typeDevice, numberDSP);  //закрыть доступ к драйверу
        printf("Press any key for exit...\n");
        getch();
        return 0;
}
```

Устанавка большей или меньшей частоты дискретизации АЦП/ЦАП

*long ZSetNextFreqADC(long typeDevice, long numberDSP, long next, double *freq)* *(Zadc.dll)*
*long ZSetNextFreqDAC(long typeDevice, long numberDSP, long next, double *freq)*
функция устанавливает большую или меньшую частоту дискретизации по сравнению с текущей частотой дискретизации.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
next – параметр для изменения частоты дискретизации.

- -1 уменьшает частоту дискретизации
- +1 увеличивает частоту дискретизации
- 0 не меняет частоту дискретизации

freq – возвращаемое значение установленной частоты дискретизации в Гц.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция меняет признак *modify* в драйвере.

Опрос текущей частоты дискретизации АЦП/ЦАП

*long ZGetFreqADC(long typeDevice, long numberDSP, double *freq)* *(Zadc.dll)*
*long ZGetFreqDAC(long typeDevice, long numberDSP, double *freq)*
опрос текущего значения частоты дискретизации
typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).
freq – значение установленной частоты дискретизации в Гц,
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установка частоты дискретизации АЦП/ЦАП
*long ZSetFreqADC(long typeDevice, long numberDSP, double freqIn, double *freqOut) (Zadc.dll)*
*long ZSetFreqDAC(long typeDevice, long numberDSP, double freqIn, double *freqOut)*
функция устанавливает частоту дискретизации, наиболее близкую к требуемой, и возвращает значение установленной частоты дискретизации. Эти частоты могут различаться.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
freqIn – требуемая частота дискретизации в Гц,
freqOut – установленная частота дискретизации в Гц,
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос текущей опорной частоты АЦП/ЦАП
*long ZGetExtFreqADC(long typeDevice, long numberDSP, double *oporFreq) (Zadc.dll)*
*long ZGetExtFreqDAC(long typeDevice, long numberDSP, double *oporFreq)*
опрос значения текущей опорной частоты.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
oporFreq – значение опорной частоты в Гц.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установка значения внешней опорной частоты АЦП/ЦАП
long ZSetExtFreqADC(long typeDevice, long numberDSP, double extFreq) (Zadc.dll)
long ZSetExtFreqDAC(long typeDevice, long numberDSP, double extFreq)
установка значения внешней опорной частоты.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
extFreq – значение опорной частоты в Гц.
Возвращаемое значение:
0 – нормальное выполнение функции,

0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос режима работы от внешней опорной частоты
*long ZGetEnableExtFreq(long typeDevice, long numberDSP, long *enable)* *(Zadc.dll)*
опрос режима работы от внешней опорной частоты.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – 0 – режим внутренней опорной частоты, 1 – режим внешней опорной частоты для формирования частоты дискретизации.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Установка режима работы от внешней опорной частоты АЦП
long ZSetEnableExtFreq(long typeDevice, long numberDSP, long enable) *(Zadc.dll)*
установка/сброс режима работы от внешней опорной частоты. Влияет и на АЦП и на ЦАП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – 0 – режим внутренней опорной частоты, 1 – режим внешней опорной частоты для формирования частоты дискретизации.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос разрешения внешнего запуска накопления данных
*long ZGetEnableExtStart(long typeDevice, long numberDSP, long *enable)* *(Zadc.dll)*
Функция опрашивает разрешение/запрещение начала накопления данных от внешнего сигнала запуска.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – параметр, разрешающий/запрещающий использование внешнего сигнала запуска.
0 – запрещение внешнего запуска. Накопление данных АЦП начинается по команде *ZStartADC()*.
1 – разрешение внешнего запуска. Накопление данных начинается после подачи команды *ZStartADC()* и последующем появлении внешнего сигнала “запуск”.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установка внешнего запуска накопления данных

long ZSetEnableExtStart(long typeDevice, long numberDSP, long enable) *(Zadc.dll)*

Функция разрешает/запрещает начало накопления данных от внешнего сигнала запуска. Влияет и на АЦП и на ЦАП. Например, для устройств ADC 16/200, APC 216 на кронштейне контроллера ввода/вывода расположены четыре разъема Lemo. Один из этих разъемов является входом сигнала внешнего запуска, другой разъем является выходом сигнала “запуск” (этот сигнал переходит в другое состояние в начале процесса накопления). Эти сигналы позволяют подключать контроллеры последовательно друг с другом для реализации синхронной схемы накопления данных. Более подробно разъемы описаны в руководстве по эксплуатации контроллера KADSP/PCI. Функция не начинает процесс накопления данных – она устанавливает режим запуска накопления данных.

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

enable – параметр, разрешающий/запрещающий использование внешнего сигнала запуска.

0 – разрешение программного запуска. Накопление данных АЦП начинается по команде *ZStartADC()*

1 – разрешение внешнего сигнала запуска. Накопление данных начинается после подачи команды *ZStartADC()* и последующем появлении внешнего сигнала “запуск”.

Возвращаемое значение:

0 – нормальное выполнение функции,

0x0100 – *numberDSP* задан неправильно,

0x0800 – ошибка открытия драйвера,

0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Управление каналами ввода (вывода) АЦП/ЦАП

Опрос количества включенных каналов АЦП/ЦАП
*long ZGetNumberInputADC(long typeDevice, long numberDSP, long *workChannel) (Zadc.dll)*
*long ZGetNumberOutputDAC(long typeDevice, long numberDSP, long *workChannel)*
опрос количества включенных каналов для ввода данных.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
workChannel – количество включенных каналов.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Опрос разрешения канала для ввода (вывода) АЦП/ЦАП
*long ZGetInputADC(long typeDevice, long numberDSP, long numberChannel, long *enable)*
*long ZGetOutputDAC(long typeDevice, long numberDSP, long numberChannel, long *enable)*
(Zadc.dll)
Опрос разрешения канала для ввода
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.)
enable – параметр разрешения канала для ввода (вывода)
0 – заданный канал не выбран,
1 – заданный канал выбран,
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Включение канала для ввода (вывода) АЦП/ЦАП
long ZSetInputADC(long typeDevice, long numberDSP, long numberChannel, long enable)
long ZSetOutputDAC(long typeDevice, long numberDSP, long numberChannel, long enable)
(Zadc.dll)
Выбор или отмена канала для ввода.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.)
enable – параметр для включения-выключения канала для ввода (вывода)
0 – выключение заданного канала,
1 – включение заданного канала,
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),

0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос типа канала ввода АЦП
*long ZGetInputDiffADC(long typeDevice, long numberDSP, long numberChannel, long *enable)*
(Zadc.dll)
Опрос типа канала для ввода (синфазный или дифференциальный).
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.)
enable – параметр разрешения дифференциального режима канала для ввода
0 – синфазный тип заданного канала,
1 – дифференциальный тип заданного канала,
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установка типа канала ввода АЦП
long ZSetInputDiffADC(long typeDevice, long numberDSP, long numberChannel, long enable)
(Zadc.dll)
Опрос типа канала для ввода (синфазный или дифференциальный).
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.)
enable – параметр для установки дифференциального режима канала для ввода
0 – установка синфазного типа заданного канала,
1 – установка дифференциального типа заданного канала (объединяются два входа и используются как один дифференциальный),
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Управление коэффициентами усиления АЦП

На некоторых типах устройств на каждый из каналов аналогового ввода установлен усилитель с программируемым коэффициентом усиления. Каждый канал выбирается независимо от других каналов. Коэффициент усиления задается по каждому каналу индивидуально.

Существует два метода установки коэффициента усиления усилителя:

- В драйвере существует список возможных коэффициентов усиления. При помощи функции *ZGetListAmplifyADC()* можно получить список возможных коэффициентов усиления. А при помощи функции *ZSetNextAmplifyADC()* можно поменять текущее значение коэффициента усиления в большую или меньшую сторону на один шаг.
- Задается требуемый коэффициента усиления, драйвер через функцию *ZSetAmplifyADC()* устанавливает ближайший возможный коэффициент усиления и возвращает точное значение установленного коэффициента усиления.

Получение списка возможных коэффициентов усиления АЦП

*long ZGetListAmplifyADC(long typeDevice, long numberDSP, long next, double *amplify)*
(Zadc.dll)

получение списка возможных коэффициентов усиления. При многократном обращении к этой функции, она возвращает возможные коэффициенты усиления.

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

next – параметр для получения списка, при первом обращении этот параметр должен быть равен 0, при последующих обращениях этот параметр не должен быть равным 0. После чтения последнего элемента функция возвращает код ошибки 0x200 и значение коэффициента усиления равно 1.

amplify – следующее возвращаемое значение коэффициента усиления.

Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – next задан неправильно (конец списка),
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Установка большего или меньшего коэффициента усиления выбранного канала АЦП

*long ZSetNextAmplifyADC(long typeDevice, long numberDSP, long numberChannel, long next, double *amplify)* *(Zadc.dll)*

установка большего или меньшего коэффициента усиления выбранного канала.

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.)

next – параметр для изменения коэффициента усиления,

- -1 уменьшает коэффициент усиления,
- +1 увеличивает коэффициент усиления,
- 0 не меняет коэффициент усиления.

amplify – возвращаемое значение установленного коэффициента усиления.

Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0011 – таймаут установки коэф. усиления,
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос коэффициента усиления выбранного канала АЦП
*long ZGetAmplifyADC(long typeDevice, long numberDSP, long numberChannel, double *amplify)* *(Zadc.dll)*
опрос коэффициента усиления выбранного канала.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.).
amplify – возвращаемое значение установленного коэффициента усиления.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Установка коэффициента усиления выбранного канала АЦП
*long ZSetAmplifyADC(long typeDevice, long numberDSP, long numberChannel, double amplifyIn, double *amplifyOut)* *(Zadc.dll)*
установка коэффициента усиления выбранного канала.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.).
amplifyIn – задаваемое значение коэффициента усиления,
amplifyOut – возвращаемое значение установленного коэффициента усиления.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0011 – таймаут установки коэф. усиления,
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

В данном примере выбирается один канал и устанавливается коэффициент усиления равный четырем.

```
#include <windows.h>
#include <stdio.h>
#include <conio.h>
#include "zadc_int.h"        // Интерфейс библиотеки Zadc.dll
```

```
int main(void)
{
	long typeDevice;
	long numberDSP;
	long err, numberChannel;
	double amplify;

	typeDevice = KD1610;	// Тип устройства - ADC 16/200
	numberDSP = 0;		// Первый сигнальный процессор

	err = ZOpen(typeDevice, numberDSP);
	if (err == 0)		printf("Devices found, handle open.\n");
	else
	{
	printf("ERROR opening device: (%0x) returned from CreateFile\n", GetLastError());
	return 0;
	}
	err = ZSetInputADC(typeDevice, numberDSP,0,1);	// выбирается 1-ый канал
	err = ZSetInputADC(typeDevice, numberDSP,1,0);	// остальные каналы выкл.
	err = ZSetInputADC(typeDevice, numberDSP,2,0);
	err = ZSetInputADC(typeDevice, numberDSP,3,0);
	err = ZSetInputADC(typeDevice, numberDSP,4,0);
	err = ZSetInputADC(typeDevice, numberDSP,5,0);
	err = ZSetInputADC(typeDevice, numberDSP,6,0);
	err = ZSetInputADC(typeDevice, numberDSP,7,0);
	// опрос количества включенных каналов
	err = ZGetNumberInputADC(typeDevice, numberDSP,&numberChannel);
	// установка коэффициента усиления равным 4
	err = ZSetAmplifyADC(typeDevice, numberDSP, 0, 4., &amplify);
	err = ZClose(typeDevice, numberDSP);
	printf("Press any key for exit...\n");
	getch();
	return 0;
}
```

Управление коэффициентами усиления ПУ 8/10

Если подключен модуль усилителя заряда ПУ 8/10, то можно управлять коэффициентом усиления его каналов. Коэффициент усиления задается по каждому каналу индивидуально.

Существует два метода установки коэффициента усиления предварительного усилителя:

- В драйвере существует список возможных коэффициентов усиления. При помощи функции *ZGetListPreAmplifyADC()* можно получить список возможных коэффициентов усиления. А при помощи функции *ZSetNextPreAmplifyADC()* можно поменять текущее значение коэффициента усиления в большую или меньшую сторону на один шаг.
- Задается требуемый коэффициента усиления, драйвер через функцию *ZSetPreAmplifyADC()* устанавливает ближайший возможный коэффициент усиления и возвращает точное значение установленного коэффициента усиления.

Получение списка возможных коэффициентов усиления предварительного усилителя

*long ZGetListPreAmplifyADC(long typeDevice, long numberDSP, long next, double *amplify)*
(Zadc.dll)

получение списка возможных коэффициентов усиления предварительного усилителя. При многократном обращении к этой функции, она возвращает возможные коэффициенты усиления предварительного усилителя.

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

next – параметр для получения списка. При первом обращении этот параметр должен быть равен 0, при последующих обращениях этот параметр не должен быть равен 0. После чтения последнего элемента функция возвращает код ошибки 0x200 и значение коэффициента усиления равно 1.

amplify – следующее возвращаемое значение коэффициента усиления предварительного усилителя.

Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – next задан неправильно (конец списка),
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Установка большего или меньшего коэффициента усиления предварительного усилителя

*long ZSetNextPreAmplifyADC(long typeDevice, long numberDSP, long numberChannel, long next, double *amplify)* *(Zadc.dll)*

установка большего или меньшего коэффициента усиления предварительного усилителя выбранного канала.

typeDevice – тип устройства.

numberDSP – номер сигнального процессора (значение от 0 до 9).

numberChannel – выбор номера канала, в котором необходимо произвести изменение коэффициента усиления предварительного усилителя и должен принимать значения от 0 – для первого канала до 7 для восьмого канала,
next – параметр для изменения коэффициента усиления предварительного усилителя,

- -1 уменьшает коэффициент усиления,
- +1 увеличивает коэффициент усиления,
- 0 не меняет коэффициент усиления.

amplify – возвращаемое значение установленного коэффициента усиления предварительного усилителя.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция меняет признак *modify* в драйвере.

Установка коэффициента усиления предварительного усилителя выбранного канала.
long ZSetPreAmplifyADC(long typeDevice, long numberDSP, long numberChannel,
*double amplifyIn, double *amplifyOut)* *(Zadc.dll)*
установка коэффициента усиления предварительного усилителя выбранного канала.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – выбор номера канала, в котором необходимо произвести изменение коэффициента усиления предварительного усилителя и должен принимать значения от 0 – для первого канала до 7 для восьмого канала,
amplifyIn – задаваемое значение коэффициента усиления предварительного усилителя,
amplifyOut – возвращаемое значение установленного коэффициента усиления предварительного усилителя.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция меняет признак *modify* в драйвере.

Опрос коэффициента усиления предварительного усилителя выбранного канала.
long ZGetPreAmplifyADC(long typeDevice, long numberDSP, long numberChannel,
*double *amplify)* *(Zadc.dll)*
опрос коэффициента усиления предварительного усилителя выбранного канала.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – выбор номера канала, в котором необходимо произвести опрос коэффициента усиления предварительного усилителя и должен принимать значения от 0 – для первого канала до 7 для восьмого канала,
amplify – возвращаемое значение установленного коэффициента усиления предварительного усилителя.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция меняет признак *modify* в драйвере.

Управление коэффициентами ослабления аттенюатора ЦАП

Опрос коэффициента ослабления аттенюатора выбранного канала
*long ZGetAttenDAC(long typeDevice, long numberDSP, long numberChannel, double *reduction)* *(Zadc.dll)*
опрос коэффициента ослабления аттенюатора выбранного канала
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с нуля.
reduction – точное значение установленного коэфф. ослабления (значение от 0.001 до 1).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установка коэффициента ослабления аттенюатора выбранного канала
*long ZSetAttenDAC(long typeDevice, long numberDSP, long numberChannel, double reductionIn, double *reductionOut)* *(Zadc.dll)*
установка коэффициента ослабления аттенюатора заданного канала аналогового вывода.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с нуля.
reductionIn – задаваемый коэффициента ослабления (значение от 0.001 до 1)
reductionOut – точное значение установленного коэффициента ослабления.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0200 – *numberChannel* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Управление процессом перекачки данных

Оцифрованные данные от аналого-цифрового преобразователя, вернее, от его контроллера поступают в память центрального процессора порциями с размером, равным размеру буфера перекачки. Эти порции данных перекачиваются в процедуре обработки прерываний. Буфер для перекачки, как правило, намного меньше буфера памяти центрального процессора для хранения данных. Такая структура построения обеспечивает непрерывный (без пропусков) ввод оцифрованных данных в буфер памяти центрального процессора и последующую обработку данных программой пользователя. Признаком накопления данных может служить счетчик прерываний, который можно получить при помощи функции *ZGetFlag(),* или указатель процесса накопления в буфере, который можно получить при помощи функции *ZGetPointerADC() (ZGetPointerDAC())*. Одномоментно оцифрованные данные, поступающие от различных каналов аналого-цифрового преобразователя, находятся в одном буфере накопления последовательно друг за другом по мере возрастания номера канала. Например, выбраны каналы для ввода 1, 3, 5, 6, 7. Тогда данные АЦП будут расположены в памяти в следующем порядке (см. табл. 1)

Таблица 1.

индекс буфера накопления	номер канала	номер кадра
0	1	0
1	3	0
2	5	0
3	6	0
4	7	0
5	1	1
6	3	1
7	5	1
8	6	1
9	7	1
10	1	2
11	3	2
12	5	2
13	6	2
14	7	2

Установка режима внешней подкачки данных или внутренней генерации сигналов
long ZSetExtCycleDAC(long typeDevice, long numberDSP, long enable) (Zadc.dll)
установка режима работы ЦАП. Цифроаналоговый преобразователь может функционировать в двух режимах:

- в режиме генерации сигналов из внутренней памяти сигнального процессора. В этом случае основным достоинством является то, что не требуется каждый раз пересылать данные в сигнальный процессор и не используются ресурсы центрального процессора. Ограничением является размер буфера для циклической или однократной генерации сигнала. Этот буфер ограничен памятью сигнального процессора и его размер можно узнать с помощью функции ZGetMaxInterruptDAC() (Размер буфера равен удвоенному размеру порции данных за прерывание).
- в режиме генерации сигналов из памяти центрального процессора. В этом режиме используются ресурсы центрального компьютера.

1-ый случай: используется модуль KADSP/PCI и к нему подключены АЦП и ЦАП, при этом модуль АЦП подключен к ведущему сигнальному процессору (с четным номером) и модуль ЦАП – к ведомому процессору (с нечетным номером). В этом случае для корректной работы модуля ЦАП размеры буферов прерываний и частоты дискретизации модулей АЦП и ЦАП должны быть связаны следующим соотношением:

$$\frac{\frac{Size\_in\_ADC}{Number\_of\_channel\_ADC}}{\frac{Size\_in\_DAC}{Number\_of\_channel\_DAC}} = \frac{Freq\_sample\_ADC}{Freq\_sample\_DAC}$$

где $Size\_in\_ADC$ - размер буфера прерывания для АЦП,

$Size\_in\_DAC$ - размер буфера прерывания для ЦАП,

$Number\_of\_channel\_ADC$ - количество включенных каналов АЦП,

$Number\_of\_channel\_DAC$ - количество включенных каналов ЦАП,

$Freq\_sample\_ADC$ - частота дискретизации модуля АЦП,

$Freq\_sample\_DAC$ - частота дискретизации модуля АЦП.

2-ой случай: используется модуль KADSP/PCI и к нему подключены два АЦП. В этом случае для корректной работы модулей АЦП размеры буферов прерываний и частоты дискретизации модулей должны быть связаны аналогичным соотношением.

3-ий случай: используются контроллеры USB. В этом случае размер буфера прерывания либо жестко задан, либо кратен объему полезных данных в одном пакете USB. Потоки данных АЦП и ЦАП являются независимыми и обрабатываются параллельно. Поэтому следить за соотношением кол-ва включенных каналов частот дискретизации и размеров пакетов прерываний для этих типов устройств не нужно.

typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – параметр для выбора режима работы ЦАП
0 – режим генерации сигнала из внутренней памяти сигнального процессора,
не 0 – режим генерации сигнала из памяти центрального процессора,
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Опрос размера буфера для перекачки данных АЦП/ЦАП
*long ZGetInterruptADC(long typeDevice, long numberDSP, long * size)* *(Zadc.dll)*
*long ZGetInterruptDAC(long typeDevice, long numberDSP, long * size)*
опрос размера буфера для перекачки данных за одно прерывание
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
size – размер буфера для перекачки данных за одно прерывание.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,

0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Опрос максимального размера буфера для перекачки данных АЦП/ЦАП
*long ZGetMaxInterruptADC(long typeDevice, long numberDSP, long * size)* *(Zadc.dll)*
*long ZGetMaxInterruptDAC(long typeDevice, long numberDSP, long * size)*
опрос размера буфера для перекачки данных за одно прерывание
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
size – размер буфера для перекачки данных за одно прерывание.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установка размера буфера для перекачки данных АЦП/ЦАП
long ZSetInterruptADC(long typeDevice, long numberDSP, long size) *(Zadc.dll)*
long ZSetInterruptDAC(long typeDevice, long numberDSP, long size)
установка размера буфера памяти сигнального процессора для перекачки данных из памяти сигнального процессора в память компьютера во время каждого прерывания. Размер *size* не может превышать значения максимального размера. Чем меньше размер буфера, тем меньше время запаздывания от оцифровки данных до обработки данных центральным процессором и меньше время реакции в системах реального времени с обратной связью, но при этом возрастают требования к ресурсам компьютера для обработки прерываний.

Для удобства обработки оцифрованных сигналов размер буфера рекомендуется делать кратным количеству включенных каналов, например 256*numberChannel.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
size – размер буфера для перекачки данных, значение задается в словах (отсчетах АЦП/ЦАП).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Установка размера буфера памяти накопления АЦП/ЦАП
long ZSetBufferSizeADC(long typeDevice, long numberDSP, long size) *(Zadc.dll)*
long ZSetBufferSizeDAC(long typeDevice, long numberDSP, long size)
устанавливает размер буфера накопления в памяти процессора. Размер не может превышать размер буфера драйвера, запрашиваемый при загрузки операционной системы. Накопление в память центрального процессора может быть циклическим или однократным. Метод накопления задается функцией *ZSetCycleSampleADC() (ZSetCycleSampleDAC())*. При циклическом накоплении, программа пользователя может отслеживать текущий указатель накопления буфера по функции *ZGetPointerADC() (ZGetPointerDAC())*, и определять, сколько новых данных записалось в память после последнего обращения. Для удобства работы размер буфера желательно делать кратным количеству включенных каналов.

Размер буфера в циклическом режиме накопления желательно устанавливать большим, чтобы не возникало эффекта уничтожения предыдущих оцифрованных данных.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
size – размер буфера в памяти центрального процессора, размер задается в 16-разрядных словах.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция меняет признак *modify* в драйвере.

Отображение буфера памяти накопления АЦП/ЦАП
*long ZGetBufferADC(long typeDevice, long numberDSP, void **buffer, long *size)* *(Zadc.dll)*
*long ZGetBufferDAC(long typeDevice, long numberDSP, void **buffer, long *size)*
Отображение буфера данных АЦП/ЦАП, опрос адреса и размера буфера данных АЦП/ЦАП. Буфер расположен в системной области памяти и резервируется драйвером. Это позволяет использовать данные из буфера одновременно нескольким программам. В конце программы необходимо дать команду освобождения буфера *ZRemBufferADC() (ZRemBufferDAC())*.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
buffer – адрес буфера в памяти процессора.
size – размер буфера в памяти центрального процессора в 16-разрядных словах.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Освобождение буфера памяти накопления АЦП/ЦАП
*long ZRemBufferADC(long typeDevice, long numberDSP, void **buffer)* *(Zadc.dll)*
*long ZRemBufferDAC(long typeDevice, long numberDSP, void **buffer)*
Освобождает дескрипторы отображения памяти для доступа к буферу данных драйвера. Эта команда подается в конце программы для того, чтобы освободить ресурсы операционной системы. Операционная система может поддерживать открытый доступ для ~ 2000 буферов.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
buffer – адрес буфера в памяти процессора.
Возвращаемое значение:

0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Установка циклического или одноразового накопления АЦП/ЦАП
long ZSetCycleSampleADC(long typeDevice, long numberDSP, long enable) *(Zadc.dll)*
long ZSetCycleSampleDAC(long typeDevice, long numberDSP, long enable)
установка циклического или одноразового накопления в буфер данных центрального процессора.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
enable – тип накопления, 0 – однократное накопление в буфер данных, 1 – циклическое накопление в буфер данных.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Пересылка последних накопленных данных АЦП
*long ZGetLastDataADC(long typeDevice, long numberDSP, long numberChannel, void *buffer, long size)* *(Zadc.dll)*
пересылает последние полученные отсчеты по заданному каналу в буфер данных пользователя.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала. Нумерация начинается с нуля.
buffer – адрес зарезервированного буфера в программе пользователя,
size – размер буфера в памяти центрального процессора (в 16-разрядных словах).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Опрос указателя накопления в буфер данных АЦП/ЦАП
*long ZGetPointerADC(long typeDevice, long numberDSP, long *pointer)* *(Zadc.dll)*
*long ZGetPointerDAC(long typeDevice, long numberDSP, long *pointer)*
опрос указателя накопления в буфер данных центрального процессора.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
pointer – адрес указателя следуещего отсчета АЦП/ЦАП (индекс целочисленного массива 16-разрядных слов), может принимать значения от 0 до *size*-1 (размер буфера в памяти центрального процессора минус единица).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.

Функция не меняет признак *modify* в драйвере.

Опрос флага прерываний
*long ZGetFlag (long typeDevice, long numberDSP, unsigned long *flag)* *(Zadc.dll)*
Функция опрашивает флаг прерываний. Флаг это количество прерываний произошедших после команды *ZStartADC()* Функция необходима для проверки накопления данных. В программе можно организовать периодический опрос флага и при изменении значения флага обрабатывать очередную порцию данных.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
flag – количество прерываний с начала накопления данных.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Опрос состояния накопления данных АЦП/ЦАП
*long ZGetStartADC(long typeDevice, long numberDSP, long *status)* *(Zadc.dll)*
*long ZGetStartDAC(long typeDevice, long numberDSP, long *status)*
Функция проверяет состояние сигнального процессора. Эта функция полезна при однократном накоплении данных.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
status – состояние сигнального процессора. Значение
1 – процесс ввода продолжается,
0 – накопление закончено или процесс ввода не начат.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Старт накопления данныхАЦП/ЦАП
long ZStartADC(long typeDevice, long numberDSP) *(Zadc.dll)*
long ZStartDAC(long typeDevice, long numberDSP)
Функция запускает процесс накопления данных в память буфера накопления данных.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Останов накопления данных АЦП/ЦАП
long ZStopADC(long typeDevice, long numberDSP) *(Zadc.dll)*
long ZStopDAC(long typeDevice, long numberDSP)
Функция останавливает процесс накопления данных АЦП/ЦАП.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция меняет признак *modify* в драйвере.

Управление модулем НСР

Опрос поддержки и подключения модуля НСР
*long ZFindHCPADC(long typeDevice, long numberDSP, long *present)* *(Zadc.dll)*
Функция опрашивает поддерживается ли модуль НСР для питания датчиков стандарта ICP.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
present – подключен ли модуль НСР. Значение
1 – модуль НСР подключен и им можно управлять,
0 – модуль НСР не найден.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается (не поддерживается НСР).
Функция не меняет признак *modify* в драйвере.

Опрос режима работы заданного канала модуля НСР
*long ZGetHCPADC(long typeDevice, long numberDSP, long numberChannel long *enable)* *(Zadc.dll)*
Функция опрашивает, включено ли питание постоянным током по заданному каналу модуля НСР.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.).
enable – режим работы модуля НСР для заданного канала. Значение
1 – питание по заданному каналу модуля НСР включено,
0 – питание по заданному каналу модуля НСР выключено.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается (не поддерживается НСР).
Функция не меняет признак *modify* в драйвере.

Установка режима работы заданного канала модуля НСР
long ZSetHCPADC(long typeDevice, long numberDSP, long numberChannel, long enable) *(Zadc.dll)*
Функция включает-выключает питание постоянным током по заданному каналу модуля НСР.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
numberChannel – номер канала, нумерация начинается с 0 (например, 0,1,2,3 и т.д.).
enable – установка режима работы модуля НСР для заданного канала. Значение
1 – включить питание по заданному каналу модуля НСР,
0 – выключить питание по заданному каналу модуля НСР.

Возвращаемое значение:

0 – нормальное выполнение функции,

0x0002 – функция и режим работы DSP несовместимы (ADC – DAC),

0x0100 – *numberDSP* задан неправильно,

0x0800 – ошибка открытия драйвера,

0xFF00 – для данного типа устройства функция не поддерживается (не поддерживается HCP).

Функция меняет признак *modify* в драйвере.

Управление цифровым портом

Опрос маски выходов цифрового порта
long ZGetDigOutEnable(long typeDevice, long numberDSP,
*unsigned long *digitalOutEnableMask)* *(Zadc.dll)*
Функция опрашивает битовую маску разрешения работы на выход входов-выходов цифрового порта.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
digitalOutEnableMask – маска выходов цифрового порта. 1 – разрешение работы в качестве выхода, 0 – работа в качестве входа.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Установить маску выходов цифрового порта
long ZSetDigOutEnable(long typeDevice, long numberDSP,
unsigned long digitalOutEnableMask) *(Zadc.dll)*
Функция устанавливает входы-выходы цифрового порта в режим входа согласно битовой маске.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
digitalOutEnableMask – маска выходов цифрового порта. 1 – разрешение работы в качестве выхода, 0 – работа в качестве входа.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Прочитать данные с входов цифрового порта
*long ZGetDigInput(long typeDevice, long numberDSP, unsigned long *digitalInput)* *(Zadc.dll)*
Функция производит чтение цифрового порта.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
digitalInput – данные цифрового порта.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Прочитать данные, выдаваемые на выходы цифрового порта
*long ZGetDigOutput(long typeDevice, long numberDSP, unsigned long *digitalOutput) (Zadc.dll)*
Функция считывает данные, которые выдаются на выходы цифрового порта.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
digitalOutput – данные, выдаваемые в цифровой порт.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.

Записать данные в цифровой порт
long ZSetDigOutput(long typeDevice, long numberDSP, unsigned long digitalOutput) (Zadc.dll)
Функция записывает данные по выходам цифрового порта.
typeDevice – тип устройства.
numberDSP – номер сигнального процессора (значение от 0 до 9).
digitalOutput – данные, выдаваемые в цифровой порт.
Возвращаемое значение:
0 – нормальное выполнение функции,
0x0100 – *numberDSP* задан неправильно,
0x0800 – ошибка открытия драйвера,
0xFF00 – для данного типа устройства функция не поддерживается.
Функция не меняет признак *modify* в драйвере.