2016

Руководство программиста

Модули «ОСПЧ». Библиотека функций

Данное руководство предназначено для разработчиков программного обеспечения на базе устройств «ОСПЧ». Содержит описание функций работы с устройствами и их применение

Rev. 1.02

Libospch SDK

Декабрь , 2016

СОДЕРЖАНИЕ

1 Общие сведения

Данное руководство предназначено для разработчиков программного обеспечения в среде операционной системы Microsoft Windows по управлению устройствами «ОСПЧ-Е3», «ОСПЧ-Е2», «ОСПЧ-Е1», «ОСПЧ-Е» и «ОСПЧ-М1», далее - «ОСПЧ».

2 Описание и назначение. Состав

Программное обеспечение «Libospch SDK» предназначено для настройки, управления и контроля устройств «ОСПЧ» и включает:

- драйвер устройств (*dgspm.sys*);
- библиотеку взаимодействия с драйвером (*libio.dll*);
- библиотеки взаимодействия с устройством (*ospche1.dll, ospche.dll, ospchm1.dll*);
- руководство программиста (*OSPCH.programmers.guide.pdf*);

Драйвер обеспечивает обнаружение и закрепление ресурсов устройства в операционной системе, включая распределение базовых адресов, выделение и отображение в пользовательское пространство непрерывных блоков физической памяти, обработку и сигнализацию поступления аппаратного прерывания.

Функции настройки и управления устройств реализованы в библиотеках *ospch*.dll*. Функции низкоуровневого доступа к устройству реализованы в библиотеке *libio.dll*. Библиотеки должны находиться в одной директории с исполняемым модулем.

Библиотеки разработаны в среде Microsoft Visual Studio на языке C++ и реализованы в виде динамически подключаемых модулей (DLL) для ОС Windows с интерфейсом WINAPI.

3 Библиотека функций

В данном разделе описывается набор функций для управления устройствами «ОСПЧ». Все функции библиотеки, за небольшим исключением, возвращают код ошибки. Нулевое значение кода свидетельствует о выполнении функции без ошибок. Если возвращаемое значение функции отлично от нуля, то при выполнении функции произошла ошибка, сведение о которой можно получить с помощью функции *GetErrorMessage*.

Практически в каждой функции одним из параметров передается указатель на порядковый номер устройства, с которым ведется работа, *DeviceNumber*. При количестве входных параметров большем одного, описание параметра *DeviceNumber* будет опускаться.

Выравнивание используемых структур для передачи в качестве параметра должно быть равно единице.

3.1 Количество поддерживаемых устройств

Прототип
ULONG DeviceSupportedCount (void);
Назначение
Получение числа обнаруженных устройств
Типы аргументов
Нет
Возвращаемое значение
Число сконфигурированных в операционной системе устройств «DGSPM»

3.2 Подключение и отключение устройства

В начале работы с устройством необходимо создать и открыть интерфейс взаимодействия с устройством путем вызова функции *CreateDeviceInstance* и передачи в нее требуемого номера устройства в качестве параметра.

Прототип	
int CreateDeviceInstance (PUCHAR DeviceNumber);	
Назначение	
Создание интерфейса взаимодействия с устройством	
Типы аргументов	
DeviceNumber	▪ указатель на порядковый номер устройства
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции.	

По завершению работы с устройством следует закрыть и удалить интерфейс взаимодействия с устройством посредством вызова функции *ReleaseDeviceInstance*.

Прототип	
void ReleaseDeviceInstance (PUCHAR DeviceNumber);	
Назначение	
Удаление интерфейса взаимодействия с устройством	
Типы аргументов	
DeviceNumber	▪ указатель на порядковый номер устройства
Возвращаемое значение	
Нет	

3.3 Получение сведения об ошибке

Если возвращаемое значение функции отлично от нуля, то произошла ошибка при вызове или при выполнении функции. Описание ошибки можно получить с помощью функции *GetErrorMessage*. Вызов функции должен быть выполнен сразу после выполнения функции, вернувшей код ошибки.

Прототип
char* GetErrorMessage (PUCHAR DeviceNumber);
Назначение
Получение расширенной информации об ошибке
Типы аргументов
DeviceNumber ▪ указатель на порядковый номер устройства
Возвращаемое значение
Информация об ошибке в формате строки с завершающим нулем.

3.4 Получение полного имени устройства

Прототип
char* GetFullDeviceName (PUCHAR DeviceNumber);
Назначение
Получение уникального имени устройства, зарегистрированного в формате хранилища ОС Windows
Типы аргументов
DeviceNumber ▪ указатель на порядковый номер устройства
Возвращаемое значение
Имя устройства в формате строки с завершающим нулем. Например:

\\?\pci#ven_bbbb&dev_001c&subsys_001cbbbb&rev_00#4&9bd1621&0&0008#{feacd89e-a40a-4e39-8aac-de6b07c114d2}

3.5 Номер слота устройства

В общем случае номер слота устройства может не совпадать с его порядковым номером. Для подключения к событиям окончания пересылки данных следует использовать в качестве номера устройства возвращаемое значение функции *GetDeviceSlotID*.

Прототип
ULONG GetDeviceSlotID (PUCHAR DeviceNumber);
Назначение
Получение номера слота устройства, пронумерованного слоем аппаратных абстракций (HAL) ОС Windows.
Типы аргументов
DeviceNumber ▪ указатель на порядковый номер устройства
Возвращаемое значение
Номер слота, закрепленный операционной системой при поиске и обнаружении устройства.

3.6 Загрузка синтезатора опорной частоты

Для устройств «ОСПЧ-Е3», «ОСПЧ-Е2» и «ОСПЧ-Е1» существует возможность задания синхронизации частоты опорного генератора от внутреннего или от внешнего источника.

Прототип
int rSynthLoad (PUCHAR DeviceNumber,

PUCHAR RefType);
Назначение
Загрузка синтезатора опорной частоты
Типы аргументов
pRefType ▪ указатель на переменную выбора опорной частоты 0 – внутренний термостатированный кварцевый генератор с частотой 100 МГц и температурной нестабильностью $2 \cdot 10^{-8}$; 1 – внешний высокостабильный источник эталонной частоты 10 МГц.
Возвращаемое значение
Нулевое значение характеризует успешное выполнение функции.

3.7 Загрузка ПЛИС

Устройство содержит несколько ПЛИС, которые доступны для загрузки извне заранее подготовленными конфигурациями в виде бинарных файлов. ПЛИС идентифицируются по названию: «DEM», «DEC», «RS».

Прототип
int FPGALoad (PUCHAR DeviceNumber, PCHAR szFPGAName, PCHAR szFilename);
Назначение
Загрузка ПЛИС заданной конфигурацией

Типы аргументов	
szFPGAName	▪ название ПЛИС для загрузки
szFilename	▪ путь к файлу конфигурации ПЛИС
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции.	

3.8 Загрузка ОЗУ

При загрузке ОЗУ для устройств «ОСПЧ-Е» и «ОСПЧ-М1» следует предварительно прогрузить ПЛИС «DEM» отдельным файлом конфигурации (*load_ram.bit*), предназначенным для последующей загрузки данных в ОЗУ. После загрузки ОЗУ необходимыми данными, требуется перезагрузить ПЛИС «DEM» файлом рабочей конфигурации.

Для устройств «ОСПЧ-Е3», «ОСПЧ-Е2» и «ОСПЧ-Е1» загрузка ОЗУ производится без предварительной загрузки ПЛИС «DEM», из рабочей конфигурации.

Прототип
int RAMLoad (PUCHAR DeviceNumber, PCHAR szFilename);
Назначение
Загрузка ОЗУ данными из файла
Типы аргументов
szFilename ▪ путь к файлу загрузки ОЗУ
Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции.

3.9 Инициализация АЦП

Аналого-цифровой преобразователь (АЦП) инициализируется путем настройки соответствующих регистров при помощи вызова функции *ADCLoad*.

Прототип
`int ADCLoad (PUCHAR DeviceNumber);`
Назначение
Настройка аналого-цифрового преобразователя
Типы аргументов
DeviceNumber ▪ указатель на порядковый номер устройства
Возвращаемое значение
Нулевое значение характеризует успешное выполнение функции.

3.10 Управление первой схемой регулировки усиления

Настройка первой схемы регулировки усиления (РУ) осуществляется посредством заполнения соответствующих полей структуры управления *FIRST_GC* или *FIRST_GC_EX* и передачи указателя на нее в качестве параметра в функцию *FirstGainControl*.

Прототип

`int FirstGainControl (`
`  PUCHAR DeviceNumber,`
`  PFIRST_GC FirstGC,`
`);`

Назначение
Настройка первой схемы РУ
Типы аргументов
FirstGC ▪ указатель на структуру управления первой схемы регулировки усиления

```
/* Структура для стандартного L-конвертора */
typedef struct _FIRST_GC {
union {
        struct {
        // D0...D11  : код усиления при РРУ
        DWORD       Gain          :12;
        // D12..D13 : зарезервировано
        DWORD       Reserved0     :2;
        // D14       : импульсный режим работы АРУ
        DWORD       PulseMode     :1;
        // D15       : 0 - АРУ, 1 - РРУ
        DWORD       Manual        :1;
        // D16..D20 : код управления аттенюатором
        DWORD       Att           :5;
        // D21..D23 : зарезервировано
        DWORD       Reserved1     :3;
        // D24..D31 : значение порога работы АРУ
        DWORD       Threshold     :8;
        } bits;
        DWORD       AsULONG;
};
        _FIRST_GC() {AsULONG = 0;};
```

```
} FIRST_GC, *PFIRST_GC;
/* Структура для модернизированного L-конвертора */
typedef struct _FIRST_GC_EX {
union {
        struct {
        // D0...D11  : нижняя или верхняя граница "нечувствительности"
        //            первой АРУ
        DWORD      Level          :12;
        // D12       : 0 - запись нижней границы, 1 - верхней
        DWORD      LHSwitch       :1;
        // D13       : 0 - стандартный конвертор, 1 - модернизированный
        DWORD      Converter      :1;
        // D14       : импульсный режим работы АРУ
        DWORD      PulseMode      :1;
        // D15       : 0 - АРУ, 1 - РРУ
        DWORD      Manual         :1;
        // D16..D22  : код управления аттенюатором
        DWORD      Att            :7;
        // D23       : зарезервировано
        DWORD      Reserved1      :1;
        // D24..D31  : прописать все биты 0
        DWORD      Reserved2      :8;
        } bits;
        DWORD      AsULONG;
};
        _FIRST_GC_EX() {AsULONG = 0;};
} FIRST_GC_EX, *PFIRST_GC_EX;
```


Возвращаемое значение
Нулевое значение характеризует успешное выполнение функции.

3.11 Управление второй схемой регулировки усиления

Настройка второй схемы регулировки осуществляется путем заполнения соответствующих полей структуры *SECOND_GC* и передачи указателя на нее в качестве параметра в функцию *SecondGainControl*.

Прототип

```
int  SecondGainControl (
        PUCHAR          DeviceNumber,
        PSECOND_GC      SecondGC,
);
```

Назначение

Настройка второй схемы РУ

Типы аргументов

SecondGC — ▪ указатель на структуру управления второй схемы регулировки усиления

```
typedef struct _SECOND_GC {
union {
        struct {
        // D0...D15  : код усиления при РРУ
        DWORD       Gain          :16;
        // D16       : 0 - АРУ, 1 - РРУ
        DWORD       Manual        :1;
        // D17..D23  : зарезервировано
        DWORD       Reserved0     :7;
```

```
        // D24..D26 : три бита управления постоянной времени АРУ (T_hi)
        DWORD       TimeHi        :3;
        // D27      : зарезервировано
        DWORD       Reserved1     :1;
        // D28..D30 : три бита управления постоянной времени АРУ (T_lo)
        DWORD       TimeLo        :3
        // D31      : зарезервировано
        DWORD       Reserved2     :1;
        } bits;
        DWORD       AsULONG;
} ctrl;
union {
        struct {
        // D0...D7  : значение порога работы АРУ
        DWORD       Threshold     :8;
        // D8..D15  : зарезервировано
        DWORD       Reserved0     :8;
        // D16..D27 : ограничитель максимального усиления АРУ
        DWORD       Limit         :12;
        // D28..D31 : зарезервировано
        DWORD       Reserved1     :4;
        } bits;
        DWORD       AsULONG;
} thresh;
        _SECOND_GC () {ctrl.AsULONG = thresh.AsULONG = 0;};
} SECOND_GC, *PSECOND_GC;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции.

3.12 Управление ФАПЧ

Настройка ФАПЧ осуществляется путем заполнения соответствующих битовых полей структуры управления *PLL_CONTROL* и передачи указателя на нее в качестве параметра в функцию *PLLControl*.

Прототип

```
int  PLLControl (
        PUCHAR              DeviceNumber,
        PPLL_CONTROL        PLLCtrl,
);
```

Назначение

Настройка ФАПЧ

Типы аргументов

PLLCtrl — ▪ указатель на структуру управления ФАПЧ

```
typedef struct _PLL_CONTROL {
union {
        struct {
        // D0…D1    : полоса ПИФ; 0 – 1/50 , 1 – 1/100 , 2 – 1/200 , 3 – 1/400 от
        //           установленного значения тактовой частоты
        DWORD       Band                :2;
        // D2       : для PI-режимов
        DWORD       PIMode              :1;
        // D3       : поворот ВД на 45° для ФМ4, на 22.5° для ФМ8
        DWORD       CDRotation          :1;
```

```
        // D4..D12   : зарезервировано
        DWORD     Reserved0         :9;
        // D13       : отключение слежения за тактовой частотой
        DWORD     TFTraceDisable    :1;
        // D14       : включение АПЧ
        DWORD     PLLEnable         :1;
        // D15       : отключение слежения за несущей частотой
        DWORD     CFTraceDisable    :1;
        // D16..D23  : порог синхронизации по несущей
        DWORD     CFThreshold       :8;
        // D24..D31  : число, определяемое количеством точек сигнального
        // созвездия (для ОСПЧ-Е3/Е2/Е1). Записывается (2^бит/символ - 1).
        // Зарезервировано для ОСПЧ-Е/М1.
        DWORD     Reserved1         :8;
        } bits;
        DWORD     AsULONG;
};
        _PLL_CONTROL() {AsULONG = 0;};
} PLL_CONTROL, *PPLL_CONTROL;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции.

3.13 Несущая частота

При работе в L-диапазоне несущая частота задается в пределах от 950 МГц до 2150 МГц.

3.13.1 Установка несущей частоты

Задание несущей частоты осуществляется путем вызова функции *SetCarrier* и передачи ей в качестве параметра указателя на переменную, содержащую необходимое значение частоты в герцах.

Прототип	
int SetCarrier (PUCHAR DeviceNumber, PDOUBLE Carrier);	
Назначение	
Установка несущей частоты	
Типы аргументов	
Carrier	▪ указатель на переменную типа *double* со значением несущей частоты
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции.	

3.13.2 Получение измеренного значения несущей частоты

В устройстве имеется возможность измерения истинного значения несущей частоты. Получить рассчитанное значение можно при помощи вызова функции *MeasureCarrier*.

Прототип
int MeasureCarrier (PUCHAR DeviceNumber,

```
    PVOID     Reserved,
    PDOUBLE   MeasuredCarrier
);
```

Назначение	
Получение измеренного значения несущей частоты	
Типы аргументов	
Reserved	▪ зарезервировано, установить в NULL
MeasuredCarrier	▪ указатель на переменную типа *double*
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции. При успешном вызове функции содержимое третьего аргумента будет соответствовать измеренному значению несущей частоты в герцах.	

3.14 Тактовая частота

Значение тактовой частоты задается в герцах в соответствии с Таблицей 1.

3.14.1 Задание тактовой частоты

Для установки тактовой частоты в функцию *SetTiming* необходимо передать указатель на переменную, содержащую значение тактовой частоты в герцах. Значение может находиться в интервале $\left[\frac{F_{АЦП}}{2^{15}}, \frac{F_{АЦП}}{2^{2}}\right]$, где $F_{АЦП}$ – частота дискретизации АЦП, равная 186 666 666,(6) Гц для конвертора L/140 и 280 МГц – для конвертора L/350.

Прототип

```
int  SetTiming (
    PUCHAR   DeviceNumber,
```


PDOUBLE Timing);
Назначение
Установка тактовой частоты
Типы аргументов
Timing ▪ указатель на переменную типа *double* со значением тактовой частоты
Возвращаемое значение
Нулевое значение характеризует успешное выполнение функции.

3.14.2 Получение измеренного значения тактовой частоты

Рассчитанное значение тактовой частоты может быть получено при помощи вызова функции *MeasureTiming*.

Прототип
int MeasureTiming (PUCHAR DeviceNumber, PVOID Reserved, PDOUBLE MeasuredTiming);
Назначение
Получение измеренного значения тактовой частоты
Типы аргументов
Reserved ▪ зарезервировано, установить в NULL MeasuredTiming ▪ указатель на переменную типа *double*

Возвращаемое значение
Нулевое значение характеризует успешное выполнение функции. При успешном вызове функции содержимое третьего аргумента будет соответствовать измеренному значению тактовой частоты в герцах.

3.14.3 Получение параметров задания тактовой частоты

Функция *GetTimingControl* позволяет получить параметры задания тактовой частоты, определенные в Таблице 1.

Прототип
int GetTimingControl (PUCHAR DeviceNumber, PTIMING_CTRL TFCtrl);
Назначение
Получение параметров задания тактовой частоты
Типы аргументов

TFCtrl ▪ указатель на структуру параметров тактовой частоты
typedef struct _TIMING_CTRL {
// количество выборок на символ
DWORD Kvps;
union {
struct {
// D0...D11 : коэффициент децимации
DWORD Kdec :12;
// D12..D15 : зарезервировано

```
            DWORD       Reserved    :4;
            // D16..D19 : коэффициент сдвига
            DWORD       Kshft       :4;
            } bits;
        DWORD       AsULONG;
        } coeff;
        DWORD       Diap;
        DWORD       Kpt;
        _TIMING_CTRL () {coeff.AsULONG = 0;};
} TIMING_CTRL, *PTIMING_CTRL;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции.

В случае успешного выполнения функции содержимое второго аргумента будет заполнено полями структуры *TIMING_CTRL*

3.14.4 Управление режимом тактовой частоты

Для задания режима тактовой частоты необходимо заполнить структуру управления *TIMING_MODE* и передать указатель на нее в качестве параметра в функцию *SetTimingMode*.

Прототип

```
int  SetTimingMode (
        PUCHAR              DeviceNumber,
        PTIMING_MODE        TFMode
);
```

Назначение

Управление режимом тактовой частоты

Типы аргументов

TFMode ▪ указатель на структуру управления режимом тактовой частоты

```
typedef struct _TIMING_MODE {
union {
        struct {
        // D0          : 1 – для ФМ4С, 0 – во всех остальных режимах
        DWORD       Mode          :1;
        // D1          : управление инверсией сигнала ТЧ на выходе
        //             демодулятора
        DWORD       Inverse       :1;
        // D2...D7     : зарезервировано
        DWORD       Reserved0     :6;
        // D8...D11    : зарезервировано
        DWORD       Reserved1     :4;
        // D12..D31    : зарезервировано
        DWORD       Reserved2     :20;
        } bits;
        DWORD       AsULONG;
};
        _TIMING_MODE() {AsULONG = 0;};
} TIMING_MODE, *PTIMING_MODE;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции.

3.15 Загрузка фильтра

Для различных видов сигнала реализованы соответствующие цифровые фильтры, коэффициенты которых последовательно представлены в текстовом файле в формате *long double (%le)* с разделяющим пробелом. Для задания фильтра необходимо указать путь к директории с файлами коэффициентов фильтра, идентификатор префикса имени фильтра и количество выборок на символ.

Прототип	
int FilterLoad (PUCHAR DeviceNumber, PCHAR szFilterDir, PUCHAR ID, PUCHAR CPS);	
Назначение	
Загрузка коэффициентов цифрового фильтра	
Типы аргументов	
szFilterDir	▪ путь к каталогу файлов коэффициентов фильтра
ID	▪ указатель на переменную, содержащую идентификатор префикса имени фильтра
CPS	▪ указатель на переменную, содержащую количество выборок на символ
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции	

Путь к каталогу с файлами коэффициентов фильтра может быть как абсолютным, так и относительным. Идентификатор префикса имени фильтра выбирается в соответствии с Таблицей 2.

Таблица 2. Идентификатор префикса имени фильтра

ID	Префикс	Примечание
0	ibsd	фильтр в соответствии со спецификацией IESS-308/309
1	n10d	фильтр Найквиста с коэффициентом скругления 0.1
2	n15d	фильтр Найквиста с коэффициентом скругления 0.15
3	n20d	фильтр Найквиста с коэффициентом скругления 0.2
4	n25d	фильтр Найквиста с коэффициентом скругления 0.25
5	n30d	фильтр Найквиста с коэффициентом скругления 0.3
6	n35d	фильтр Найквиста с коэффициентом скругления 0.35
7	n40d	фильтр Найквиста с коэффициентом скругления 0.4
8	nf40d	фильтр Найквиста для нефильтрованных на передающей стороне сигналов с коэффициентом скругления 0.4
9	wded	"широкий" фильтр, предназначенный для работы в режиме построения панорамы по сигналам каналов I и Q, а также для записи реализаций сигнала

Количество выборок на символ может принимать значения 4, 8, 16, 32. В последних версиях библиотек этот параметр рассчитывается автоматически. Поэтому его можно инициализировать произвольным значением.

3.16 Управление IQ-каналом

Управление каналом синфазно-квадратурных данных осуществляется путем задания полям структуры *IQCH_CTRL* соответствующих значений и передачи указателя на заполненную структуру в качестве одного из аргументов функции *IQControl*.

Прототип

```
int  IQControl (
        PUCHAR          DeviceNumber,
        PIQCH_CTRL      IQCtrl
);
```

Назначение

Управление параметрами передачи данных IQ-канала

Типы аргументов

IQCtrl — ▪ указатель на структуру параметров IQ-канала

```
typedef struct _IQCH_CONTROL {
union {
        struct {
        // D0          : 1 - режим ВД, 0 - режим спектра
        DWORD       Mode          :1;
        // D1…D3       : зарезервировано
        DWORD       Reserved0     :3;
        // D4          : 1 - пересылка данных в режиме Master
        DWORD       Master        :1;
        // D5          : 1 - режим однократных пересылок
        DWORD       SingleMode    :1;
        // D6..D7      : зарезервировано
        DWORD       Reserved1     :2;
        // D8          : 1 - сброс счетчиков
        DWORD       Reset         :1;
        // D9…D11      : зарезервировано
        DWORD       Reserved2     :3;
```

```
        // D12..D14 : размер пересылаемого блока данных
        DWORD       FIFOBlock    :3;
        // D15..D28 : зарезервировано
        DWORD       Reserved3    :14;
        // D29..D30 : 0 - 8 бит; 1 - 16 бит, 2 - сигнал с АЦП; 3 - "пила"
        DWORD       Format       :2;
        // D31      : 1 - пересылка данных в режиме Slave
        DWORD       Slave        :1;
        } bits;
        DWORD       AsULONG;
};
        _IQCH_CONTROL() {AsULONG = 0;};
} IQCH_CTRL, *PIQCH_CTRL;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции

3.17 Управление DMD-каналом

Управление каналом демодулированных данных осуществляется посредством задания полям структуры *DMDCH_CTRL* соответствующих значений и передачи указателя на заполненную структуру в качестве одного из аргументов функции *DMDControl*.

Прототип

```
int  DMDControl (
        PUCHAR          DeviceNumber,
        PDMDCH_CTRL DMDCtrl
);
```


Назначение
Управление параметрами передачи данных DMD-канала
Типы аргументов
DMDCtrl ▪ указатель на структуру параметров DMD-канала

```
typedef struct _DMDCH_CONTROL {
union {
        struct {
        // D0..D3     : упаковка канала
        DWORD      Mode        :4;
        // D4         : 1 - пересылка данных в режиме Master
        DWORD      Master      :1;
        // D5         : зарезервировано
        DWORD      Reserved1   :1;
        // D6         : 1 - изменяет упаковку бит в байты на обратную (ранний -
        //            младший на ранний - старший)
        DWORD      Bitorder    :1;
        // D7         : 1 - изменяет упаковку бит в сигнальном созвездии на
        //            обратную (ранний – младший на ранний - старший) в
        //            режиме записи "Демодулятор-упаковка"
        DWORD      BitorderCD  :1;
        // D8         : 1 - сброс счетчиков
        DWORD      Reset       :1;
        // D9…D11     : зарезервировано
        DWORD      Reserved2   :3;
        // D12..D14   : размер пересылаемого блока данных
```

```
        DWORD       FIFOBlock    :3;
        // D15      : зарезервировано
        DWORD       Reserved3    :1;
        // D16..D20 : код деления частоты
        DWORD       Delimiter    :5;
        // D21..D30 : зарезервировано
        DWORD       Reserved4    :10;
        // D31      : 1 - пересылка данных в режиме Slave
        DWORD       Slave        :1;
        } bits;
        DWORD       AsULONG;
};
        _DMDCH_CONTROL() {AsULONG = 0;};
} DMDCH_CTRL, *PDMDCH_CTRL;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции

3.18 Управление DMA3-каналом

Канал DMA3 представляет собой выборки синфазно-квадратурного канала для построения векторной диаграммы (сигнального созвездия). Для управления DMA3-каналом используются поля структуры *DMA3CH_CONTROL* и функция *DMA3CHControl*.

Прототип

```
int  DMA3CHControl (
        PUCHAR              DeviceNumber,
        PDMA3CH_CTRL        Ctrl
```


); **Назначение**
Управление параметрами передачи данных DMA3-канала
Типы аргументов

Ctrl ▪ указатель на структуру параметров DMA3-канала
typedef struct _DMA3CH_CONTROL {
union {
struct {
// D0..D3 : зарезервировано
DWORD Reserved0 :4;
// D4 : 1 - пересылка данных в режиме Master
DWORD Master :1;
// D5 : 1 - режим однократных пересылок
DWORD SingleMode :1;
// D6..D7 : зарезервировано
DWORD Reserved1 :2;
// D8 : 1 - сброс счетчиков
DWORD Reset :1;
// D9…D11 : зарезервировано
DWORD Reserved2 :3;
// D12..D14 : размер пересылаемого блока данных
DWORD FIFOBlock :3;
// D15..D31 : зарезервировано
DWORD Reserved3 :10;
} bits;

```
        DWORD       AsULONG;
};
        _DMA3CH_CONTROL() {AsULONG = 0;};
} DMA3CH_CTRL, *PDMA3CH_CTRL;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции

3.19 Работа устройства в режиме Busmaster

Устройство имеет возможность передавать данные в нескольких режимах. Одним из них является режим передачи данных путем захвата шины PCI и прямого доступа к памяти – Busmaster DMA.

3.19.1 Получение закрепленных буферов

Для использования режима Busmaster DMA драйвером выделяется определенное количество буферов памяти, которые в дальнейшем отображаются в пользовательское пространство и могут быть получены посредством вызова функции *MasterGetBuffers*. Количество и размер буферов задаются в реестре:

HKLM\SYSTEM\CurrentControlSet\services\dgspm\Parameters, где

BuffersPerCH – количество буферов на канал;

CHCount – количество каналов;

CHBufferLENs – размеры буферов.

Общее количество буферов будет равно произведению числа каналов и количества буферов на канал: *CHCount · BuffersPerCH.*

Прототип

```
int  MasterGetBuffers (
        PUCHAR              DeviceNumber,
        PMEMORY_DATA        MemoryData
);
```

Назначение

Получение закрепленных буферов памяти

Типы аргументов

MemoryData — ▪ указатель на структуру буферов

```
typedef struct _MEMORY_DATA {
        // общее число буферов
        ULONG                   Count;
        // массив буферов
        PMEMORY_BUFFER_UM Buffers;
        // зарезервировано
        ULONG                   Reserved0;
        // зарезервировано
        ULONG                   Reserved1;
} UNALIGNED MEMORY_DATA, *PMEMORY_DATA;

typedef struct _MEMORY_BUFFER_UM {
        // размер буфера в байтах
        ULONG           Size;
        // адрес буфера в пользовательском пространстве
        PULONG          UserVA;
```

```
        // зарезервировано
        PVOID           Reserved;
        // адрес физического размещения буфера
        LARGE_INTEGER   MemoryPA;
        // зарезервировано
        PVOID           Reserved0;
        // зарезервировано
        PVOID           Reserved1;
        // зарезервировано
        PVOID           Reserved2;
} UNALIGNED MEMORY_BUFFER_UM, *PMEMORY_BUFFER_UM;
```

Возвращаемое значение

Нулевое значение характеризует успешное выполнение функции.

В результате успешного выполнения функции, содержимое второго аргумента будет заполнено данными структуры буферов.

Буфера каналов располагаются последовательно на каждый из каналов в массиве *Buffers[]*. Данные размещаются в буферах однократным способом или циклично, когда после окончания заполнения последнего буфера канала начинается заполнение первого (см. *MasterStart*).

Закрепленные буфера необходимо подставить устройству при помощи вызова функции *MasterLoad*. При этом существует возможность изменить в меньшую сторону размеры буферов, указав соответствующее значение в поле *Size* структуры *MEMORY_BUFFER_UM*.

Прототип	
int MasterLoad (PUCHAR DeviceNumber, PMEMORY_DATA MemoryData);	
Назначение	
Загрузка адресов памяти в устройство	
Типы аргументов	
MemoryData	▪ указатель на структуру буферов (см. *MasterGetBuffers*)
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции.	

3.19.2 Запуск и останов передачи данных

Функция *MasterStart* инициализирует передачу данных устройства в режиме Busmaster DMA.

Прототип	
int MasterStart (PUCHAR DeviceNumber, PUINT Mode,);	
Назначение	
Запуск передачи данных устройства	

Типы аргументов	
Mode	▪ указатель на режим передачи данных устройства: однократный (0x80000000) или циклический (0x40000000)
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции.	

Функция *MasterStop* останавливает ранее запущенную передачу данных в режиме Busmaster DMA.

Прототип	
int MasterStop (PUCHAR DeviceNumber);	
Назначение	
Останов передачи данных устройства	
Типы аргументов	
DeviceNumber	▪ указатель на порядковый номер устройства
Возвращаемое значение	
Нулевое значение характеризует успешное выполнение функции.	

3.19.3 Индикация поступления аппаратных прерываний

Завершение устройством обработки очередного буфера данных может быть сигнализировано в виде аппаратного прерывания. Для дальнейшего уведомления программного модуля о поступлении прерывания от

устройства служат сигналы глобальных именованных событий с автоматическим сбросом и форматом имени вида «*dgspmNevntCH*», где:

N – номер слота устройства, полученный при помощи вызова функции *GetDeviceSlotID*;

CH – номер канала: *0 … (CHCount – 1)*.

Количество поступающих прерываний контролируется при помощи вызова функции *MasterGetInterruptInfo*.

Прототип
int MasterGetInterruptInfo (HANDLE hDevice, PINTERRUPT_INFO InterruptInfo);
Назначение
Статистика поступления аппаратных прерываний с момента подключения к устройству
Типы аргументов

InterruptInfo ▪ указатель на структуру INTERRUPT_INFO
typedef struct _INTERRUPT_INFO {
// количество каналов
ULONG Count;
// массив структур CHIRQ_BUFFER
PCHIRQ_BUFFER Buffers;
// зарезервировано
ULONG Reserved0;
// зарезервировано
ULONG Reserved1;

```
} INTERRUPT_INFO, *PINTERRUPT_INFO;


typedef struct _CHIRQ_BUFFER {
        // номер канала
        ULONG           CH;
        // количество прерываний устройства по данному каналу
        ULONG           cntIRQ;
        // количество неуспешных попыток вызова отложенной процедуры
        // обработки прерывания для канала
        ULONG           cntDPCFail;
        // общее число прерываний устройства
        ULONG           TotalIRQ;
        // зарезервировано
        ULONG           Reserved0;
        // зарезервировано
        ULONG           Reserved1;
} CHIRQ_BUFFER, *PCHIRQ_BUFFER;
```

Возвращаемое значение

При успешном выполнении функции поля структуры по указателю InterruptInfo будут заполнены соответствующими значениями.

4 Рекомендуемый порядок работы

1. Определить число поддерживаемых устройств: *DeviceSupportedCount*
2. Найти требуемое устройство по идентификатору и подключиться к нему: *GetFullDeviceName, CreateDeviceInstance*
3. Для устройств ОСПЧ-E3/E2/E1 загрузить синтезатор опорной частоты: *rSynthLoad*

4. Задать тип принимаемого сигнала: *SetSignalType*
5. Получить параметры заданного типа сигнала: *GetSignalParameters*
6. Загрузить ПЛИС и АЦП: *FPGALoad*, *ADCLoad*
7. Инициализировать синтезатор L-диапазона и таблицу напряжений: *lSynthInit*, *InitLConvertorVList*
8. Установить лицензионный ключ (при необходимости): *InstallLicense*
9. Настроить первую и вторую схемы РУ: *FirstGainControl*, *SecondGainControl*
10. Настроить ФАПЧ: *PLLControl*
11. Установить несущую частоту: *SetCarrier*
12. Задать значение и режим работы тактовой частоты: *GetTimingControl*, *SetTiming*, *SetTimingMode*
13. Загрузить фильтр: *FilterLoad*
14. Настроить необходимые каналы передачи данных: *DMDControl*, *IQControl*, *DMA3CHControl*

15.1. Инициализировать и вести прием данных в режиме Busmaster: *MasterStop*, *MasterLoad*, *MasterStart*

15.2. При использовании механизма аппаратных прерываний, ожидать соответствующее событие завершение ввода данных (п. 3.19.3) и контролировать поступление данных: *MasterGetInterruptInfo*

15. Закрыть интерфейс устройства: *ReleaseDeviceInstance*