

Руководство по программированию отчётов и обработке данных с использованием OLE Automation.

Доступ к данным, типы, функции и процедуры, поддерживаемые программным обеспечением системы UniChrom

Оглавление

Регистрация.....	3
Работа с программой.....	3
Работа с дочерними окнами (спектрами).....	3
Управление дочерними окнами	3
Доступ к спектру	3
Количество открытых спектров	4
Завершение выполнения отчёта	4
Закрытие окна спектра	4
Глобальные свойства спектра.....	5
Количество глобальных свойств спектра	5
Доступ к имени глобального свойства по индексу	5
Доступ к значению глобального свойства по индексу	5
Доступ к псевдониму глобального свойства по индексу	5
Получение индекса глобального свойства по имени	5
Получение имени слоя спектра по индексу слоя	6
Доступ к значению глобального свойства по имени	6
Установка нового значения глобального свойства	6
Локальные свойства спектра.....	6
Доступ к имени локального свойства по индексу	7
Доступ к значению локального свойства по индексу	7
Доступ к псевдониму локального свойства по индексу	7
Получение индекса локального свойства по имени	7
Работа со слоями.....	7
Количество слоёв спектра	7
Получение индекса текущего (активного) слоя спектра	8
Информация о пиках.....	8
Определение индекса пика по имени	8
Получение полной информации о пике	8
Доступ к одному из свойств пика	9
Установка новых свойств пика	10
Добавление новых пиков	10
Рисунок хроматограммы.....	10
Информация по калибровке.....	10
Копирование калибровочной кривой	10
Получение калибровочной информации	10
Пример разборки строки, возвращаемой функцией GetCalibTable:.....	11
Управление состоянием спектра.....	11
Получение информации о наличии процесса измерения	11
Управление процессом измерения	12
Запуск сценария обработки спектра	12
Примеры для VB/VBA.....	13

Регистрация

Регистрация библиотеки типов осуществляется автоматически при установке программного обеспечения либо запуском программы UWin32.exe с командной строкой "/regserver".
Дерегистрация библиотеки типов осуществляется запуском программы UWin32.exe с командной строкой "/unregserver":

UWin32.exe /regserver
UWin32.exe /unregserver

Работа с программой

Доступ к программе UWin32.exe по OLE каналу осуществляется вызовом функции CreateObject со строковым параметром "Unichrom.App":

```
Dim Unichrom As Object  
Set Unichrom = CreateObject("Unichrom.App")
```

Функция CreateObject возвращает указатель на программу UWin32.exe (Application) .

Управление состоянием главного окна программы UWin32.exe осуществляется вызовом процедуры MainWndState с целым параметром State:

```
Unichrom.MainWndState(State)
```

Параметр State принимает следующие значения:

0	-	нормальное (Normal) окно
1	-	свёрнутое (Minimized) окно
2	-	распахнутое (Maximized) окно

Работа с дочерними окнами (спектрами)

Управление дочерними окнами

осуществляется следующими процедурами:

Unichrom.CascadeWindows	-	размещение окон каскадом,
Unichrom.TileWindows	-	размещение окон черепицей,
Unichrom.CloseAll	-	закрытие всех окон.

Доступ к спектру

осуществляется следующими функциями:

Создать новый спектр (создание нового окна спектра по шаблону Default.\$\$\$)

Function NewSpec As OleVariant

Открыть спектр с именем файла FileName (FileName – полный путь к файлу спектра)

Function OpenSpec(const FileName As WideString) As OleVariant

Количество открытых спектров

определяется вызовом функции GetSpecCount:

```
Dim SpectrumCount As Integer  
SpectrumCount = Unichrom.GetSpecCount
```

Получить ссылку на спектр по его индексу (SpectrumIndex – индекс спектра в списке открытых спектров – принимает значения 0, 1, 2, ..., SpectrumCount – 1 (см. п. “Количество открытых спектров”))

Function GetSpec(SpectrumIndex As Integer) As OleVariant

Получить ссылку на текущий (активный) спектр

Function GetActiveSpec As OleVariant

Получить ссылку на спектр – клиент (в режиме исполнения сценария обработки спектра)

Function GetActiveClient As OleVariant

Перечисленные выше функции возвращают ссылку на соответствующий спектр. Пример использования этих функций:

```
Dim Spectrum As OleVariant, Client As OleVariant  
Set Spectrum = Unichrom.OpenSpec("C:\Unichrom\Примеси в бензоле.***")  
Set Client = Unichrom.GetActiveClient
```

Завершение выполнения отчёта

при использовании клиентов в режиме исполнения сценария обработки спектров должно сопровождаться вызовом процедуры ClearClient:

Unichrom.ClearClient

Заккрытие окна спектра

осуществляется вызовом процедуры Close:

Spectrum.Close

Глобальные свойства спектра

Количество глобальных свойств спектра

определяется вызовом функции GetPropCount:

```
Dim PropertyCount As Integer  
PropertyCount = Spectrum.GetPropCount  
(PropertyCount = Client.GetPropCount)
```

Доступ к имени глобального свойства по индексу

осуществляется вызовом функции GetPropName:

```
Dim PropertyName As WideString  
PropertyName = Spectrum.GetPropName(PropertyIndex)
```

Индекс глобального свойства спектра PropertyIndex в списке глобальных свойств спектра принимает целые значения 0, 1, 2, ..., PropertyCount – 1

Доступ к значению глобального свойства по индексу

осуществляется вызовом функции GetPropValue:

```
Dim PropertyValue As OleVariant  
PropertyValue = Spectrum.GetPropValue(PropertyIndex)
```

Функция GetPropValue возвращает значения строковых, целых и вещественных свойств спектра.

Альтернативный способ доступа к значению глобального свойства по индексу:

```
PropertyValue = Spectrum.PropValue[PropertyIndex]
```

Доступ к псевдониму глобального свойства по индексу

осуществляется вызовом функции GetPropAlias:

```
Dim PropertyAlias As WideString  
PropertyAlias = Spectrum.GetPropAlias(PropertyIndex)
```

Получение индекса глобального свойства по имени

осуществляется вызовом функции GetPropIndexByName со строковым выражением PropertyName:

```
Dim I As Integer  
I = Spectrum.GetPropIndexByName(PropertyName)
```

Пример использования функции GetPropIndexByName при определении индекса свойства “Частота измерения” с именем Freq:

```
I = Spectrum.GetPropIndexByName("Freq")
```

Получение имени слоя спектра по индексу слоя

LayIndex осуществляется вызовом функции GetLayName:

```
Dim LayName As WideString  
LayName = Spectrum.GetLayName(LayIndex)
```

Номер слоя спектра LayIndex принимает целые значения 0, 1, 2, ..., LayCount (см. п. “Количество слоёв спектра”). Если параметр LayIndex равен нулю, то обращение произойдёт к текущему (активному) слою спектра.

Доступ к значению глобального свойства по имени

осуществляется вызовом функции GetGlobalProperty со строковым параметром PropertyName:

```
Dim PropertyValue As OleVariant  
PropertyValue = Spectrum.GetGlobalProperty(PropertyName)
```

Установка нового значения глобального свойства

осуществляется вызовом функции SetGlobalProperty с указанием имени свойства PropertyName и его нового значения:

```
Dim PropertyValue As OleVariant  
PropertyValue = 5  
Call Spectrum.SetGlobalProperty(PropertyName, PropertyValue)
```

Локальные свойства спектра

Локальные свойства

- свойства, принадлежащие выбранному слою, в других слоях могут отсутствовать. Локальные свойства хранят различные данные для разных слоев.

Количество локальных свойств спектра

определяется вызовом функции GetLocalPropCount с параметром LayIndex целого типа, который определяет номер слоя спектра:

```
Dim LocalPropertyCount As Integer
LocalPropertyCount = Spectrum.GetLocalPropCount(LayIndex)
(LocalPropertyCount = Client.GetLocalPropCount(LayIndex))
```

Доступ к имени локального свойства по индексу

осуществляется вызовом функции GetLocalPropName с двумя параметрами:

```
Dim LocalPropertyName As WideString
LocalPropertyName = Spectrum.GetLocalPropName(LayIndex, PropertyIndex)
```

Доступ к значению локального свойства по индексу

осуществляется следующим образом:

```
Dim LocalPropertyValue As WideString
LocalPropertyValue = Spectrum.LocalPropValue[LayIndex, PropertyIndex]
```

Доступ к псевдониму локального свойства по индексу

осуществляется вызовом функции GetLocalPropAlias:

```
Dim LocalPropertyAlias As WideString
LocalPropertyAlias = Spectrum.GetLocalPropAlias(LayIndex, PropertyIndex)
```

Получение индекса локального свойства по имени

осуществляется вызовом функции GetLocalPropIndexByName: со строковым выражением PropertyName:

```
Dim I As Integer
I = Spectrum.GetLocalPropIndexByName(LayIndex, LocalPropertyName)
```

Параметр LocalPropertyName является строковым выражением. Пример использования функции GetLocalPropIndexByName при определении индекса свойства “Масса навески” с именем Mprobe в пятом слое:

```
Const FifthLay = 5
I = Spectrum.GetPropIndexByName(FifthLay, "Mprobe")
```

Работа со слоями

Количество слоёв спектра

определяется двумя способами. Первый – вызов функции GetLayCount, второй – обращение к свойству спектра:

```
Dim LayCount As Integer  
LayCount = Spectrum.GetLayCount  
LayCount = Spectrum.LayCount
```

Получение индекса текущего (активного) слоя спектра

осуществляется вызовом следующей функции:

```
Dim ActiveLayIndex As Integer  
ActiveLayIndex = Spectrum.GetActiveLay
```

Информация о пиках

Количество пиков в слое определяется вызовом функции GetPeakCountInLay:

```
Dim PeakCount As Integer  
PeakCount = Spectrum.GetPeakCountInLay(LayIndex)
```

Определение индекса пика по имени

осуществляется вызовом функции GetPeakIndexByName с двумя параметрами:

```
Dim PeakIndex As Integer  
PeakIndex = Spectrum.GetPeakIndexByName(LayIndex, PeakName)
```

Параметр PeakName – название пика - является строковым выражением. Пример использования функции GetPeakIndexByName:

```
MethaneIndex = Spectrum.GetPeakIndexByName(FifthLay, "Methane")
```

Функция возвращает индекс пика из списка пиков в указанном слое спектра. Значение индекса принимает целые значения 0, 1, 2, ..., PeakCount - 1

Получение полной информации о пике

осуществляется вызовом функции FullPeakInfo с тремя параметрами:

```
Dim PeakInfo As WideString  
PeakInfo = Spectrum.FullPeakInfo(LayIndex, PeakIndex, Mask)
```

Функция FullPeakInfo возвращает строку свойств пика. Свойства отделены друг от друга точкой с запятой. Свойства пика и их последовательность в возвращаемой строке определяются параметром Mask:

Параметр пика	Мнемоника маски	Значение HEX	Значение DEC
Название пика	mName	\$00000001	1
Время выхода	mTime	\$00000002	2
Индекс удерживания	mIndex	\$00000004	4
Индекс группы	mGroup	\$00000008	8
Масса	mWeight	\$00000010	16
Температура кипения	mTemperature	\$00000020	32
Плотность	mDensity	\$00000040	64
Начало пика	mLeft	\$00000080	128
Конец пика	mRight	\$00000100	256
Площадь	mArea	\$00000200	512
Амплитуда	mAmplitude	\$00000400	1024
Коэффициент чувствительности	mFactor	\$00000800	2048
Объёмная концентрация	mVolume	\$00001000	4096
Массовая концентрация	mMass	\$00002000	8192
Мольная концентрация	mMolar	\$00004000	16384
Титр	mTitre	\$00008000	32768
Молярность	mMolarity	\$00010000	65536
Состояние пика	mState	\$00020000	131072
Число теоретических тарелок (ТТ)	mPlates	\$00040000	262144
Число эффективных теор. тарелок	mEffPlates	\$00080000	524288
Высота экв. эффективной ТТ.	mHeightETP	\$00100000	1048576
Фактор хвостатости пика на высоте 5%	mTailing5	\$00200000	2097152
Полуширина пика	mHalfWidth	\$00400000	4194304
Коэффициент извлечения	mExtraction	\$00800000	8388608
Разрешение со следующим	mResToNext	\$01000000	16777216
Фактор хвостатости пика на высоте 10%	mTailing10	\$02000000	33554432
Полная информация	mTotal	\$0FFFFFFF	268435455

Так, если Mask = 1, то функция FullPeakInfo возвратит строку, содержащую только название пика, например:

```
PeakInfo = "Methane;"
```

Если Mask = 1 + 512 = 513, то функция FullPeakInfo возвратит строку, содержащую название пика и его площадь, например:

```
PeakInfo = "Isooctane;255,001932;"
```

Доступ к одному из свойств пика

осуществляется вызовом функции OneParam, в которую передаются индексы слоя, пика и свойства пика:

```
Dim PeakProperty As OleVariant
PeakProperty = Spectrum.OneParam(LayIndex, PeakIndex, PeakPropertyMask)
```

Значение параметра PeakPropertyMask определяется из таблицы, приведенной в п. "Получение полной информации о пике". Пример получения массовой концентрации 4-го пика из слоя 7-го:

```
MasConc = Spectrum.OneParam(7, 4, 8192)
```

или

```
Const mMass = 8192  
MasConc = Spectrum.OneParam(7, 4, mMass)
```

Функция OneParam возвращает значения строковых, целых и вещественных свойств пика.

Установка новых свойств пика

осуществляется вызовом процедуры SetOneParam, в которую передаются индексы слоя и пика, маска свойства пика и новое значение этого свойства:

```
Spectrum.SetOneParam(LayIndex, PeakIndex, PeakPropertyMask, NewPropertyValue)
```

Тип переменной NewPropertyValue может быть строковый, целый или вещественный.

Добавление новых пиков

осуществляется вызовом функции AddPeak, в которую передаются индексы слоя, а также положения левой границы, центра и правой границы пика. Функция возвращает индекс пика в слое (0 .. число пиков -1)

```
Spectrum.AddPeak(LayIndex, LeftPos, CenterPos, RightPos) as Integer
```

Свойства вновь созданного пика можно изменить с помощью вызова **SetOneParam()**.

Рисунок хроматограммы

Копирование видимой части хроматограммы из слоя с индексом LayIndex в Clipboard осуществляется процедурой GetPicture:

```
Spectrum.GetPicture(LayIndex)
```

Копирование выполняется в формате метафайла Windows (*.wmf)

Информация по калибровке

Копирование калибровочной кривой

для пика с индексом PeakIndex в Clipboard осуществляется процедурой GetCalibPicture:

```
Spectrum.GetCalibPicture(PeakIndex)
```

Копирование выполняется в формате метафайла Windows (*.wmf)

Получение калибровочной информации

по всем пикам спектра осуществляется вызовом функции GetCalibTable:

```
Dim Calibration As WideString  
Calibration = Spectrum.GetCalibTable
```

Функция GetCalibTable возвращает строку, содержащую названия пиков и их калибровочные коэффициенты. Параметры в строке разделены точкой с запятой.

Пример разборки строки, возвращаемой функцией GetCalibTable:

```
' Параметры калибровочной таблицы (Ax{2}+Bx{1}+Cx{0} / exp(C)x{B}  
Const ctName      = 1      'Имя калибровочной последовательности (имя пика)  
Const ctFacA      = 2      'Фактор A  
Const ctFacB      = 3      'Фактор B  
Const ctFacC      = 4      'Фактор C  
Const ctSDev      = 5      'Стандартное отклонение  
Const ctCVal      = 6      'Достоверность аппроксимации (R2)  
Const ctType      = 7      'Тип калибровки  
Const ctMode      = 8      'Параметр калибровки  
  
' Типы калибровки  
Const ctLin0      = 0      'y=Bx{1}  
Const ctLine      = 1      'y=Bx{1}+Cx{0}  
Const ctQuad      = 2      'y=Ax{2}+Bx{1}+Cx{0}  
Const ctPowr      = 3      'y=exp(C)x{B}  
  
' Параметры калибровки  
Const cpArea      = 0      'Калибровка по площади  
Const cpHeight    = 1      'Калибровка по высоте  
  
Function GetCalibData(ByVal PeakName, ByVal Parameter)  
    Dim Table, Result, I, J, K  
    Result = Empty  
    Table = Spectrum.GetCalibTable  
    I = InStr(1, Table, PeakName, vbBinaryCompare)  
    If Not IsEmpty(I) And (I > 0) Then  
        J = I - 1  
        For K = 1 To Parameter  
            I = J + 1  
            J = InStr(I, Table, Separator, vbBinaryCompare)  
        Next  
        Result = Mid(Table, I, J - I)  
    End If  
    GetCalibData = Result  
End Function
```

Управление состоянием спектра

Получение информации о наличии процесса измерения

осуществляется функцией Acquire:

```
Dim SpectrumState As Integer
Const asQuery = 0
SpectrumState = Spectrum.Acquire(asQuery)
```

Функция Acquire возвращает следующие значения:

Значение	Мнемоника	Пояснение
0	asError	Ошибка
1	asUsual	Обычное состояние – нет измерения
2	asWait	Состояние ожидания старта
3	asAcquire	Процесс измерения

Управление процессом измерения

также осуществляется функцией Acquire:

Spectrum.Acquire(asWait)	-	подготовка к старту измерения
Spectrum.Acquire(asAcquire)	-	старт измерения
Spectrum.Acquire(asUsual)	-	остановка измерения

В этом режиме функция Acquire, также как и в предыдущем случае, возвращает текущее состояние спектра.

Запуск сценария обработки спектра

осуществляется следующей функцией:

```
Function StartScenario(CanStart As Integer) As Integer
```

Функция StartScenario возвращает нуль, если сценарий в данный момент не выполняется. В противном случае, функция возвращает единицу.

Параметр CanStart может принимать одно из следующих значений:

Значение	Пояснение
0	Запрещение старта сценария обработки после окончания измерения и остановка сценария, если он в данный момент выполняется
1	Разрешение исполнения сценария обработки спектра после окончания измерения или запуск сценария на выполнение, если в данный момент спектр находится в обычном состоянии (нет ожидания старта и нет измерения)

OLE AUTOMATION ИНТЕРФЕЙС НА ЯЗЫКЕ PASCAL

Функции и свойства, относящиеся к системе UniChrom в целом (UniChrom.App):

```
procedure CascadeWindows;
procedure TileWindows;
procedure CloseAll;
procedure MainWndState(val: Integer);
function NewSpec: OleVariant;
```

```

function OpenSpec(const FileName: WideString): OleVariant;
function GetSpecCount: Integer;
function GetSpec(index: Integer): OleVariant;
function GetActiveSpec: OleVariant;
function GetActiveClient: OleVariant;
procedure ClearClient;

```

Функции и свойства, относящиеся к спектру, его слоям и пикам (UniChrom.SpectraDoc):

Работа со спектром

```

function Acquire(amode: Integer): Integer;
function StartScenario(CanStart: Integer): Integer;
function GetLayCount: Integer;
function GetPropCount: Integer;
function GetPropName(index: Integer): WideString;
function GetPropValue(index: Integer): OleVariant;
function GetPropIndexByName(const pname: WideString): Integer;
function GetPeakCountInLay(nlay: Integer): Integer;
function GetActiveLay: Integer;
function FullPeakInfo(lay: Integer; Num: Integer; amask: Integer): WideString;
function GetPropAlias(ndx: Integer): WideString;
procedure GetPicture(lay: Integer);
procedure GetCalibPicture(Num: Integer);
function GetCalibTable: WideString;
function Get_LayCount: Integer;
procedure Set_LayCount(LayCount: Integer);
function Get_PropValue(index: Integer): WideString;
procedure Set_PropValue(index: Integer; const Value: WideString);
function GetPeakIndexByname(lay: Integer; const pname: WideString): Integer;
function OneParam(lay: Integer; Num: Integer; par: Integer): OleVariant;
function GetLayName(lay: Integer): WideString;
function Get_LocalPropValue(lay: Integer; ndx: Integer): WideString;
procedure Set_LocalPropValue(lay: Integer; ndx: Integer; const Value: WideString);
function GetLocalPropCount(lay: Integer): Integer;
function GetLocalPropAlias(lay: Integer; index: Integer): WideString;
function GetLocalPropName(lay: Integer; index: Integer): WideString;
function GetLocalPropIndexByName(lay: Integer; const AName: WideString): Integer;
procedure SetOneParam(lay: Integer; Num: Integer; par: Integer; val: OleVariant);
procedure Close;
procedure SetGlobalProperty(const PropName: WideString; PropValue: OleVariant);
function GetGlobalProperty(const PropName: WideString): OleVariant;
function RawData(Indx: Integer; ptndx: Integer): Double;
property LayCount: Integer;
property PropValue[index: Integer]: WideString;
property LocalPropValue[lay: Integer; ndx: Integer]: WideString;

```

Примеры для VB/VBA

` Пример задание констант

```

Const error_ObjUnichrom = 65535
Const error_StartAcquisition = error_ObjUnichrom - 1

```

` Процедура автоматического старта шаблона на выполнение

```

Sub Auto_Open()
On Error GoTo ErrorHandler ` Пример обработки исключительных ситуаций

```

```

    Init
    Run
    Done

```

```

ErrorHandler:

```

```

    Select Case Err
        Case error_ObjUnichrom: printmsg msg:="Объект ""Unichrom"" не установлен", msgclass:=mcError
        Case error_StartAcquisition: printmsg msg:="Измерение не стартует", msgclass:=mcError
        Case Else: printmsg msg:=Error(Err), msgclass:=mcError
    End Select
    ExitCode% = 1
    Done

```

```

End Sub

```

` Пример подключения к ЮниХром и его запуск, если последний не запущен

```

Private Sub CreateUnichrom()
On Error GoTo ErrorHandler
    printmsg msg:="Загружается Unichrom", msgclass:=mcMessage
    Set unichrom = CreateObject("Unichrom.App"): unichrom.MainWndState wsMinimized
    printmsg msg:="Загрузка Unichrom завершена", msgclass:=mcMessage
    Exit Sub
ErrorHandler:
    Error error_ObjUnichrom
End Sub

```

` Пример получения указателя на спектр

```
Private Function LoadSpectrum(FileName$) As Object
    printmsg msg:="Загружается спектр " & FileName$, msgclass:=mcMessage
    With unichrom: Set LoadSpectrum = .OpenSpec(FileName$): .TileWindows: End With
    printmsg msg:="Загрузка спектра завершена", msgclass:=mcMessage
End Function
```

` Пример временной задержки выполнения программы (на всякий случай)

```
Private Sub Delay(secondcount%)
    newHour% = Int(secondcount% / 3600)
    newMinute% = Int(secondcount% / 60): newMinute% = newMinute% Mod 60
    newSecond% = secondcount% Mod 60
    Application.Wait TimeSerial(Hour(Now()) + newHour%, Minute(Now()) + newMinute%, second(Now()) + newSecond%)
End Sub
```

` Пример управления состоянием спектра (измерение, ожидание и т.д.)

```
Private Sub Cycle()
    CycleCount% = CycleCount% + 1
    printmsg msg:="Стартовал " & CycleCount% & " цикл измерений", msgclass:=mcMessage
    MeasurementCount% = 0

    For Each spectrum In spectrums
        With spectrum
            MeasurementCount% = MeasurementCount% + 1
            printmsg msg:="Стартовало " & MeasurementCount% & " измерение", msgclass:=mcMessage

            SpectrumName$ = .getpropvalue(.getpropindexbyname("Name"))
            CurrentLayName$ = .getpropvalue(.getpropindexbyname("CurLayDesc"))
            TotalTimeAcquisition& = .getpropvalue(.getpropindexbyname("Xend")) * 60

            FreeChannel .getpropvalue(.getpropindexbyname("Channel"))

            printmsg msg:="Установка режима ""Ожидание Старта"" для спектра "" & _
            UCase(SpectrumName$) & "" в слое "" & UCase(CurrentLayName$) & """, msgclass:=mcMessage
            .Acquire asWait

            If .Acquire(asQuery) <> asWait Then
                Error error_WaitStart
            End If

            printmsg msg:="Режим ""Ожидание Старта"" корректно установлен", msgclass:=mcMessage

            printmsg msg:="Подготовьте прибор для ввода пробы, введите пробу и нажмите кнопку старта на блоке АЦП",
            msgclass:=mcInformation
            Do While .Acquire(asQuery) = asWait
                printmsg msg:="ОЖИДАЮ СТАРТ", msgclass:=mcProgress
                Delay 1
            Loop

            start& = Timer

            If .Acquire(asQuery) <> asAcquire Then
                Error error_StartAcquisition
            End If

            printmsg msg:="Выполняется измерение спектра "" & _
            UCase(SpectrumName$) & "" в слое "" & UCase(CurrentLayName$) & """, msgclass:=mcMessage
            Do While .Acquire(asQuery) = asAcquire
                currentTimer& = Timer
                printmsg msg:="ИДЁТ ИЗМЕРЕНИЕ " & TotalTimeAcquisition& + start& - currentTimer&, msgclass:=mcProgress,
                SelfActivate:=False
                Delay 1
            Loop

            finish& = Timer

            ErrorT1me& = 5
            If finish& + ErrorT1me& - start& < TotalTimeAcquisition& Then
                Error error_RunAcquisition
            End If

            Do While .Acquire(asQuery) <> asUsual: Loop
            printmsg msg:="Измерение корректно завершено", msgclass:=mcMessage

            printmsg msg:="Выполняется сценарий обработки измеренных данных", msgclass:=mcMessage
            Do While .StartScenario(ssQuery) = ssRun
                printmsg msg:="РАБОТАЕТ СЦЕНАРИЙ", msgclass:=mcProgress
                Delay 1
            Loop
            printmsg msg:="Обработка по сценарию корректно завершена", msgclass:=mcMessage
```

```

Calculation spectrum

printmsg msg:=MeasurementCount% & " измерение корректно завершено", msgclass:=mcMessage
End With
Next

printmsg msg:=CycleCount% & " цикл измерений корректно завершён", msgclass:=mcMessage
End Sub

Private Sub FreeChannel(DeviceChannel%)
printmsg msg:="Проверка канала №" & DeviceChannel%, msgclass:=mcMessage
With unichrom
For i = 0 To .getspeccount - 1
    Set spectrum = .getspec(i)
    With spectrum
    If DeviceChannel% = .getpropvalue(.getpropindexbyname("Channel")) And _
        (spectrum.Acquire(asQuery) <> asUsual) Then
        Do While spectrum.Acquire(asQuery) <> asUsual
            printmsg msg:="Канал ЗАНЯТ", msgclass:=mcProgress
            Delay 1
        Loop
    End If
    End With
Next i
End With
printmsg msg:="Канал свободен", msgclass:=mcMessage
End Sub

```

----- Получаем данные от ЮНИХРОМа ----- САМАЯ ДЛИТЕЛЬНАЯ ОПЕРАЦИЯ

```

n% = 0 ' Количество пиков без учета фиктивных пиков и пиков не для отчета
For i% = 0 To Spectrum.GetPeakCountInLay(sActiveSheet%) - 1
    F% = Spectrum.OneParam(sActiveSheet%, i%, mState) ' Получаем состояние пика
    If ((F% And stNoReport) <> 0) Or ((F% And stFictive) <> 0) Then GoTo GetData_Continue ' Если пик не для отчета или
    фиктивный, то пропускаем его
    n% = n% + 1
    ReDim Preserve Peaks(n%) ' Расширяем массив пиков до n%
    With Peaks(n%)
        .Order = n% ' Порядок выхода пика (Order) совпадает с индексом пика в массиве пока массив отсортирован по ордерам
        .Name = Spectrum.OneParam(sActiveSheet%, i%, mName)
        .Time = Spectrum.OneParam(sActiveSheet%, i%, mTime) ' От ЮниХрома получаем абсолютные времена удерживания
        .Index = Spectrum.OneParam(sActiveSheet%, i%, mIndex)
        .Group = Spectrum.OneParam(sActiveSheet%, i%, mGroup)
        .Mass = Spectrum.OneParam(sActiveSheet%, i%, mMass)
        .BoilPoint = Spectrum.OneParam(sActiveSheet%, i%, mBoilPoint)
        .Density = Spectrum.OneParam(sActiveSheet%, i%, mDensity)
        .Left = Spectrum.OneParam(sActiveSheet%, i%, mLeft)
        .Right = Spectrum.OneParam(sActiveSheet%, i%, mRight)
        .Area = Spectrum.OneParam(sActiveSheet%, i%, mArea)
        .Height = Spectrum.OneParam(sActiveSheet%, i%, mHeight)
        .Response = Spectrum.OneParam(sActiveSheet%, i%, mResfact)
        .VolumeConc = Spectrum.OneParam(sActiveSheet%, i%, mVolumeConc)
        .MassConc = Spectrum.OneParam(sActiveSheet%, i%, mMassConc)
        .MolarConc = Spectrum.OneParam(sActiveSheet%, i%, mMolarConc)
        .Titre = Spectrum.OneParam(sActiveSheet%, i%, mTitre)
        .Molarity = Spectrum.OneParam(sActiveSheet%, i%, mMolarity)
    End With
GetData_Continue:
Next i%
If n% = 0 Then Error error_PeakCount ' Проверка на наличие пиков в слое
sPeakCount% = n%

```

Алгоритмы быстрых сортировок и поиска (памятка)

***** СОРТИРОВКА компонентов ТЕКУЩЕЙ ТАБЛИЦЫ пиков по ПОРЯДКОВОМУ НОМЕРУ *****

```

Private Sub PQuickSortByOrder(l%, r%)
Dim PeakParams As TPeakInfo
Do
    i% = l%
    j% = r%
    n% = Peaks((l% + r%) \ 2).Order
    Do
        While Peaks(i%).Order < n%: i% = i% + 1: Wend
        While Peaks(j%).Order > n%: j% = j% - 1: Wend
        If i% <= j% Then
            PeakParams = Peaks(i%)
            Peaks(i%) = Peaks(j%)
            Peaks(j%) = PeakParams
            i% = i% + 1
            j% = j% - 1
        End If
    Loop Until i% > j%

```

```

        If l% < j% Then Call PQuickSortByOrder(l%, j%)
        l% = i%
    Loop Until i% >= r%
End Sub

```

***** СОРТИРОВКА компонентов ТЕКУЩЕЙ ТАБЛИЦЫ пиков по ИМЕНАМ *****

' Используется функция PGetName, которая применима только к отсортированной по ордерам библиотеке при RefInd <> 0 Private

```

Sub PQuickSortByName(l%, r%)
Dim PeakParams As TPeakInfo
Do
    i% = l%
    j% = r%
    n$ = PGetName((l% + r%) \ 2)
    Do
        While StrComp(PGetName(i%), n$) < 0: i% = i% + 1: Wend
        While StrComp(PGetName(j%), n$) > 0: j% = j% - 1: Wend
        If i% <= j% Then
            PeakParams = Peaks(i%)
            Peaks(i%) = Peaks(j%)
            Peaks(j%) = PeakParams
            i% = i% + 1
            j% = j% - 1
        End If
    Loop Until i% > j%
    If l% < j% Then Call PQuickSortByName(l%, j%)
    l% = i%
Loop Until i% >= r%
End Sub

```

***** ПОИСК компонента БИБЛИОТЕКИ по ИНДЕКСУ удерживания *****

' Функция LFindByIndex применима только к отсортированному по ордерам списку пиков

```

Public Function LFindByIndex(BeginOrder%, EndOrder%, Index#, PeakOrder%) As Boolean
Find = False
l% = BeginOrder%
h% = EndOrder%
dImin# = 1600
While l% <= h%
    i% = (l% + h%) \ 2
    c# = DBTable(i%, lLinIndex) - Index#
    dI# = Abs(c#)
    If (dI# < dIndexLimit) And (dImin# > dI#) Then
        dImin# = dI#
        PeakOrder% = DBTable(i%, lOrder)
        Find = True
    End If
    If c# < 0 Then
        l% = i% + 1
    Else
        h% = i% - 1
    End If
    If c# = 0 Then l% = i%
Wend
LFindByIndex = Find
End Function

```