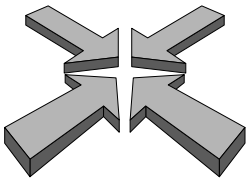

Ultramark™ OCX Development Guide

BIO-RAD

Bio-Rad Laboratories

Life Science Group

2000 Alfred Nobel Drive
Hercules, CA 94547

P/N 4000148 RevA

Catalog Number 170-9522

Bio-Rad Technical Services Department
Open Monday–Friday, 8:00 a.m. to 4:00 p.m., Pacific Standard Time.

Phone: (800) 424-6723, option 2, option 3
(510) 741-6576
Fax: (510) 741-5802
E-mail: LSG.TechServ.US@Bio-Rad.com (U.S.)
LSG.TechServ.Intl@Bio-Rad.com (International)

ISO 9001 Registered

Warranty

Bio-Rad Laboratories warrants that the Ultramark™ OCX software shall substantially conform, in all operational features, to Bio-Rad's current specifications as published in Bio-Rad's user and installation guides and that, when properly installed, it will be free of material defects which affect system performance.

The Purchaser must notify Bio-Rad in writing, within 30 days of delivery of the software (not including delivery of any subsequent modifications to the software), of any defect. If the software is found to be defective by Bio-Rad, Bio-Rad's sole obligation under this warranty is to remedy the defect in a manner consistent with Bio-Rad's regular business practices. For a defect, which adversely affects the performance of the software, Bio-Rad shall use its best efforts to cure such defect as soon as reasonably practicable after receipt of Purchaser's notice. For minor defects, Bio-Rad shall use its best efforts to correct such minor defects in the next release of its software. If, however, Bio-Rad is unable to cure a major defect within 90 days of receipt of Purchaser's notice, Purchaser shall have the option to cancel this agreement, whereupon Bio-Rad shall refund only the software fees paid.

The warranties set forth in this agreement are in lieu of all other representations and warranties, expressed or implied, including warranties of merchantability and fitness for a particular purpose and any other statutory or common-law warranty. Bio-Rad on its own behalf expressly disclaims and excludes any and all such other representations and warranties. Liability of Bio-Rad to Purchaser, if any, for breach of warranty, or any other claim relating to this agreement, shall be limited to the total amount of software fees paid by purchaser to Bio-Rad. In no event shall Bio-Rad be liable for incidental or consequential damages, loss of business or profits, special or indirect damages of any nature whatsoever. No amendment, waiver, or other alteration of the warranties in this agreement may be made except by mutual agreement in writing.

Purchaser agrees that Bio-Rad's liability arising out of contract, negligence, strict liability in tort or warranty shall not exceed the amount of software license fees paid by Purchaser.

A multi-use license may be purchased to allow the software to be used on more than one computer owned by the purchaser, including a shared disk system.

Table of Contents

1. Introduction

The Ultramark OCX Development Kit is designed to allow developers to integrate the Ultramark into their automation systems. This kit provides system integrators with the means to easily integrate the Ultramark into their Visual Basic and/or Visual C++ applications through the Ultramark ActiveX (OCX) developed by Bio-Rad Laboratories.

The Ultramark OCX Development Kit contains an ActiveX (OCX) control, a Development Guide (this document), and sample code. Using the kit, the developer writes his own application and designs a unique interface to integrate the Ultramark with his existing systems.

The Ultramark OCX control provides an interface to find, communicate, and control the Ultramark. It runs on Windows 95, Windows 98, and Windows NT systems and is compatible with Visual Basic and Visual C++. Other development platforms that support ActiveX (OCX) controls—such as Delphi and Visual Java—should also be compatible; however, compatibility is not guaranteed for these platforms. The OCX interface has been designed to access all of the Ultramark's capabilities, support nonblocking plate and imaging reads, and provide error notification and recovery.

This guide is divided into the following sections.

- Section 1: Introduction
- Section 2: Installing the Development Kit
- Section 3: Preparing the Ultramark
- Section 4: Ultramark OCX Control
- Section 5: Ultramark OCX Event Handling
- Section 6: Ultramark OCX Error Handling
- Section 7: Ultramark OCX Error Codes
- Section 8: Sample Code

Distributed Files

The Ultramark OCX Development Kit contains the following:

- Ultramark OCX Development Guide (this document)
- Ultramark OCX Development Kit disks, containing:
  - UltramarkOCX.OCX
  - UltramarkOCX.TBL
  - UltramarkOCX.OCX Documentation
  - Visual C++ Sample Code
  - Visual Basic Sample Code

2. Installing the Development Kit

Locate the diskettes contained in the Ultramark OCX Development Kit. Insert the Setup disk (Disk 1) into the floppy disk drive on your computer. On your Windows taskbar, click the *Start* button, then select *Run*. Type **a:\setup** in the field, and click on *OK*.

The installer will prompt you to insert the remaining disks. The default program directory is C:\Program Files\Bio-Rad\Ultramark OCX Development Kit. You can select a different directory when prompted to do so by the installer.

The installer will create a program group containing shortcuts to this development guide (Ultramark OCX Development Guide), a MFC sample application (TestUltraOCX), and ReadMe notes. The installer also copies sample code for Visual C++ 6.0 and Visual Basic 6.0 into the installation directory. The Ultramark OCX is also installed and registered.

3. Preparing the Ultramark

A SCSI-2 cable and card (cat. no. 170-9521) are required to connect the Ultramark to the host PC. Make sure all devices are turned off before making or changing cable connections. See the hardware manual for details.

The SCSI II port is located on the back of the instrument. There are two connectors, which allow the user to connect more than one SCSI instrument in a chain. If the Ultramark is the last instrument in the chain, make sure that the termination switch (next to the SCSI connectors) is in the “ON” position. If the Ultramark is an intermediate unit in the chain, make sure that the termination switch is in the “OFF” position. If the Ultramark is the only unit connected to the SCSI cable, make sure that the switch is in the “ON” position.

There is a small dial next to the SCSI connectors on the back of the unit. This dial selects the SCSI ID number for the unit at boot-up. Be sure to select an ID number that is not being used by any other SCSI device in the chain.

After connecting the Ultramark to the host PC, reboot your system by first turning on the Ultramark then the host PC. Make sure WINASPI.DLL is installed and compatible with your system’s SCSI card. A 29xx Series Adaptec SCSI card is recommended. To test the connection and SCSI communications, run the sample application (TestUltraOCX.exe) installed with the Ultramark OCX Development Kit.

4. Ultramark OCX Control

The Ultramark ActiveX (OCX) control provides complete operational control over the functional behavior of the Ultramark. The Ultramark command set is completely encapsulated in a set of control methods and events easily called by the developer. Operational control is achieved by calling these methods and responding to raised errors and fired events.

Once the UltramarkOCX control is added to a project, all device functionality is immediately available through the control's methods. All SCSI communications between the Ultramark and host computer are handled internally within the control. The only SCSI requirement is that the WINASPI.DLL is installed and is compatible with your system's SCSI card. It is recommended that a 29xx Series Adaptec SCSI card be used.

To communicate with the Ultramark, you first establish a connection to the device through the OpenUltramark or OpenUltramarkEx methods. OpenUltramark opens a connection to the first Ultramark found on the SCSI bus, while OpenUltramarkEx opens a connection to the Ultramark at the specified SCSI address (busId, targetId, and lunId). After communication is established, all of the query and control methods are accessible. When you are finished, call CloseUltramark to release the connection. The FindAttachedUltramarks and QueryAttachedUltramarkInfo methods are often used in conjunction with the OpenUltramarkEx when more than one Ultramark is attached to a system. These methods find and identify all the Ultramarks on each SCSI bus.

Control Methods

Method	Description
FindAttachedUltramarks	Gets the number of found Ultramarks.
QueryAttachedUltramarkInfo	Gets the SCSI busId, targetId, and lunId for the specified Ultramark.
OpenUltramark	Opens a port to the first Ultramark found.
OpenUltramarkEx	Opens a port to the Ultramark at the specified SCSI location.
CloseUltramark	Closes the port to the open Ultramark.
GetSCSIInfo	Gets the SCSI busId, targetId, and lunId of the opened Ultramark.
QueryInstrumentName	Gets the Ultramark's name string.
QueryFirmwareVersion	Gets the Ultramark's firmware version string.
QueryHardwareVersion	Gets the Ultramark's hardware version string.

Method	Description
QueryFirmwareVersionNumbers	Gets the Ultramark's firmware major and minor version numbers.
QueryHardwareVersionNumbers	Gets the Ultramark's hardware major and minor version numbers.
QueryNumberFilters	Gets the number of filters supported by the Ultramark.
QueryFilterName	Gets the filter name for the specified filter.
GetFilterMaxNameLength	Gets the maximum filter name length.
SetFilterName	Sets the filter name for the specified filter.
QueryNumberPlateSizes	Gets the number of supported plate sizes.
QueryPlateSizeInfo	Gets the plate size string and dimensions for the specified plate size.
QueryNumberMixingSpeeds	Gets the number of mixing speeds supported by the Ultramark.
QueryMaxMixTime	Gets the maximum mixing time.
QueryMixSpeedName	Gets the mixing speed name of the specified mixing speed.
QueryIncubatorSupported	Returns a non-zero value if the incubator control is supported; otherwise 0.
QueryIncubatorTempRange	Gets the incubator's temperature range in tenths of a degree Celsius.
QueryIncubatorStatus	Gets the incubator's On/Off state, current temperature, and setpoint.
QueryNumberResolutions	Gets the number of supported scanning resolutions.
QueryResolution	Gets the scanning resolution in spatial units.
SelectResolution	Selects the scanning resolution for the next imaging.
GetSpatialUnitsStr	Gets the string representation of the Ultramark's spatial units (millimeters, pixels, micrometers, centimeters, or inches).
GetSpatialUnits	Gets the Ultramark's spatial units.
QueryReadTime	Gets the estimated reading time for the specified plate.
QueryImagingReadTime	Gets the estimated reading time for imaging an entire plate.

Method	Description
QueryODTableInfo	Gets OD table information.
QueryODTable	Gets OD table entries.
QueryMaximumImageArea	Gets the maximum imaging area that can be scanned.
QueryMaximumImageSize	Gets the maximum image size that can be scanned.
SelectScanArea	Selects the imaging area to be scanned.
QueryImageArea	Gets the imaging area that will be scanned.
QueryImageSize	Gets the image size that will be scanned.
SingleWavelengthReadPlate	Starts a single wavelength plate read.
DualWavelengthReadPlate	Starts a dual wavelength plate read.
ImagingReadPlate	Starts an imaging plate read.
IsReading	Returns a non-zero value if a reading is in progress; otherwise 0.
AbortRead	Stops a read.
GetNumberColsRead	Gets the number of columns read.
GetNumberRowsRead	Gets the number of rows read.
GetValueAt	Gets the raw read data value of the point specified by a dataIndex.
GetValueAtRowCol	Gets the raw read data value of the point specified by a row and col.
GetODValueAt	Gets the OD value of the read point specified by a dataIndex.
GetODValueAtRowCol	Gets the OD value of the read point specified by a row and col.
GetFilterNumberMemory	Gets the number of the filter currently in the filter arm.
SetFilterNumberMemory	Sets the number of the filter currently in the filter arm.
ErrorReset	Resets Ultramark error conditions.
OpenDoor	Opens and extends the plate carrier.
CloseDoor	Closes and retracts the plate carrier.

Method	Description
GetOCXVersionNumber	Gets the OCX control version number.
GetOCXMajorRevisionNumber	Gets the OCX control major revision number.
GetOCXMinorRevisionNumber	Gets the OCX control minor revision number.
GetOCXBuildNumber	Gets the OCX control build number.

Control Events

Event	Description
ReadComplete	The read completed without error.
ReadCanceled	The read was canceled.
ReadError	The read was terminated due to an error.

FindAttachedUltramarks Method

short FindAttachedUltramarks ();

Return Value

The number of Ultramarks found on the SCSI buses.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

WINASPI installed.

Remarks

This method searches the SCSI buses for Ultramarks and returns the number of Ultramarks found. To find an Ultramark, the instrument must be connected to the PC and powered on. WINASPI must also be installed on the PC. **Note:** Windows 95 and Windows 98 come with WINASPI. WINASPI must be installed on computers using Windows NT. After calling this method, use QueryAttachedUltramarkInfo to retrieve the SCSI busId, targetId, and lunId for each attached Ultramark.

C++ Example

```
short nFound;
short busId, targetId, lunId;

nFound = Ultra.FindAttachedUltramarks();

for (short i = 0; i < nFound; i++)
{
   Ultra.QueryAttachedUltramarkInfo(i,&busId,&targetId,&lunId);
}
```

VB Example

```
Dim i As Integer
Dim nFound As Integer
Dim busId As Integer
Dim targetId As Integer
Dim lunId As Integer

nFound = ultra.FindAttachedUltramarks()

For i = 0 To nFound - 1
   Call ultra.QueryAttachedUltramarkInfo(i, busId, targetId, lunId)
Next i
```

QueryAttachedUltramarkInfo Method

void QueryAttachedUltramarkInfo (short Index,
short* busId,
short* targetId,
short* lunId
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

Prior call to FindAttachedUltramarks ().

Remarks

This method retrieves the SCSI busId, targetId, and lunId for the specified Ultramark. The index is zero-based and should not exceed the number of Ultramarks found using the FindAttachedUltramarks method.

C++ Example

```
short nFound;
short busId, targetId, lunId;

nFound = Ultra.FindAttachedUltramarks();

for (short i = 0; i < nFound; i++)
{
  Ultra.QueryAttachedUltramarkInfo(i,&busId,&targetId,&lunId);
}
```

VB Example

```
Dim i As Integer
Dim nFound As Integer
Dim busId As Integer
Dim targetId As Integer
Dim lunId As Integer

nFound = ultra.FindAttachedUltramarks()

For i = 0 To nFound - 1
   Call ultra.QueryAttachedUltramarkInfo(i, busId, targetId, lunId)
Next i
```

OpenUltramark Method

void OpenUltramark ();

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the Error Handling section for details.

Prerequisites

WINASPI must be installed and a port to an Ultramark must **not** be open.

Remarks

This method attempts to open a port to the first Ultramark found on the SCSI buses. To find an Ultramark, the instrument must be connected to the PC and powered on. WINASPI must also be installed on the PC. **Note:** Windows 95 and Windows 98 come with WINASPI, but it must be installed on Windows NT machines. If an Ultramark is not found, an error will be thrown.

C++ Example

```
Ultra.OpenUltramark();
```

VB Example

```
Ultra.OpenUltramark
```

OpenUltramarkEx Method

void OpenUltramarkEx (short busId,
short targetId,
short lunId
);

Return Value
None.

Errors
This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites
WINASPI must be installed and a port to an Ultramark must **not** be open.

Remarks
This method attempts to open a port to the Ultramark at the SCSI location specified by the busId, targetId, and lunId. To find an Ultramark, the instrument must be connected to the PC and powered on. WINASPI must also be installed on the PC. **Note:** Windows 95 and Windows 98 come with WINASPI, but it must be installed on Windows NT machines. If an Ultramark is not found, an error will be thrown.

C++ Example
```
Ultra.OpenUltramarkEx(busId, targetId, lunId);
```

VB Example
```
Dim busId As Integer
Dim targetId As Integer
Dim lunId As Integer

busId = 0
targetId = 2
lunId = 0
Call Ultra.OpenUltramarkEx(busId, targetId, lunId)
```

CloseUltramark Method

void CloseUltramark ();

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, and the OCX control must not be reading a plate.

Remarks

This method closes the port to the open Ultramark.

C++ Example

```
Ultra.CloseUltramark();
```

VB Example

```
Ultra.CloseUltramark
```

GetSCSIInfo Method

void GetSCSIInfo (short* busId,
short* targetId,
short* lunId
);

Return Value
None.

Errors
This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites
A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks
This method returns the SCSI busId, targetId, and lunId of the opened instrument (Ultramark). The values are returned in the arguments busId, targetId, and lunId.

C++ Example
```
short busId, targetId, lunId;
Ultra.GetSCSIInfo( &busId, &targetId, &lunId);
```

VB Example
```
DIM busId as Integer
DIM targetId as Integer
DIM lunId as Integer
Ultra.GetSCSIInfo( busId, targetId, lunId)
```

QueryInstrumentName Method

BSTR QueryInstrumentName ();

Return Value

The Ultramark's name string.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the name of the instrument.

C++ Example

```
CString  instrumentName;

instrumentName = Ultra.QueryInstrumentName();
```

VB Example

```
Dim instrumentName As String
instrumentName = Ultra.QueryInstrumentName()
```

QueryFirmwareVersion Method

BSTR QueryFirmwareVersion ();

Return Value

The instrument’s firmware version string.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the instrument’s firmware version string.

C++ Example

```
CString  firmwareVersion;

firmwareVersion = Ultra.QueryFirmwareVersion();
```

VB Example

```
Dim firmwareVersion As String
firmwareVersion = Ultra.QueryFirmwareVersion()
```

QueryHardwareVersion Method

BSTR QueryHardwareVersion ();

Return Value

The instrument's hardware version string.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the instrument's hardware version string.

C++ Example

```
CString  hardwareVersion;

hardwareVersion = Ultra.QueryFirmwareVersion();
```

VB Example

```
Dim hardwareVersion As String
hardwareVersion = Ultra.QueryFirmwareVersion()
```

QueryFirmwareVersionNumbers Method

void QueryFirmwareVersionNumbers (long * major,
long * minor
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the instrument's major and minor version numbers in the method arguments `major` and `minor`, respectively.

C++ Example

```
long major, minor;

Ultra.QueryFirmwareVersionNumbers(&major, &minor);
```

VB Example

```
Dim major As Long
Dim minor As Long
Call Ultra.QueryFirmwareVersionNumbers(major, minor)
```

QueryHardwareVersionNumbers Method

void QueryHardwareVersionNumbers (long * major,
long * minor
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the instrument's major and minor version numbers in the method arguments `major` and `minor`, respectively.

C++ Example

```
long major, minor;

Ultra.QueryHardwareVersionNumbers(&major, &minor);
```

VB Example

```
Dim major As Long
Dim minor As Long
Call Ultra.QueryHardwareVersionNumbers(major, minor)
```

QueryNumberFilters Method

short QueryNumberFilters ();

Return Value

The number of filters supported by the Ultramark.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the number of filters supported by the Ultramark.

C++ Example

```
short nFilters = Ultra.QueryNumberFilters ();
```

VB Example

```
Dim nFilters As Integer
nFilters = Ultra.QueryNumberFilters ()
```

QueryFilterName Method

BSTR QueryFilterName (short filterIndex);

Return Value

The filter name string for the filter specified by filterIndex. The filter index is a zero-based index.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the filter name for the specified filter. The filter index is a zero-based index.

C++ Example

```
short    filterIndex = 0;
CString  filterName;

// Gets the filter name for the filter at position 1
filterName = Ultra.QueryFilterName(filterIndex);
```

VB Example

```
Dim filterIndex As Integer
Dim filterName As String

filterIndex = 0

‘ Gets the filter name for the filter at position 1
filterName = Ultra.QueryFilterName (filterIndex)
```

GetFilterMaxNameLength Method

short GetFilterMaxNameLength ();

Return Value

The maximum filter name length that can be set using the SetFilterName method. This length does not include the terminating null character.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the maximum filter name length that can be held in the Ultramark. The length does not include the terminating null character. This limit is imposed by the SetFilterName method.

C++ Example

```
short maxFilterNameLength = Ultra.GetFilterMaxNameLength ();
```

VB Example

```
Dim maxFilterNameLength As Integer
maxFilterNameLength = Ultra.GetFilterMaxNameLength ()
```

SetFilterName Method

void SetFilterName (short filterIndex,
LPCTSTR filterName
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method sets the filter name for the specified filter. The filter index is a zero-based index.

C++ Example

```
Ultra.SetFilterName (0,  “260nm”);
```

VB Example

```
Call Ultra.SetFilterName (0,  “260nm”)
```

QueryNumberPlateSizes Method

short QueryNumberPlateSizes ();

Return Value

The number of plate sizes supported for single point reads.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the number of plate sizes supported for single point reads. Single point reads are nonimaging reads that take a single reading in the center of each well for the specified plate. See QueryPlateSizeInfo and SelectPlateSize method descriptions for more plate size information and its usage.

C++ Example

```
short numberPlateSizes;
numberPlateSizes = Ultra.QueryNumberPlateSizes ();
```

VB Example

```
Dim numberPlateSizes As Integer
numberPlateSizes = Ultra.QueryNumberPlateSizes()
```

QueryPlateSizeInfo Method

BSTR QueryPlateSizeInfo (short plateSizeIndex,
long* numberRows,
long* numberColumns
);

Return Value

The plate size string for the specified plate size. The index is zero-based. The plate dimensions are returned in the method arguments numberRows and numberColumns.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method gets the plate size string and dimensions for the specified plate size. The index is zero-based.

C++ Example

```
CString plateName;
long  numberColumns, numberRows
plateName = Ultra.QueryPlateSizeInfo(0, &numberRows, &numberColumns);
```

VB Example

```
DIM numberRows As Long
DIM numberColumns As Long
DIM plateName As String

‘ Query plate size info for plate type index 0
plateName = Ultra.QueryPlateSizeInfo(     0, numberRows, numberColumns)
```

QueryNumberMixingSpeeds Method

short QueryNumberMixingSpeeds ();

Return Value

The number of supported mixing speeds.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the number of mixing speeds supported by the instrument. See QueryMixingSpeedName and QueryMaxMixTime method descriptions for mixing capability information.

C++ Example

```
short numberMixingSpeeds;
numberMixingSpeeds  = Ultra.QueryNumberMixingSpeeds();
```

VB Example

```
Dim numberMixingSpeeds As Integer
numberMixingSpeeds = Ultra.QueryNumberMixingSpeeds()
```

QueryMaxMixTime Method

short QueryMaxMixTime ();

Return Value

The maximum supported mixing time in seconds.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the maximum mixing time supported by the instrument in seconds.

C++ Example

```
short maxMixingTimeSecs;
maxMixingTimeSecs = Ultra.QueryMaxMixTime();
```

VB Example

```
Dim maxMixingTimeSecs As Long
maxMixingTimeSecs = Ultra.QueryMaxMixTime()
```

QueryMixSpeedName Method

BSTR QueryMixSpeedName (short mixingIndex);

Return Value

The name of the mixing speed specified by the mixing speed index. The mixing speed index is zero-based.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the name of the mixing speed specified by the mixing speed index.

C++ Example

```
short  mixingSpeedIndex = 0;
CString mixingSpeedName;

mixingSpeedName = Ultra.QueryMixSpeedName(mixingSpeedIndex);
```

VB Example

```
Dim mixingSpeedIndex As Integer
Dim mixingSpeedName As String
mixingSpeedIndex = 0
mixingSpeedName = Ultra.QueryMixSpeedName(mixingSpeedIndex)
```

QueryIncubatorSupported Method

short QueryIncubatorSupported ();

Return Value

Non-zero returned if incubator control is supported; otherwise 0.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns a non-zero value if incubator control is supported; otherwise, 0 is returned.

C++ Example

```
BOOL bIncubatorSupport;
bIncubatorSupport = Ultra.QueryIncubatorSupported();
```

VB Example

```
Dim bIncubatorSupport As Boolean
bIncubatorSupport = Ultra.QueryIncubatorSupported()
```

QueryIncubatorTempRange Method

void QueryIncubatorTempRange(long* MinTemp,
long* MaxTemp
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the incubator's temperature range in the method arguments MinTemp and MaxTemp. The temperature values are in tenths of a degree Celsius.

C++ Example

```
long MinTemp, MaxTemp;
Ultra.QueryIncubatorTempRange(&MinTemp, &MaxTemp);
```

VB Example

```
DIM MinTemp As Long
DIM MaxTemp As Long

Call Ultra.QueryIncubatorTempRange(MinTemp, MaxTemp)
```

QueryIncubatorStatus Method

void QueryIncubatorStatus (short* bOn,
long* currentTemp
long* setPoint
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the incubator On/Off state, current temperature, and setpoint in bOn, currentTemp, and setPoint, respectively. The temperature values are in tenths of a degree Celsius.

C++ Example

```
BOOL bOn;
long currentTemp, setPoint;
Ultra.QueryIncubatorStatus(&bOn, &currentTemp, &setPoint);
```

VB Example

```
DIM bOn as Integer
DIM currentTemp as Long
DIM setPoint as Long

Call Ultra.QueryIncubatorStatus(bOn, currentTemp, setPoint)
```

SetIncubatorStatus Method

void SetIncubatorStatus (short bOn,
long setPoint
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method sets the incubator on/off state and incubator setpoint. These settings take effect immediately. If bOn is FALSE, the setPoint argument is ignored, so the setpoint won't change. The setPoint is in tenths of a degree Celsius.

C++ Example

```
BOOL bOn = TRUE;
long setPoint = 310; // 31.0 degrees Celsius
Ultra.SetIncubatorStatus(bOn, setPoint);
```

VB Example

```
Dim bOn As Integer
Dim setPoint As Long

bOn = True
setPoint = 310

‘ Incubator on at 31.0 degrees Celsius
Call Ultra.SetIncubatorStatus(bOn, setPoint)
```

QueryNumberResolutions Method

short QueryNumberResolutions ();

Return Value

The number of supported scanning resolutions.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the number of supported scanning resolutions. Resolution selection only applies to imaging reads. It doesn't apply to single point reads.

C++ Example

```
short nResolutions = Ultra.QueryNumberResolutions();
```

VB Example

```
Dim nResolutions As Integer
nResolutions = Ultra.QueryNumberResolutions()
```

QueryResolution Method

long QueryResolution (short resolutionIndex);

Return Value

The scanning resolution in spatial units specified by the resolutionIndex. The resolutionIndex is zero-based.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the scanning resolution in spatial units specified by the resolutionIndex. The GetSpatialUnitsStr and GetSpatialUnits methods can be used to get the spatial units for the instrument. See GetSpatialUnitsStr and GetSpatialUnits for details.

C++ Example

```
short resIndex = 0;
long  resolution = Ultra.QueryResolution(resIndex);
```

VB Example

```
Dim resIndex As Integer
Dim resolution As Long
resIndex = 0
resolution = ultra.QueryResolution(resIndex)
```

SelectResolution Method

void SelectResolution (long resolution);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method selects the scanning resolution for the next imaging read. The return value from QueryResolution should be used as the resolution argument.

C++ Example

```
Ultra.SelectResolution(Ultra.QueryResolution(0));
```

VB Example

```
Dim resIndex As Integer
resIndex = 0

Call Ultra.SelectResolution(Ultra.QueryResolution(resIndex))
```

GetSpatialUnitsStr Method

BSTR GetSpatialUnitsStr ();

Return Value

The string representation of the instrument's spatial units (millimeters, pixels, micrometers, centimeters, or inches).

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the string representation of the instrument's spatial units (millimeters, pixels, micrometers, centimeters, or inches). See GetSpatialUnits for retrieving the enumeration value of the instrument's spatial units.

C++ Example

```
CString spatialUnitsStr;
spatialUnitsStr  = Ultra.GetSpatialUnitsStr();
```

VB Example

```
Dim spatialUnitsStr As String
spatialUnitsStr  = Ultra.GetSpatialUnitsStr()
```

GetSpatialUnits Method

short GetSpatialUnits ();

Return Value

The instruments spatial units
0 — Millimeters
1 — Pixels
2 — Micrometers
3 — Centimeters
4 — Inches

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the instrument's spatial units.

C++ Example

```
short spatialUnits;
spatialUnits = Ultra.GetSpatialUnits();
```

VB Example

```
Dim spatialUnits As Integer
spatialUnits = Ultra.GetSpatialUnits()
```

QueryReadTime Method

short QueryReadTime (short plateSizeIndex, short bDualWavelength);

Return Value

An estimate of how long it will take to read the specified plate type in single wavelength and/or dual wavelength mode. The return value is in seconds.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns an estimate of the number of seconds it will take to read the specified plate. This is an estimate and it does not include mixing time.

C++ Example

```
short readTimeSecs;     // Single wavelength read of plate type 1
readTimeSecs = Ultra.QueryReadTime( 0, FALSE);
```

VB Example

```
Dim plateTypeIndex As Integer
Dim readTimeSecs As Integer
Dim bDualWavelengthRead As Integer

plateTypeIndex = 0
bDualWavelengthRead = False
readTimeSecs = Ultra.QueryReadTime(plateTypeIndex, bDualWavelengthRead)
```

QueryImagingReadTime Method

short QueryImagingReadTime
(short resolutionIndex,
short bDualWavelength
);

Return Value

An estimate of how long it will take to image the entire plate in single wavelength and/or dual wavelength mode. The return value is in seconds.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns an estimate of the number of seconds it will take to image the entire plate. This is an estimate and it does not include mixing time.

C++ Example

```
short readTimeSecs;
readTimeSecs = Ultra.QueryImagingReadTime( 0, FALSE);
```

VB Example

```
Dim resIndex As Integer
Dim readTimeSecs As Integer
Dim bDualWavelength As Integer

resIndex = 0
bDualWavelength = False
readTimeSecs = Ultra.QueryImagingReadTime(resIndex, bDualWavelength)
```

QueryODTableInfo Method

void QueryODTableInfo (long* numEntries,
long* divider
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the number of OD table entries and its divider in numEntries and divider, respectively. This information is necessary for getting and interpreting the OD table via the QueryODTable method. The ReadPlateData and GetValueAt methods return raw data values, not OD values, so the OD table can be used to convert from raw data values to OD values. The conversion is as follows:

OD = ODTable[RawDataValue] / divider.

C++ Example

```
long numEntries, divider;
Ultra.QueryODTableInfo( &numEntries, &divider);
```

VB Example

```
DIM numEntries As Long
DIM divider As Long
Call Ultra.QueryODTableInfo( numEntries, divider)
```

QueryODTable Method

void QueryODTable (long numEntries,
long* pODTable
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the first numEntries of the OD table in pODTable. The caller must ensure that pODTable can hold the requested number of OD table entries.

C++ Example

```
long numEntries, divider;
Ultra.QueryODTableInfo( &numEntries, &divider);

long *ODTable = new long[numEntries];
Ultra.QueryODTable( numEntries, ODTable);
delete[] ODTable;
```

VB Example

```
’ Retrieve OD table info and table
Dim NumEntries As Long
Dim Divider As Long
Call ultra.QueryODTableInfo(NumEntries, Divider)

Dim ODTable() As Long
ReDim ODTable(NumEntries) ’ Allocate enough space for the table
Call ultra.QueryODTable(NumEntries, ODTable(0))
```

QueryMaximumImageArea Method

void QueryMaximumImageArea (long* left,
long* top,
long* right,
long* bottom
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the maximum imaging area that can be scanned. Do not assume the imaging area is based at (0,0), because it is not. The dimensions are in spatial units. See the description for GetSpatialUnits for details on determining spatial units.

C++ Example

```
long left, top, right, bottom;
Ultra.QueryMaximumImageArea(&left, &top, &right, &bottom);
```

VB Example

```
DIM left As Long
DIM top As Long
DIM right As Long
DIM bottom As Long

Call Ultra.QueryMaximumImageArea(left, top, right, bottom)
```

QueryMaximumImageSize Method

void QueryMaximumImageSize(long* cx,
long* cy
);

Return Value

None

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the maximum image size in spatial units that can be scanned. The maximum width is returned in cx and the maximum height is returned in cy.

C++ Example

```
long cx, cy;
Ultra.QueryMaximumImageSize(&cx, &cy);
```

VB Example

```
DIM cx As Long
DIM cy As Long

Call Ultra.QueryMaximumImageSize(cx, cy)
```

QueryImageArea Method

void QueryImageArea (long* left,
long* top,
long* right,
long* bottom
);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the imaging area that will be scanned. The SelectScanArea method requests an area to scan; however, the actual scanned area may be slightly larger to accommodate the selected scanning resolution. It is recommended that QueryImageArea be called after selecting the scan area to retrieve the actual scan area. The dimensions are in spatial units. See the description for GetSpatialUnits for details on determining spatial units.

C++ Example

```
long left, top, right, bottom;
Ultra.QueryImageArea(&left, &top, &right, &bottom);
```

VB Example

```
DIM left As Long
DIM top As Long
DIM right As Long
DIM bottom As Long

Call Ultra.QueryImageArea(left, top, right, bottom)
```

SelectScanArea Method

```
void SelectScanArea (long   left,
                     long   top,
                     long   right,
                     long   bottom
                    );
```

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method sets the imaging area to be scanned. This method is a request; the actual scanned area may be slightly larger to accommodate the selected scanning resolution or adjusted to follow pixel boundaries. It is recommended that QueryImageArea be called after selecting the scan area to retrieve the actual scan area. The minimum width and height of the imaging area are 1,000 spacial units. The imaging area must also be contained within the maximum imaging area returned from the QueryMaximumImageArea method. The dimensions are in spatial units. See the description for GetSpatialUnits for details on determining spatial units.

C++ Example

```
long left, top, right, bottom;
Ultra.QueryMaximumImageArea(&left, &top, &right, &bottom);
Ultra.SelectImageArea(left, top, right, bottom);
```

VB Example

```
DIM left As Long
DIM top As Long
DIM right As Long
DIM bottom As Long

Call Ultra.QueryMaximumImageArea(left, top, right, bottom)
Call Ultra.SelectImageArea(left, top, right, bottom)
```

QueryImageSize Method

void QueryImageSize (long* cx, long* cy);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the image size that will be scanned. The SelectScanArea method requests an area to scan; however, the actual scan area may be slightly larger to accommodate the selected scanning resolution. It is recommended that QueryImageSize be called after selecting the scan area to retrieve the actual scan size. The dimensions are in spatial units. See the description for GetSpatialUnits for details on determining spatial units.

C++ Example

```
long cx, cy;
Ultra.QueryImageSize(&cx, &cy);
```

VB Example

```
DIM cx As Long
DIM cy As Long

Call Ultra.QueryImageSize(cx, cy)
```

SingleWavelengthReadPlate Method

void SingleWavelengthReadPlate (short plateSizeIndex, short measurementFilterIndex, short mixingSpeedIndex, short mixingTimeSec);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, and the OCX control must not be reading a plate.

Remarks

This method starts a single wavelength plate read. The plateSizeIndex specifies the type of plate to read (96, 384, or 1536 wells), measurementFilterIndex specifies the measurement filter, mixingSpeedIndex specifies the mixing speed, and mixingTimeSec specifies the mixing time in seconds. Once a reading is started, it can be stopped using the AbortRead () method; otherwise a reading event will be fired on completion of the read. See **Reading Events** for details.

C++ Example

```
Ultra.SingleWavelengthReadPlate ( 0,/* plateSizeIndex */
                                  0,/* measurementFilterIndex */
                                  0,/* mixingSpeedIndex */
                                  0 /* mixingTimeSec *
                                );
```

VB Example

```
Dim PlateTypeIndex As Integer
Dim MeasFilterIndex As Integer
Dim MixingSpeedIndex As Integer
Dim MixingTimeSec As Long
MixingTimeSec = 2
PlateTypeIndex = 0
MeasFilterIndex = 0
MixingSpeedIndex = 0
MixingTimeSec = 2

Call ultra.SingleWavelengthReadPlate(PlateTypeIndex, MeasFilterIndex, _
                                     MixingSpeedIndex, MixingTimeSec )
```

DualWavelengthReadPlate Method

void DualWavelengthReadPlate (short plateSizeIndex, short measurementFilterIndex, short referenceFilterIndex, short mixingSpeedIndex, short mixingTimeSec);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, and the OCX control must not be reading a plate.

Remarks

This method starts a dual wavelength plate read. The plateSizeIndex specifies the type of plate to read (96, 384, or 1536 wells), measurementFilterIndex specifies the measurement filter, referenceFilterIndex specifies the reference filter, mixingSpeedIndex specifies the mixing speed, and mixingTimeSec specifies the mixing time in seconds. Once a reading is started, it can be stopped using the AbortRead () method; otherwise a reading event will be fired on completion of the read. See **Reading Events** for details.

C++ Example

```
UltramarkCtrl.DualWavelengthReadPlate
   ( 0 /* plateTypeIndex */,
     0 /* MeasFilterIndex */, 1 /* refFilterIndex */,
     0 /* MixingSpeedIndex */,          0 /* MixingTimeSec */ );
```

VB Example

```
Dim PlateTypeIndex As Integer
Dim MeasFilterIndex As Integer
Dim RefFilterIndex As Integer
Dim MixingSpeedIndex As Integer
Dim MixingTimeSec As Long
MixingTimeSec = 2
PlateTypeIndex = 0
MeasFilterIndex = 0
MixingSpeedIndex = 0
MixingTimeSec = 0

Call ultra.DualWavelengthReadPlate(PlateTypeIndex, MeasFilterIndex, _
   RefFilterIndex, MixingSpeedIndex, MixingTimeSec )
```

ImagingReadPlate Method

```
void ImagingReadPlate ( short  measFilterIndex,
                        short  mixingSpeedIndex,
                        short  mixingTimeSec,
                        short  resolutionIndex,
                        long   imageAreaLeft,
                        long   imageAreaTop,
                        long   imageAreaRight,
                        long   imageAreaBottom
                      );
```

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, and the OCX control must not be reading a plate.

Remarks

This method starts an imaging plate read. The measurementFilterIndex specifies the measurement filter, mixingSpeedIndex specifies the mixing speed, mixingTimeSec specifies the mixing time in seconds, and the imaging area is specified in spatial units. Once a reading is started it can be stopped using the AbortRead () method; otherwise a reading event will be fired on completion of the read. See **Reading Events** for details.

C++ Example

```
UltramarkCtrl.ImagingReadPlate ( measFilterIndex, mixingSpeedIndex,
                                 mixingTime, resIndex,
                                 left, top, right, bottom
                               );
```

VB Example

```
Dim left As Long
Dim top As Long
Dim right As Long
Dim bottom As Long

Call ultra.QueryMaximumImageArea(left, top, right, bottom)
Call ultra.ImagingReadPlate   ( measFilterIndex, mixingSpeedIndex, _
                                mixingTime, resIndex, _
                                left, top, right, bottom _
                              )
```

IsReading Method

BOOL IsReading ();

Return Value

A non-zero value is returned if a reading is in progress; otherwise, zero is returned.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method.

Remarks

This method returns a non-zero value if a reading is in progress; otherwise, 0 is returned.

C++ Example

```
BOOL bIsReading = UltramarkCtrl.IsReading();
```

VB Example

```
Dim bIsReading As Boolean
bIsReading = UltramarkCtrl.IsReading()
```

AbortRead Method

void AbortRead ();

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method sets an internal abort flag that causes the reading to stop the next time the flag is checked within the reading thread. This will cause the reading thread to stop and an asynchronous event to be fired. Termination is not immediate. See the ***Reading Events*** sections for more details.

C++ Example

```
Ultra.AbortRead();
```

VB Example

```
Ultra.AbortRead
```

GetNumberColsRead Method

long GetNumberColsRead ();

Return Value

The number of columns read.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, the OCX control must not be reading a plate, and a plate read must have been successfully completed.

Remarks

This method returns the number of columns read during the last plate reading. This is a value for both imaging and nonimaging plate reads.

C++ Example

```
long numCols = UltramarkCtrl.GetNumberColsRead ();
```

VB Example

```
Dim numCols As Long
numCols = UltramarkCtrl.GetNumberColsRead ()
```

GetNumberRowsRead Method

long GetNumberRowsRead ();

Return Value

The number of rows read.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, the OCX control must not be reading a plate, and a plate read must have been successfully completed.

Remarks

This method returns the number of rows read during the last plate reading. This is a value for both imaging and nonimaging plate reads.

C++ Example

```
long numRows = UltramarkCtrl.GetNumberRowsRead ();
```

VB Example

```
Dim numRows As Long
numRows = UltramarkCtrl.GetNumberRowsRead ()
```

GetValueAt Method

long GetValueAt (long dataIndex);

Return Value

The raw read data value of the point specified by dataIndex. The dataIndex is zero-based.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, the OCX control must not be reading a plate, and a plate read must have been successfully completed.

Remarks

This method returns the raw read data value of the point specified by dataIndex. The dataIndex is zero-based. The mapping from (Row, Column) to dataIndex is as follows:

dataIndex = (Row * nColumns) + Column;

Note: Row and Column are zero-based values.

C++ Example

```
short value = UltramarkCtrl.GetValueAt (0);
```

VB Example

```
Dim lPoints As Long
lPoints = ultra.GetNumberColsRead() * ultra.GetNumberRowsRead()

’ Retrieve data points
Dim rawValue As Integer
Dim index As Long

For index = 0 To lPoints - 1
    rawValue = ultra.GetValueAtRowCol(index)
Next index
```

GetValueAtRowCol Method

long GetODValueAtRowCol (long row, long col);

Return Value

The raw read value of the point specified by row and col; row and col are zero-based.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, the OCX control must not be reading a plate, and a plate read must have been successfully completed.

Remarks

This method returns the raw value of the read point specified by row and col. Row and col are zero-based indexes.

C++ Example

```
long row = 0, col =0;
long value = UltramarkCtrl.GetValueAtRowCol (row, col);
```

VB Example

```
Dim lCols As Long
Dim lRows As Long

lCols = ultra.GetNumberColsRead()
lRows = ultra.GetNumberRowsRead()

Dim rawValue As Integer

’ Retrieve data points
Dim row As Long
Dim col As Long
For row = 0 To lRows - 1
    For col = 0 To lCols - 1
        rawValue = ultra.GetValueAtRowCol(row, col)
    Next col
Next row
```

GetODValueAt Method

float GetODValueAt (long dataIndex);

Return Value

The read OD value of the point specified by dataIndex; dataIndex is zero-based.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, the OCX control must not be reading a plate, and a plate read must have been successfully completed.

Remarks

This method returns the OD value of the read point specified by dataIndex. The dataIndex is zero-based. The mapping from (Row, Column) to dataIndex is as follows:

dataIndex = (Row * nColumns) + Column;

Note: Row and Column are zero-based values.

C++ Example

```
float value = UltramarkCtrl.GetODValueAt (0);
```

VB Example

```
Dim lPoints As Long
lPoints = ultra.GetNumberColsRead() * ultra.GetNumberRowsRead()

’ Retrieve data points
Dim OdValue As Double
Dim index As Long

For index = 0 To lPoints - 1
    rawValue = ultra.GetODValueAtRowCol(index)
Next index
```

GetODValueAtRowCol Method

float GetODValueAtRowCol (long row, long col);

Return Value

The read OD value of the point specified by row and col; row and col are zero-based.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened using the OpenUltramark or OpenUltramarkEx method, the OCX control must not be reading a plate, and a plate read must have been successfully completed.

Remarks

This method returns the OD value of the read point specified by row and col. Row and col are zero-based indexes.

C++ Example

```
float value = UltramarkCtrl.GetODValueAtRowCol (row, col);
```

VB Example

```
Dim lCols As Long
Dim lRows As Long

lCols = ultra.GetNumberColsRead()
lRows = ultra.GetNumberRowsRead()

Dim OdValue As Double

’ Retrieve data points
Dim row As Long
Dim col As Long
For row = 0 To lRows - 1
    For col = 0 To lCols - 1
        OdValue = ultra.GetODValueAtRowCol(row, col)
    Next col
Next row
```

GetFilterNumberMemory Method

short GetFilterNumberMemory ();

Return Value

The filter number currently in the filter arm. This is a filter number, not an index (0 = none; –1 = unknown; otherwise, a filter number is used (1 to nFilters).

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method returns the number of the filter currently in the filter arm. Filter numbers range from 1 to nFilters. 0 is returned for none, and –1 for unknown. This method is typically used during error recovery after a filter transfer error is detected. During recovery, the user must open the filter side panel door and visually determine and specify which filter is in the filter arm. After this, the SetFilterNumberMemory (filterNumber) method should be called with the user-supplied filter position, followed by a call to ResetError (). This procedure sets the proper filter number and clears the error condition.

C++ Example

```
short filterNumber;
filterNumber = Ultra.GetFilterNumberMemory();
```

VB Example

```
Dim filterNumber As Integer
filterNumber = Ultra.GetFilterNumberMemory()
```

SetFilterNumberMemory Method

void SetFilterNumberMemory (short filterNumber);

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method tells the instrument firmware what filter is currently in the filter arm. Valid filter numbers are from 0 to nFilters (0 = none). This method is typically used during error recovery after a filter transfer error is detected. During recovery, the user must open the filter side panel door and visually determine and specify which filter is in the filter arm. After this, the SetFilterNumberMemory (filterNumber) method should be called with the user supplied filter position, followed by a call to ResetError (). This procedure sets the proper filter number and clears the error condition.

C++ Example

```
Ultra.SetFilterNumberMemory(filterNumber);
```

VB Example

```
Dim filterNumber As Integer
filterNumber = 0

’ Set filter position to NONE (position 0)
Call Ultra.SetFilterNumberMemory(filterNumber)
```

ErrorReset Method

void ErrorReset ();

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method.

Remarks

This method is typically used during error recovery after a filter transfer error is detected. During recovery, the user must open the filter side panel door and visually determine and specify which filter is in the filter arm. After this, the SetFilterNumberMemory (filterNumber) method should be called with the user supplied filter position, followed by a call to ResetError (). This procedure sets the proper filter number and clears the error condition.

C++ Example

```
Ultra.ResetError();
```

VB Example

```
Ultra.ResetError
```

OpenDoor Method

void OpenDoor ();

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. MFC clients can catch these errors using a try/catch block:

```
Try
{
	…
}
catch ( ColeDispatchException *pException )
{
	…
}
```

VB clients can catch these errors using

TODO:

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method and the OCX control must not be reading a plate.

Remarks

Opens the door of the plate carrier and extends the plate carrier to its plate receiving position. If the plate carrier door is open and extended, this method immediately returns without error.

C++ Example

```
Ultra.OpenDoor();
```

VB Example

```
Ultra.OpenDoor
```

CloseDoor Method

void CloseDoor ();

Return Value

None.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

A port to the Ultramark must have been successfully opened by the OpenUltramark method and the OCX control must not be reading a plate.

Remarks

Closes the door of the plate carrier and retracts the plate carrier to its plate reading position. If the plate carrier door is retracted and closed, this method will home the plate carrier and returns without error.

C++ Example

```
Ultra.CloseDoor();
```

VB Example

```
Ultra.CloseDoor
```

GetOCXVersionNumber Method

short GetOCXVersionNumber ();

Return Value

The version number of the OCX control.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

None.

Remarks

This method returns the version number of the OCX control.

C++ Example

```
short versionNumber
versionNumber = Ultra.GetOCXVersionNumber ();
```

VB Example

```
Dim versionNumber As Integer
versionNumber = Ultra.GetOCXVersionNumber ()
```

GetOCXMajorRevisionNumber Method

short GetOCXMajorRevisionNumber ();

Return Value

The major revision number of the OCX control.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

None.

Remarks

This method returns the major revision number of the OCX control.

C++ Example

```
short majorRevisionNumber
majorRevisionNumber = Ultra.GetOCXMajorRevsionNumber ();
```

VB Example

```
Dim majorRevisionNumber As Integer
majorRevisionNumber = Ultra.GetOCXMajorRevsionNumber ()
```

GetOCXMinorRevisionNumber Method

short GetOCXMinorRevisionNumber ();

Return Value

The minor revision number of the OCX control.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

None.

Remarks

This method returns the major revision number of the OCX control.

C++ Example

```
short minorRevisionNumber;
minorRevisionNumber = Ultra.GetOCXMinorRevsionNumber ();
```

VB Example

```
Dim minorRevisionNumber As Integer
minorRevisionNumber = Ultra.GetOCXMinorRevisionNumber ()
```

GetOCXBuildNumber Method

short GetOCXBuildNumber ();

Return Value

The OCX build number.

Errors

This control calls ColeControl::ThrowError to signal the control user that an error has occurred. See the ***Error Handling*** section for details.

Prerequisites

None.

Remarks

This method returns the OCX build number.

C++ Example

```
short buildNumber;
buildNumber = Ultra.GetOCXBuildNumber ();
```

VB Example

```
Dim buildNumber As Integer
buildNumber = Ultra.GetOCXBuildNumber ()
```

5. Ultramark OCX Event Handling

The OCX notifies its client when a plate read has completed by firing an event. There are three different events that signal the completion of a read: the ***ReadComplete***, ***ReadCanceled***, and ***ReadError***. The ReadComplete event signals successful completion of the read. The ReadCanceled event is fired when the read is stopped using the AbortRead method or by a user pressing the ***Stop*** button on the Ultramark's front panel. The ReadError event is fired whenever an error occurs during a read. For this event, an error code and error description are passed along with the event.

Visual Basic Example:

```
' To handle the events raised by an event source, you declare a
' variable of the object's class using the WithEvents keyword.
Private WithEvents ultraEvents As UltramarkOCX

Dim ultra As UltramarkOCX     ' Reserve space for an UltramarkOCX object

Set ultra = New UltramarkOCX  ' Instantiate the UltramarkOCX object

Set ultraEvents = ultra ' Set the event object

' Event Handlers

' ReadComplete Event
Private Sub ultraEvents_ReadComplete()
…
End Sub

' ReadCanceled Event
Private Sub ultraEvents_ReadCanceled()
…
End Sub

' ReadError Event
Private Sub ultraEvents_ReadError(ByVal scode As Long, ByVal description As
String)
   MsgBox "Ultramark Read Error #" + Str(scode) + ", " + description
End Sub
```

Visual C++ MFC example:

```
// Declare an event sink map in CDlg header
DECLARE_EVENTSINK_MAP()

// Setup an event sink map in CDlg source. Id is child Id used to create
// the OCX. CDlg is the OCX parent window handling the events.
//
BEGIN_EVENTSINK_MAP(CDlg, CDialog)
ON_EVENT(CDlg, Id, 1 /* ReadComplete */, OnReadComplete, VTS_NONE)
ON_EVENT(CDlg, Id, 2 /* ReadCanceled */, OnReadCanceled, VTS_NONE)
```

```
ON_EVENT(CDlg, Id, 3 /* ReadError    */, OnReadError, VTS_I4 VTS_BSTR)
END_EVENTSINK_MAP()

// Add event handlers

// ReadComplete Event Handler
void CDlg::OnReadComplete()
{
…
}

// ReadCanceled Event Handler
void CDlg::OnReadCanceled()
{
…
}

// ReadError Event Handler
void CDlg::OnReadError(long scode, LPCTSTR description)
{
…
}
```

6. Ultramark OCX Error Handling

The OCX notifies its client of an error by raising an error. When calling methods, it should always be assumed that an error could occur. It is recommend that Visual Basic clients use *On error* handler and Visual C++ MFC client use *try/catch (COleDispatchException *pException)* blocks to handle error conditions.

Visual Basic Example:

```
On Error GoTo errorHandler

...

errorHandler:

’ Display error message
MsgBox "Ultramark Error #" + Str(Err.Number) + ", " + Err.description

’ Test error code to perform recovery
If (Err.Number = ???) Then

’ Do error recovery
End If
```

Visual C++ MFC example:

```
try
{
   ...
}
catch (COleDispatchException *pException) // Catch OCX exception
{
   // Optionally display error description
   pException->ReportError();
   // Optionally use error code to perform error recovery.
   switch (pException->m_wCode)
{
   ...
}
   // Must delete exception to avoid memory leak.
   pException->Delete();
}
```

7. Ultramark OCX Error Codes

The OCX notifies its client of an error by raising an error. The following is a list of custom errors that can be raised by the OCX control.

Error Code	Description
1000	Generic error
1001	Unknown error
1002	Invalid method argument
1003	Invalid method index
1100	SCSI communication error
1110	SCSI timeout
1200	WINASPI.DLL error
1210	WINASPI.DLL driver not found
1300	Scanner command error
1310	Invalid scanner command
1315	Invalid scanner command argument
1400	Scanner error
1415	Scanner filter transfer error
1450	Scanner already open error
1451	Scanner not open error
1452	Scanner busy error
1453	Scanner not found error

8. Sample Codes

The Ultramark™ OCX Development Kit includes Visual C++ 6.0 and Visual Basic 6.0 sample code to build applications that access and exercise the Ultramark™ OCX control. This sample code has been provided to help developers better understand the Ultramark™ OCX control and its capabilities and clarify any issues raised in this document. This sample code is automatically installed during installation of the development kit.

To build the Visual C++ sample application, open the workspace file TestUltraOCX.dsw in Visual C++ 6.0 and select the “Rebuild All” build option. This sample application exercises the capabilities of the OCX control and demonstrates method calling and event and error handling.

To run the Visual Basic sample application, open the Visual Basic project file TestUltramarkOCX.vbp and run it. The Visual Basic sample application has a much simpler user interface, but, like the Visual C++ sample application, it completely exercises the capabilities of the OCX control.