

ControMate^{OLE} Component
ActiveX Automation component for *ControMate*

Trademarks and Copyright

© 2004 Matrix Technologies Corp. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, in whole or in part, without the prior written permission of Matrix Technologies Corp. ControlMate^{OLE} is a trademark and Matrix Technologies Corp. and the Matrix logo are registered trademarks of Matrix Technologies Corp. The software product is protected by copyright laws and international treaty provisions. Therefore, you must treat the software product like any other copyrighted material except that you may either (a) make one copy of the software product solely for backup or archival purposes, or (b) install the SOFTWARE PRODUCT on a single computer provided you keep the original solely for backup or archival purposes. Third-party trademarked products are trademarks or registered trademarks of their respective companies in the United States and/or other countries.

Contents

Section	Page
1 Introduction	3
1.1 Build and Distribution	3
1.2 Development Environment.....	3
2 Component Usage	5
2.1 Component Reference	5
2.2 Sample Application.....	5
2.3 Class Definitions.....	6
2.3.1 Class Diagram.....	6
2.3.2 <i>Connect</i> Class.....	6
2.3.3 <i>CommandList</i> Class	8
2.3.4 <i>Fields</i> Class.....	8
2.3.5 <i>Field</i> Class	9
2.3.6 <i>FileInfo</i> Class	10
2.3.7 <i>MessageDialogue</i> Class	10
A Appendix – Example usage	13
A1 Opening a Sequence File.....	13
A2 Changing Field Values	13
A3 Sequence File Execution.....	14
A4 Message Boxes.....	14
B Appendix - Sequence File Command and Field Names	17
B1 <i>ControMate</i> Intrinsic Commands.....	17
B2 <i>SerialMate</i> Commands.....	18
B3 <i>PlateMate^{Plus}</i> Commands	19
C Appendix - Sample Application	21
C1 Sample Application Source Code	21

1 Introduction

ControMateOLE provides an application developer with an ActiveX automation interface for the *ControMate* device control application. The component will enable an application to instantiate an object which can load, modify and execute *ControMate* Sequence Files.

The calls made to the *ControMate* components do not involve the need to load the *ControMate* application into memory.

A very powerful and useful feature of the *ControMateOLE* component is the ability to dynamically change and set command field values within a sequence file. This allows the external integration of information such as source values (e.g. aspiration volumes) being extrapolated from databases.

1.1 Build and Distribution

The *ControMateOLE* component can be distributed as a required component within the client application. Any component dependencies will be included as part of the application build. The component cannot be distributed, on its own, without prior written consent of Matrix Technologies Corp or its distributors.

1.2 Development Environment

The *ControMateOLE* component has been developed and tested for use with the following programming and development application suites :-

- Microsoft Visual Basic (versions 5 and 6)
- Microsoft Visual Studio C++ (versions 5 and 6)
- Microsoft Visual Studio .NET (VB and C#) projects (as OLE object reference)
- Borland C++ Builder
- Borland Delphi

2 Component Usage

2.1 Component Reference

In order to instantiate and use the component in Visual Basic, the *Project References* menu option must be used to add a reference to the Visual Basic project. The component is named *ControlMateOLE*.

2.2 Sample Application

A sample Visual Basic 6 project is included on the CD-ROM. To install the necessary file choose the Custom installation option and select to install the sample application and source files. The source code for this is listed in Appendix B (note, this code does not cover all Attributes and procedures and is included here only for illustration purposes only).

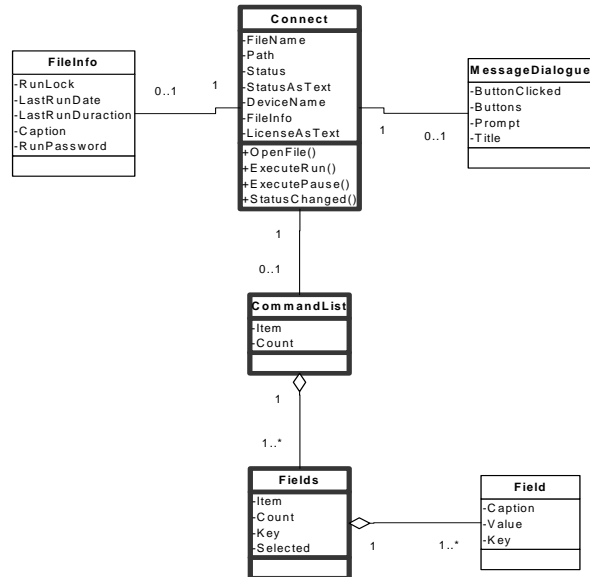
The sample application can be used as a reference or template of attribute and procedure usage. This sample application will allow a user to specify filenames and paths of *ControlMate* sequence files. The user can also Open, Execute and Pause the processing of a sequence file. In addition to this, command field values can be modified prior to a run, a list of which is included in Appendix A. The changes made are not saved in the sequence file and are therefore only valid for the run time of the application.

Command Key	Field Key	Caption	Value
REFRESH.13	cboDevice	Device	SerialMate 220 ul
REFRESH.13	chkTipUsage	Track tip usage	False
REFRESH.13	txtUseCount	Tip change count	10
REFRESH.13	cboEject	Eject location	Waste
REPEATLOOP.7	txtTitle	Title	Copy all 8 rows, 5ul
REPEATLOOP.7	txtLoop	Number of iterations	8

2.3 Class Definitions

2.3.1 Class Diagram

The class diagram below shows the component architecture and the available procedures and attributes for each class.



2.3.2 Connect Class

The connector class is used to create an instantiated *ControlMateOLE* object. It is through this object that all procedures and attributes are made available. The normal procedure for instantiating the object is shown below :-

```

Dim WithEvents oControlMateOLE As ControlMateOLE.Connect

Private Sub Form_Load()
    'create the ControlMateOLE object
    Set oControlMateOLE = New ControlMateOLE.Connect
  
```

Declaring the variable `WithEvents` will provide a means by which event messages are passed back to the client application.

2.3.2.1 Attributes

Name	Return Type	Comments
Filename	String	Name of sequence file loaded during last <i>OpenFile()</i> call
Path	String	Name of sequence file path used during last <i>OpenFile()</i>
Status	Integer	Current execution status
StatusAsText	String	Current execution status as a text message
DeviceName	String	Device name as per installation, e.g. <i>PlateMate^{Plus}</i>
CommandList	CommandList	Object containing the command list
FileInfo	FileInfo	Object containing sequence file details
LicenseAsText	String	License information for the current installation
MessageDialogue	MessageDialogue	Object is raised during file execution in place of a message box dialogue

2.3.2.2 Procedures

Name	Return	Comments
OpenFile	Boolean	Opens a <i>ControlMate</i> sequence file. Parameters: <i>Path</i> - String, path name to be used <i>Filename</i> - String, sequence file name Returns: <i>True</i> - File opened <i>False</i> - File not opened
ExecuteRun	Boolean	Executes the currently opened <i>ControlMate</i> sequence file. During execution the <i>StatusChanged</i> event can be used to determine status.
ExecutePause		Pauses the currently executing <i>ControlMate</i> sequence file. A dialogue will be shown prompting for resume or abort.

2.3.2.3 Events

Name	Comments
StatusChanged	Event is triggered whenever the <i>.Status</i> attribute changes. Parameters: <i>Status</i> - Integer, enumerated value <i>ControlMateStatus</i> as :- <i>cmBusy</i> = 1001 <i>cmWaiting</i> = 1002

Name	Comments
	<i>cmAborted = 1003</i>
	<i>cmInvalidLicense = 1004</i>
	<i>cmRunPaused = 1005</i>
	<i>cmMessageDialogue = 1006</i>
	<i>ControlMateStatus value</i>
	<i>StatusText - String, status as a text message</i>

2.3.3 CommandList Class

A *CommandList* object is instantiated when a sequence file is opened and is exposed by the *ControlMateOLE.CommandList* attribute. It is used for exposing the commands within a *ControlMate* sequence file.

If a file is not successfully opened then this object is null.

2.3.3.1 Attributes

Name	Return	Comments
Item	Fields	<i>Returns a Fields object, which itself will return individual fields.</i> Parameters: IndexKey - Variant, either it's referential value (numeric) or text key. For example .Item(1) to return the first, or .Item("ASPIRATE.3") to return the command whose key is ASPIRATE.3.
Count	Long	<i>Number of Fields objects for the current sequence file (directly relates to the number of commands).</i>

2.3.4 Fields Class

A *Fields* object is instantiated when a *CommandList* object is instantiated and used for exposing the fields within an individual command.

A *Fields* object can either be accessed by it's positional index item value or by it's index name value, both via the *CommandList.Item* attribute. For example :-

```
Set oFields = oControlMateOLE.CommandList.Item(1)
OR
Set oFields = oControlMateOLE.CommandList("ASPIRATE.3")
```

2.3.4.1 Attributes

Name	Return	Comments
Item	Fields	<i>Returns a Field object, which itself will return individual field parameters.</i> Parameters: IndexKey - Variant, either it's referential value (numeric) or text key. For example .Item(1) to return the first, or .Item("txtVolume") to return the field whose key is "txtVolume" for the specific command.
Count	Long	<i>Number of Field objects for the current command.</i>
Key	String	<i>Textual representation of the command key, e.g. "ASPIRATE.3"</i>
Selected	Boolean	<i>Determines the state of whether to include the command during execution.</i> <i>True - Include command</i> <i>False - Skip command during execution</i>

2.3.5 Field Class

A *Field* object is instantiated when a *CommandList.Fields* object is instantiated and is used for exposing the field parameters within an individual command field. The normal use for this field would be to provide direct access to a field value, for example to change a volume within an aspirate command.

A *Field* object can either be accessed by it's positional index item value or by it's index name value, both via the *CommandList.Item(...).Item(...)* attribute.

For example :-

```
`instantiate a temporary Fields object  
Set oFields = oControlMateOLE.CommandList.Item(1)  
OR  
Set oFields = oControlMateOLE.CommandList("ASPIRATE.3")
```

then

```
Set oField = oFields.Item(1)  
OR  
Set oField = oFields("txtVolume")
```

2.3.5.1 Attributes

Name	Return	Comments
Key	String	<i>Textual representation of the command key, e.g. "txtVolume"</i>
Value	Variant	<i>Actual field value, e.g. Volume. The return type is variant. When setting the value it is important that the correct casting is used, e.g. Boolean's must be Boolean and not numeric etc.</i>
Caption	String	<i>The field caption.</i>

2.3.6 FileInfo Class

A *FileInfo* object is instantiated when a *ControlMate* sequence file is successfully opened and is used for exposing the information relating to a Sequence File header.

2.3.6.1 Attributes

Name	Return	Comments
RunLock	Boolean	<i>Specifies whether a run lock is active. If it is then the RunPassword attribute must be used for validation before the file can be executed.</i>
LastRunDate	String	<i>Text representing the date of the last run, via ControlMate, for the current particular sequence file.</i>
LastRunDuration	String	<i>Text representing the duration of the last run, via ControlMate, for the current particular sequence file.</i>
Caption	String	<i>The sequence file caption text.</i>
RunPassword	String	<i>The password required if a RunLock attribute is True.</i>

2.3.7 MessageDialogue Class

During normal file execution within *ControlMate* messages are displayed to the user via the standard message box windows dialogues. Execution is then halted until the user responds by pressing the correct button (e.g. OK or Abort, Retry or Ignore). The dialogues are normally only presented due to device error.

When sequence file execution is controlled via *ControlMateOLE* it may not be advantageous to use this method, perhaps due to an unattended nature of the client application. Therefore message boxes are suppressed and instead the *StatusChanged* event is raised, with an event code of *cmMessageDialogue* (see *Connect.StatusChanged* event information above).

When the event is raised which denotes that a message box response is required the execution will halt. The client application must then do the following :-

- Use the *Buttons* attribute to determine the responses required
e.g.

```
if oControlMateOLE.MessageDialogue.Buttons and vbAbortRetryIgnore then ...
```

- Use the *Title* and *Caption* attributes to display, for example a message dialogue to the user
- Return the required resume action via the *ButtonClicked* attribute
e.g.

```
oControlMateOLE.MessageDialogue.ButtonClicked = vbRetry
```

2.3.7.1 Attributes

Name	Return	Comments
ButtonClicked		Sets the required resume action (operates in the same way that the return value to a Visual Basic <i>msgbox</i> function).
Buttons	Integer	Represents the buttons that would normally would have been displayed via a message box dialogue, e.g. <i>vbAbortRetryIgnore</i>
Prompt	String	The message box prompt
Title	String	The message box title

A Appendix – Example usage

Example component usage is shown in the following appendix sections.

A1 Opening a Sequence File

A *ControlMate* sequence file contains the programmatic sequence of commands required to control a device (such as *SerialMate* or *PlateMate^{Plus}*). A file must first be opened before it can be executed for device control through a client using *ControlMateOLE*.

The following example source code shows how :-

```
Private Sub OpenFile()  
    With moControlMateOLE  
        'the OpenFile operation will return false if unable to open file  
        If Not .OpenFile(txtPath.Text, txtFilename.Text) Then  
            MsgBox "Unable to open '" & .Path & "\" & .FileName & "'", _  
                vbApplicationModal + vbExclamation + vbOKOnly + vbMsgBoxSetForeground, _  
                "ControlMateOLE Open File"  
        Else  
            'file has opened successfully so display the file parameters  
            lblCaption = .FileInfo.Caption  
            lblLastRunDate = .FileInfo.LastRunDate  
            lblLastRunDuration = .FileInfo.LastRunDuration  
            lblRunLock = CStr(.FileInfo.RunLock)  
            txtPassword = ""  
        End If  
    End With  
End Sub
```

A2 Changing Field Values

A very powerful and useful feature of the *ControlMateOLE* component is the ability to dynamically change and set command field values within a sequence file. This allows the external integration of information such as source values (e.g. aspiration volumes) being extrapolated from databases.

The example Visual Basic code shown below illustrates how to change field values. The assumption is made here that a sequence file has already been opened.

```
Private Sub EditValue()  
    Dim oFields As ControlMateOLE.Fields  
    Dim oField As ControlMateOLE.Field  
  
    'check to make sure we actually have a command list loaded  
    If Not moControlMateOLE.CommandList Is Nothing Then  
        With grdCommandList  
            'check to make sure we have actually highlighted a row  
            If .Row > 0 Then  
                'load the edit dialogue  
                Load frmEdit  
  
                'the edit dialogue has two object properties to expose the respective  
                'command and field to be changed
```

```
        Set oFields = oControlMate.CommandList(.TextMatrix(.Row, 0))
        Set oField = oFields(.TextMatrix(.Row, 1))
        Set frmEdit.Fields = oFields
        Set frmEdit.Field = oField

        'show the form and wait
        frmEdit.Show vbModal

        'update the command list flex grid
        DrawCommandList
    End If
End With
End If
End Sub

Private Sub Set Field(ByRef oNewValue As ControlMateOLE.Field)
    'set the local member object
    Set moField = oNewValue

    'update the edit dialogue labels
    With moField
        lblFieldKey.Caption = .Key
        lblFieldCaption.Caption = .Caption
        lblValueType.Caption = TypeName(.Value)
        lblCurrentValue.Caption = .Value
        txtNewValue.Text = .Value
    End With
End Sub
```

A3 Sequence File Execution

To execute a sequence file, first open it and then call the *ExecuteStart()* function. An example of Visual Basic code is shown below:-

```
Private Sub cmdButton_Click(Index As Integer)
    'process the selected button index value
    With moControlMate
        Select Case Index
            Case conCMDOPEN
                OpenFile
            Case conCMDEXECUTE
                'the execute run function will return false if unable to run file
                If Not .ExecuteRun() Then
                    MsgBox "Unable to execute '" & .Path & "\" & .FileName & "'.", _
                        vbApplicationModal + vbExclamation + vbOKOnly _
                        + vbMsgBoxSetForeground, "ControlMateOLE Execute Run"
                End If
            Case conCMDPAUSE
                .ExecutePause
            Case conCMDEDIT
                EditValue
            Case conCMDOK
                Unload Me
        End Select
    End With
End Sub
```

A4 Message Boxes

An example of the *Connect.MessageDialogue* attribute is to simply generate a message box whenever the *StatusChanged* event is raised due to a message event. Although this is not the suggested normal use it does demonstrate the use of the return attributes and user interface information.

```
Private Sub oControlMate_StatusChanged(ByVal iStatus As
ControlMateOLE.ControlMateStatus, ByVal sStatusText As String)
'optional status value conditional processing
Select Case iStatus
Case cmBusy
Case cmWaiting
Case cmInvalidLicense
Case cmMessageDialogue
    With oControlMate.MessageDialogue
        'set the return value to that of the user selection
        .ButtonClicked = MsgBox(.Prompt, vbExclamation + vbApplicationModal _
+ .Buttons, .Title)
    End With
Case cmRunPaused
Case cmAborted
End Select

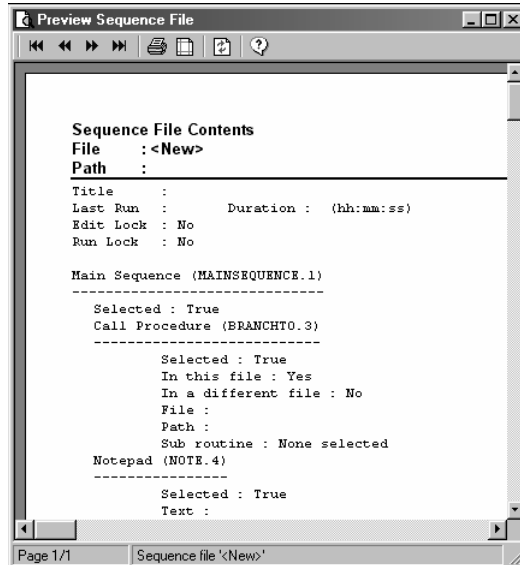
'display the status values
lblStatus = iStatus & ", " & sStatusText
End Sub
```

B Appendix - Sequence File Command and Field Names

This appendix lists the command and field names available for each *ControMate* device module. These field names are to be used when referencing items for the respective *Fields* and *Field* objects.

During run time each command is assigned a unique key which is a concatenation of its generic name, for example ASPIRATE, and its sequence index. A new sequence index value is generated each time a command is added to the sequence file, for example adding a Move and Aspirate command may generate two keys as *MOVE.5* and *ASPIRATE.6* respectively. If a Move and Dispense command were then added to the sequence their keys would be *MOVE.7* and *DISPENSE.8*.

The unique command keys can be viewed via the *ControMate* Print Preview menu option an example of which is shown below :-



B1 *ControMate* Intrinsic Commands

Command (Key)	Field	Field Key
Call Procedure (BRANCHTO)	Call within this file selected	chkThisFile
	Call external file selected	chkDifferentFile
	External filename	lblFileName
	External file path	lblPath
	Procedure name	cboList

Command (Key)	Field	Field Key
Notepad (NOTE)	Text	txtNote
Group Commands (REPEATLOOP)	Caption	txtTitle
	Iterations	txtLoop

B2 SerialMate Commands

Command (Key)	Field	Field Key
Aspirate (ASPIRATE)	Mix selected	chkMix
	Mix cycle count	txtMixCycles
	Mix dwell time	txtMixDelay
	Volume	txtVolume
	Air blow-out selected	chkBlowOut
	Air blow out volume	txtBlowOutVolume
	Liquid level sense selected	chkLiquidLevel
	Tip touch selected	chkTipTouch
	Post air gap selected	chkPostAirGap
	Dwell time	txtAspirateDelay
Dispense (DISPENSE)	Dispense all selected	chkDispenseAll
	Volume	txtVolume
	Tip touch selected	chkTipTouch
	Mix selected	chkMix
	Mix cycle count	txtMixCycles
	Mix dwell time	txtMixDelay
	Mix volume	txtMixVolume
	Post dispense shake selected	chkShake
	Shake duration	txtSeconds
	Dwell time	txtDispenseDelay
Collect Tips (CHANGETIPS)	Track tip usage	chkTipUsage
	Tip usage count	txtUseCount
	Tip eject location	cboEject
Home Pipettor (HOMETIPS)	Eject tip selected	chkEject
Move Pipettor (MOVETIPS)	Tip eject location	cboEject
	Vessel type	cboPlateType
	Channel selection	cboOrientation
	Position	txtPosition
	Start column	txtStartColumn
	End column	txtEndColumn
	Height	txtHeight
	Pause (PAUSE)	Pause for fixed duration selected
	Pause duration	txtSeconds
	Pause and wait for user selected	chkPressKey
	Sound device alarm selected	chkAlarm
	Alarm sound duration	txtAlarmSeconds
Purge Tips (PURGETIPS)	Immediate purge selected	chkPurgeNow
	Purge at end of run selected	chkPurgeRunEnd

Command (Key)	Field	Field Key
Speed Control (SPEEDCONTROL)	Purge to waste selected	chkWaste
	Purge to reservoir selected	chkReservoir
	Reservoir position	txtPosition
	Reservoir type	cboPlateType
	Piston speed control selected	chkPiston
	Piston speed value	hsbPiston
	Z axis speed control selected	chkVertical
	Z axis speed value	hsbVertical

B3 *PlateMate*^{Plus} Commands

Command (Key)	Field	Field Key
Aspirate (ASPIRATE)	Volumetric	cboVolumetric
	Air gap selected	chkAirGap
	Air gap volume	txtAirGap
	Volume	txtVolume
	Overstroke selected	chkOverstroke
	Dwell time	txtDwellTime
	Tip touch selected	chkTipTouch
Dispense (DISPENSE)	Tip touch action	cboTipTouch
	Volumetric	cboVolumetric
	Dispense all with blow-out selected	optBlowOut
	Blow out volume	txtBlowOut
	Dispense all selected	optDispenseAll
	Specific volume selected	optVolume
	Specific volume	txtVolume
Change Tip Magazine (CHANGETIPS)	Dwell time	txtDwellTime
	Tip touch selected	chkTipTouch
	Tip touch action	cboTipTouch
	Track tip usage selected	chkTipUsage
	Tip usage count	txtUseCount
	Manual change selected	optManual
	Automatic change selected	optAutomatic
Home Axes (Home)	Home all axes selected	chkAll
	Home piston axis selected	chkPiston
	Home Stage 1 axis selected	chkStage1
	Home Stage 2 axis selected	chkStage2
	Home Stage 3 axis selected	chkStage3
Mix (MIX)	Home Stage 4 axis selected	chkStage4
	Mix cycle count	txtMixCycles
	Mix volume	txtMixVolume
	Air blow out selected	chkBlowOut
	Air blow out pre set height	cboHeightPreset
Move to Position (MOVE)	Air blow out volume	txtBlowOut
	Position	cboPosition
	Vessel type	cboPlateType
	Pre set height selected	optHeightPreset

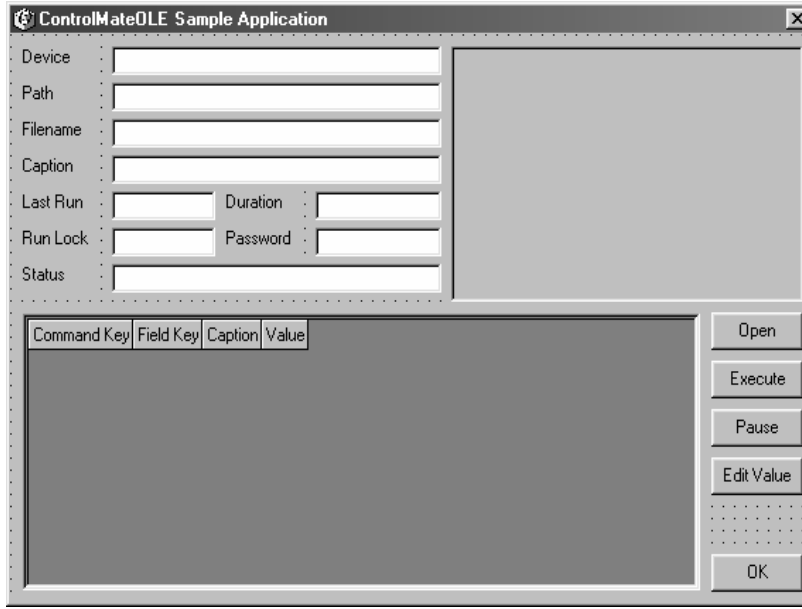
Command (Key)	Field	Field Key
	Pre set height	cboHeightPreset
	Specific height selected	optHeightSpecific
	Specific height	txtHeight
	Get plate from stacker selected	optGetPlate
	Jog plate selected	chkPlateJog
	Put plate in stacker selected	optPutPlate
	Quadrant	cboQuadrant
	Tip well offset selected	chkTipWell
	Predefined tip well offset selected	optPredefined
	Predefined tip well offset	cboTipWell
	Specific tip well offset selected	optSpecific
	Specific x axis offset	txtXoffset
	Specific y axis offset	txtYoffset
	Incremental column move selected	chkColumn
	Start column	txtStart
	End column	txtEnd
	Column increment	txtIncrement
	Fill reservoir selected	chkFillReservoir
	Fill via sensor trigger selected	optFillSensor
	Fill via fixed duration selected	optFillTime
Pause (PAUSE)	Pause for time period selected	optDelay
	Pause duration	txtSeconds
	Sound alarm after pause selected	chkAlarm
	Pause and wait for user selected	optPresskey
Speed Control (SPEEDCONTROL)	Piston speed control selected	chkPiston
	Piston axis speed value	hsbPiston
	Z axis speed control selected	chkVertical
	Z axis speed value	hsbVertical
	Y axis speed control selected	chkFrontToBack
	Y axis speed value	hsbFrontToBack
	X axis speed control selected	chkLeftToRight
	X axis speed value	hsbLeftToRight
Wash Tips (WASHTIPS)	Tip wash cycles	txtCycles
	Tip wash volume	txtVolume
	Air blow out selected	chkBlowOut
	Air blow out volume	txtBlowOut

C Appendix - Sample Application

The sample application that is supplied on the distribution CD-ROM demonstrates the usage of the component for opening, editing and running *ControlMate* Sequence Files.

C1 Sample Application Source Code

The source code for sample application is listed here as an example reference. It is not intended to demonstrate the only method by which the component can be used.



```
\*****  
\ Application   : ControlMateOLESampleApp.exe  
\ Description   : Sample application demonstrating the use of the  
\                ControlMateOLE component  
\*****  
Option Explicit  
\main object component, declared as WithEvents to the StatusChanged event  
  
Dim WithEvents oControlMate As ControlMateOLE.Connect  
  
\command button constants  
Const conCMDOPEN   As Integer = 0  
Const conCMDEDIT   As Integer = 1  
Const conCMDEXECUTE As Integer = 2  
Const conCMDPAUSE   As Integer = 3  
Const conCMDOK     As Integer = 4  
  
\*****
```

```
' Sub          : cmdButton_Click
' Description   : Command button click event
' Parameters    : Index : Integer, index number of button clicked
' Returns       : Nothing
'=====
Private Sub cmdButton_Click(Index As Integer)
'process the selected button index value
With oControlMate
  Select Case Index
    Case conCMDOPEN
      OpenFile
    Case conCMDEXECUTE
      'the execute run function will return false if unable to run file
      If Not .ExecuteRun() Then
        MsgBox "Unable to execute '" & .Path & "\" & .FileName _
          & "'.", vbApplicationModal + vbExclamation + vbOKOnly _
          vbMsgBoxSetForeground, "ControlMateOLE Execute Run"
      End If
    Case conCMDPAUSE
      .ExecutePause
    Case conCMDEDIT
      EditValue
    Case conCMDOK
      Unload Me
  End Select
End With
End Sub

'=====
' Sub          : Form_load
' Description   : Initial procedure called upon form instantiation
' Parameters    : None
' Returns       : Nothing
'=====
Private Sub Form_Load()
'create the ControlMateOLE object
Set oControlMate = New ControlMateOLE.Connect

'display the license as returned by the license component
lblLicense.Caption = oControlMate.LicenseAsText

'display a status message, the component will always initialise
'with a waiting status
'if the license is valid or license invalid status
With oControlMate
  lblStatus.Caption = .Status & ", " & .StatusAsText
  lblDevice = .DeviceName
End With
End Sub

'=====
' Sub          : Form_Unload
' Description   : Called upon form instance termination
' Parameters    : Cancel : Integer, determines whether or not to
'               :               cancel the cancel event
' Returns       : Nothing
'=====
Private Sub Form_Unload(Cancel As Integer)
'kill the object
Set oControlMate = Nothing
End Sub

'=====
```

```

\ Sub          : oControlMate_StatusChanged
\ Description  : Called whenever the StatusChanged event is triggered
\ Parameters   : iStatus -
\              enumerated:ControlMateOLE.ControlMateStatus, status as a value
\              sStatusText - string, status as text
\ Returns      : Nothing
=====
Private Sub oControlMate_StatusChanged(ByVal iStatus As _
    ControlMateOLE.ControlMateStatus, ByVal sStatusText As String)
    'optional status value conditional processing
    Select Case iStatus
        Case cmBusy
        Case cmWaiting
        Case cmInvalidLicense
    End Select

    'display the status values
    lblStatus = iStatus & ", " & sStatusText
End Sub

\ Sub          : txtPassword_Change
\ Description  : Raised when the run password field is edited
\ Parameters   : None
\ Returns      : Nothing
=====
Private Sub txtPassword_Change()
    'change the run password setting if the file has been opened
    '(i.e. the fileinfo object exists)
    If Not oControlMate.FileInfo Is Nothing Then
        oControlMate.FileInfo.RunPassword = txtPassword.Text
    End If
End Sub

\ Sub          : OpenFile
\ Description  : Open a ControlMate sequence file for processing
\ Parameters   : None
\ Returns      : Nothing
=====
Private Sub OpenFile()
    With oControlMate
        'the OpenFile operation will return false if unable to open file
        If Not .OpenFile(txtPath.Text, txtFilename.Text) Then
            MsgBox "Unable to open '" & .Path & "\" & .FileName & "'.", _
                vbApplicationModal + vbExclamation + vbOKOnly _
                vbMsgBoxSetForeground, "ControlMateOLE Open File"
        Else
            'file has opened successfully so display the file parameters
            lblCaption = .FileInfo.Caption
            lblLastRunDate = .FileInfo.LastRunDate
            lblLastRunDuration = .FileInfo.LastRunDuration
            lblRunLock = CStr(.FileInfo.RunLock)
            txtPassword = ""
        End If
    End With

    'update the command list grid
    DrawCommandList
End Sub

\ Sub          : DrawCommandList

```

```

` Description      : Populates the command list flex grid with
`                   commands associated with the current
`                   open file.
` Parameters       : None
` Returns          : Nothing
=====
Private Sub DrawCommandList()
    Dim oCommandList As ControlMateOLE.CommandList
    Dim oFields As ControlMateOLE.Fields
    Dim oField As ControlMateOLE.Field

    `create a throw away command list object
    Set oCommandList = oControlMate.CommandList

    `if the object was created then update the grid
    If Not oCommandList Is Nothing Then
        With grdCommandList
            `prevents grid flicker during update
            .Redraw = False

            `rebuild it from scratch
            .Clear
            .Rows = 2           `MSFlexGrid tweak to get around the
            .FixedRows = 1     `problem of wanting an empty
            .Rows = 1          `grid with column headings - not
                             `allowed to have fixed
                             `without data rows :-)
            .FormatString = "<Command Key|<Field Key|<Caption|<Value"

            `set all column widths to 1500
            .ColWidth(-1) = 1500

            `now work through the command list to pull out each command
            `the command list is enumerated so allows "for each" syntax
            `and holds the command
            `in the same sequence as they are meant to be processed
            For Each oFields In oCommandList
                `now work through each field for the respective command
                `the field list is enumerated so allows "for each" syntax
                `the order of the
                `commands is not important
                For Each oField In oFields
                    `add an entry to the bottom of the grid
                    .AddItem oFields.Key & vbTab & oField.Key & vbTab _
                        & oField.Caption & vbTab & oField.Value
                Next oField
                `add a blank line after the command (separator, makes it
                `easier to read)
                .AddItem ""
            Next oFields

            `display the grid
            .Redraw = True
        End With
    End If
End Sub

=====
` Sub              : EditValue
` Description      : Allows the editing of a field value. The value to
`                   be edited is whichever row
`                   is highlighted in the command list flex grid
` Parameters       : None

```

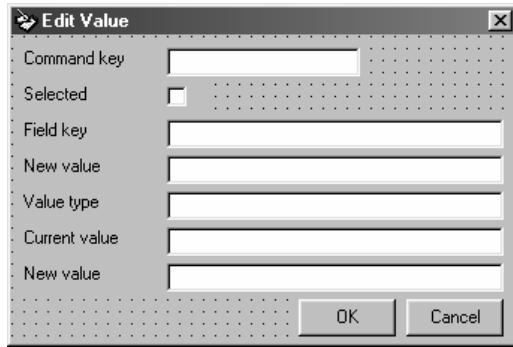
```
` Returns      : Nothing
`=====
Private Sub EditValue()
Dim oFields As ControlMateOLE.Fields
Dim oField As ControlMateOLE.Field

`check to make sure we actually have a command list loaded
If Not oControlMate.CommandList Is Nothing Then
    With grdCommandList
        `check to make sure we have actually highlighted a row
        If .Row > 0 Then
            `load the edit dialogue
            Load frmEdit

            `the edit dialogue has two object Attributes to expose the
            `respective command and field to be changed
            Set oFields = oControlMate.CommandList(.TextMatrix(.Row, 0))
            Set oField = oFields(.TextMatrix(.Row, 1))
            Set frmEdit.Fields = oFields
            Set frmEdit.Field = oField

            `show the form and wait
            frmEdit.Show vbModal

            `update the command list flex grid
            DrawCommandList
        End If
    End With
End If
End Sub
```



```

'*****
' Application   : ControlMateOLESampleApp.exe
' Description   : Edit dialogue
'*****
Option Explicit
'private objects for command and field to be edited
Private moFields As ControlMateOLE.Fields
Private moField As ControlMateOLE.Field

'command button constants
Private Const conCMDOK      As Integer = 0
Private Const conCMDCANCEL As Integer = 1

'=====
' Attribute Set : Fields
' Description   : Sets the moFields object value to the respective
'                 command containing the field to be edited
' Parameters    : oNewValue - Object reference to a
'                 ControlMateOLE.Fields
' Returns       : Nothing
'=====
Public Attribute Set Fields(ByRef oNewValue As ControlMateOLE.Fields)
' set the local member object
Set moFields = oNewValue

'update the edit dialogue label
lblCommandKey.Caption = moFields.Key
chkSelected.Value = Abs(moFields.Selected)
End Attribute

'=====
' Attribute Set : Field
' Description   : Sets the moField object value to the respective
'                 field to be edited
' Parameters    : oNewValue - Object reference to a
'                 ControlMateOLE.Field
' Returns       : Nothing
'=====
Public Attribute Set Field(ByRef oNewValue As ControlMateOLE.Field)
' set the local member object
Set moField = oNewValue

'update the edit dialogue labels
With moField
    lblFieldKey.Caption = .Key
    lblFieldCaption.Caption = .Caption

```

```

        lblValueType.Caption = TypeName(.Value)
        lblCurrentValue.Caption = .Value
        txtNewValue.Text = .Value
    End With
End Attribute

'=====
' Sub          : cmdButton_Click
' Description  : Command button click event
' Parameters   : Index : Integer, index number of button clicked
' Returns      : Nothing
'=====
Private Sub cmdButton_Click(Index As Integer)
    Select Case Index
        Case conCMDOK
            SaveChanges
            Unload Me
        Case conCMDCANCEL
            Unload Me
    End Select
End Sub

'=====
' Sub          : SaveChanges
' Description  : Saves the changes made to the new value. This
'              : routine is required due to the type casting
'              : required for the values. The field values are of
'              : type Variant however, in this sample app the edit
'              : field is a text box so type caster is required to
'              : ensure that the field values are set according to
'              : their original type.
' Parameters   : None
' Returns      : Nothing
'=====
Private Sub SaveChanges()
    'set the command selected value. This determines whether or not to
    'include the command during run execution
    moFields.Selected = CBool(chkSelected.Value)

    'now set the field value to the new value
    With moField
        Select Case TypeName(.Value)
            Case "String"
                .Value = txtNewValue
            Case "Integer"
                .Value = CInt(Val(txtNewValue))
            Case "Long"
                .Value = CLng(Val(txtNewValue))
            Case "Boolean"
                .Value = CBool(txtNewValue)
            Case "Date"
                .Value = CDate(txtNewValue)
            Case "Double"
                .Value = CDbl(Val(txtNewValue))
            Case "Single"
                .Value = CSng(Val(txtNewValue))
        End Select
    End With
End Sub

'=====
' Sub          : Form_Unload
' Description  : Called upon form instance termination

```

```
` Parameters      : Cancel : Integer, determines whether or not to
`                  cancel the cancel event
` Returns         : Nothing
`=====
Private Sub Form_Unload(Cancel As Integer)
`kill the objects
  Set moField = Nothing
  Set moFields = Nothing
End Sub
```


Matrix Technologies Corp.
22 Friars Drive
Hudson, NH 03051, USA
Toll-free: (800) 345 0206
www.matrixtechcorp.com