```vb
Imports System.IO.Ports
Imports System.Windows.Forms.VisualStyles.VisualStyleElement

Public Class SerialPortForm
    ' --- NEW: Ring Counter State & Timer ---
    Private WithEvents RingTimer As New System.Windows.Forms.Timer()
    ' Tracks the current index (1-6) for the ring counter sequence
    Private RingCounterStep As Integer = 1
    ' ----------------------------------------

    ' This method now checks if the port is open before attempting to configure and open it.
    Sub Connect()
        If Not SerialPort1.IsOpen Then
            ' Set port configuration
            SerialPort1.BaudRate = 9600 'Q@ Board Default
            SerialPort1.Parity = Parity.None
            SerialPort1.StopBits = StopBits.One
            SerialPort1.DataBits = 8
            SerialPort1.PortName = "COM5"

            Try
                SerialPort1.Open()
                Console.WriteLine("COM port opened successfully.")
            Catch ex As Exception
                ' Handle the case where the port cannot be opened (e.g., in use, wrong port name)
                Console.WriteLine($"Error opening COM port: {ex.Message}")
            End Try
        Else
            Console.WriteLine("COM port is already open.")
        End If
    End Sub

    ' The form's Load event is the best place to call Connect initially.
    Private Sub SerialPortForm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        ' Initialize the Ring Counter Timer
        RingTimer.Interval = 100 ' Set rotation speed to 250ms (4 steps per second)
        RingTimer.Enabled = False ' Start disabled

        Connect()
    End Sub

    ' New: Centralized function to handle the serial write logic based on an index (1-8).
    Sub SendCommand(ByVal caseIndex As Integer)
        If Not SerialPort1.IsOpen Then
```

```
                Console.WriteLine("Command skipped: COM port is closed.")
                Return
            End If

            Dim byteToSend(1) As Byte

            ' The caseIndex (from 1 to 8) now determines which command is 
              sent.
            Select Case caseIndex
                Case 1  ' Step 0: ~0.50 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H0   ' 0x00 (bits 
                      7:3=00000)
                Case 2  ' Step 1: ~0.56 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H8   ' 0x08 (bits 
                      7:3=00001)
                Case 3  ' Step 2: ~0.63 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H10  ' 0x10
                Case 4  ' Step 3: ~0.69 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H18  ' 0x18
                Case 5  ' Step 4: ~0.76 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H20  ' 0x20
                Case 6  ' Step 5: ~0.82 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H28  ' 0x28
                Case 7  ' Step 6: ~0.88 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H30  ' 0x30
                Case 8  ' Step 7: ~0.95 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H38  ' 0x38
                Case 9  ' Step 8: ~1.01 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H40  ' 0x40
                Case 10 ' Step 9: ~1.08 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H48  ' 0x48
                Case 11 ' Step 10: ~1.14 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H50  ' 0x50
                Case 12 ' Step 11: ~1.22 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H58  ' 0x58
                Case 13 ' Step 12: ~1.28 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H60  ' 0x60
                Case 14 ' Step 13: ~1.34 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H68  ' 0x68
                Case 15 ' Step 14: ~1.41 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H70  ' 0x70
                Case 16 ' Step 15: ~1.47 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H78  ' 0x78
                Case 17 ' Step 16: ~1.54 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H80  ' 0x80
                Case 18 ' Step 17: ~1.60 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H88  ' 0x88
                Case 19 ' Step 18: ~1.66 ms
                    byteToSend(0) = &H24 : byteToSend(1) = &H90  ' 0x90
```

```vb
            Case 20 ' Step 19: ~1.73 ms
                byteToSend(0) = &H24 : byteToSend(1) = &H98  ' 0x98
            Case 21 ' Step 20: ~1.79 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HA0  ' 0xA0
            Case 22 ' Step 21: ~1.86 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HA8  ' 0xA8
            Case 23 ' Step 22: ~1.92 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HB0  ' 0xB0
            Case 24 ' Step 23: ~1.98 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HB8  ' 0xB8
            Case 25 ' Step 24: ~2.05 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HC0  ' 0xC0
            Case 26 ' Step 25: ~2.11 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HC8  ' 0xC8
            Case 27 ' Step 26: ~2.18 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HD0  ' 0xD0
            Case 28 ' Step 27: ~2.24 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HD8  ' 0xD8
            Case 29 ' Step 28: ~2.30 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HE0  ' 0xE0
            Case 30 ' Step 29: ~2.37 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HE8  ' 0xE8
            Case 31 ' Step 30: ~2.43 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HF0  ' 0xF0
            Case 32 ' Step 31: ~2.50 ms
                byteToSend(0) = &H24 : byteToSend(1) = &HF8  ' 0xF8
            Case Else
                Console.WriteLine($"Invalid index: {caseIndex}")
                Return
        End Select

        ' Write the 2-byte command
        SerialPort1.Write(byteToSend, 0, 2)
        Console.WriteLine($"Sent command for Case {caseIndex} (Value: &H
          {byteToSend(1).ToString("X2")})")
        UpdateLogBox($"Sent command for Case {caseIndex} (Value: &H
          {byteToSend(1).ToString("X2")})")
    End Sub
    Private Sub UpdateLogBox(ByVal text As String)
        ' This ensures the update happens safely on the UI thread
        If Me.TransmissionToPicTextBox.InvokeRequired Then
            Me.Invoke(Sub() UpdateLogBox(text))
        Else
            Me.TransmissionToPicTextBox.AppendText(text &
              Environment.NewLine)
            Me.TransmissionToPicTextBox.ScrollToCaret()
        End If
    End Sub
    Sub Write()
```

```
        If SerialPort1.IsOpen Then
            Dim data(0) As Byte 'put bytes into array
            data(0) = &B0 'actual data as a byte
            SerialPort1.Write(data, 0, 1) 'send bytes as array, start at index 0, send 1 byte
        Else
            Console.WriteLine("Error: Cannot Write. COM port is closed.")
        End If
    End Sub

    Sub Output_High()
        ' Now calls SendCommand Case 8 (All High)
        SendCommand(32)
    End Sub

    Sub Output_Low()
        ' Now calls SendCommand Case 7 (All Low)
        SendCommand(1)
    End Sub

    Sub Read()
        ' Reading only happens if data is available (triggered by DataReceived event)
        Try
            Dim bytesToRead As Integer = SerialPort1.BytesToRead
            If bytesToRead > 0 Then
                Dim data(bytesToRead - 1) As Byte ' Array size is bytesToRead - 1 (0-based)
                SerialPort1.Read(data, 0, bytesToRead)

                For i = 0 To UBound(data)
                    Console.WriteLine($"Byte {i}: {Chr(data(i))}")
                Next

                Console.WriteLine($"Bytes read: {bytesToRead}")
            End If
        Catch ex As Exception
            Console.WriteLine($"Error during Read operation: {ex.Message}")
        End Try
    End Sub

    Function CheckIfQuietBoard() As Boolean
        If SerialPort1.IsOpen Then
            Dim bytes(0) As Byte
            bytes(0) = &B11110000
            SerialPort1.Write(bytes, 0, 1)
            Return True
        Else
```

```vb
            Console.WriteLine("Error: Cannot CheckIfQuietBoard. COM port is closed.")
            Return False
        End If
    End Function

    ' --- Ring Counter Logic ---
    Sub RingCounter()
        If RingTimer.Enabled Then
            RingTimer.Stop()
            ' Stop the rotation and turn all outputs OFF (Case 7)
            SendCommand(7)
            Console.WriteLine("Ring Counter Stopped.")
        Else
            ' Reset the step to start at the first output (Case 1)
            RingCounterStep = 1
            RingTimer.Start()
            Console.WriteLine("Ring Counter Started.")
        End If
    End Sub

    ' Event handler that fires every time the RingTimer interval elapses
    Private Sub RingTimer_Tick(sender As Object, e As EventArgs) Handles RingTimer.Tick
        ' The ring counter cycles through Cases 1 through 6
        If RingCounterStep > 32 Then
            RingCounterStep = 1 ' Wrap back to the first step
        End If

        ' Send the command for the current step
        SendCommand(RingCounterStep)

        ' Move to the next step
        RingCounterStep += 1
    End Sub

    ' --- Event Handlers ---

    Private Sub SerialPortForm_Click(sender As Object, e As EventArgs) Handles Me.Click
        Write()
    End Sub

    Private Sub SerialPort1_DataReceived(sender As Object, e As SerialDataReceivedEventArgs) Handles SerialPort1.DataReceived
        ' 1. Read ALL available bytes into the buffer.
        '    BytesToRead is volatile, but this operation will empty the buffer
        '    of whatever was there when it executes.
        Dim bytesToRead As Integer = SerialPort1.BytesToRead
        Dim buffer(bytesToRead - 1) As Byte

        ' This single Read() command extracts all the data
        SerialPort1.Read(buffer, 0, bytesToRead)

        ' 2. Convert and update the UI using Me.Invoke (essential for
          thread safety)
        Dim hexData As String = ConvertBytesToHexString(buffer)

        Me.Invoke(Sub()
                      UpdateTextBox(hexData)
                  End Sub)

        Try
            Console.WriteLine($"Data received. Bytes read: {bytesToRead}.
              Remaining: {SerialPort1.BytesToRead}")
        Catch ex As Exception
            Console.WriteLine("oops! Error accessing BytesToRead.")
        End Try
    End Sub

    Private Function ConvertBytesToHexString(ByVal data As Byte()) As
      String
        Dim sb As New System.Text.StringBuilder()
        For Each b As Byte In data

            sb.Append(b.ToString("X2") & " ")
        Next

        Return sb.ToString().TrimEnd()

    End Function

    Private Sub UpdateTextBox(ByVal text As String)
        VBRecieveTextBox.AppendText(text & Environment.NewLine)
        VBRecieveTextBox.ScrollToCaret()

    End Sub

    Private Sub SendDataButton_Click(sender As Object, e As EventArgs)
      Handles SendDataButton.Click
        If Not SerialPort1.IsOpen Then
            Console.WriteLine("Error: Cannot Send Data. COM port is
              closed.")
            UpdateLogBox("ERROR: COM port is closed. Cannot send data.")
            Return
        End If

        ' **ASSUMPTION:** The text box is named 'DataToSendTextBox'
        Dim hexInput As String = InputTextBox.Text

        If String.IsNullOrWhiteSpace(hexInput) Then
            Console.WriteLine("Cannot send: Text box is empty.")
            Return
        End If

        Try
            ' 1. Convert the Hex string (e.g., "24 F8") into a Byte array
            Dim dataToSend As Byte() = ConvertHexStringToByteArray(hexInput)

            ' 2. Write the byte array to the serial port
            SerialPort1.Write(dataToSend, 0, dataToSend.Length)

            ' 3. Log the action
            Console.WriteLine($"Sent {dataToSend.Length} bytes: {hexInput.Trim()}")
            UpdateLogBox($"Sent Bytes: {hexInput.Trim()}")

        Catch ex As FormatException
            ' Handle error from the conversion function
            Console.WriteLine($"Error in hex format: {ex.Message}")
            UpdateLogBox($"ERROR: Invalid Hex Format. {ex.Message}")
        Catch ex As Exception
            ' Handle general serial port error
            Console.WriteLine($"Error sending data: {ex.Message}")
            UpdateLogBox($"ERROR: Serial Write Failed. {ex.Message}")
        End Try
    End Sub

    ''' <summary>
    ''' Converts a space-separated string of hex values (e.g., "24 F8 0A") into a Byte array.
    ''' </summary>
    Private Function ConvertHexStringToByteArray(ByVal hexString As String) As Byte()
        ' Remove leading/trailing spaces and split the string by spaces
        Dim hexValues As String() = hexString.Trim().Split(" "c)

        ' Determine the size of the output array
        Dim byteCount As Integer = hexValues.Length
        If byteCount = 0 Then Return New Byte() {} ' Return empty array if input is empty

        Dim bytes As Byte() = New Byte(byteCount - 1) {}

        For i As Integer = 0 To byteCount - 1
```

```vb
            Try
                ' Remove any non-hex characters (like commas, if present)
                Dim hex As String = hexValues(i).Trim().Replace(",", "")

                ' Convert the 1 or 2 character hex string to a Byte
                bytes(i) = Convert.ToByte(hex, 16) ' Base 16 (Hexadecimal)
            Catch ex As Exception
                ' Handle conversion error (e.g., "G2" is not valid hex)
                Throw New FormatException($"Invalid hexadecimal value 
                  found: '{hexValues(i)}'", ex)
            End Try
        Next

        Return bytes
    End Function

    Private Sub HighOutputButton_Click(sender As Object, e As EventArgs) 
      Handles HighOutputButton.Click
        Output_High()
    End Sub

    Private Sub LowOutputButton_Click(sender As Object, e As EventArgs) 
      Handles LowOutputButton.Click
        Output_Low()
    End Sub

    ' NEW: TrackBar Scroll Event Handler
    Private Sub TrackBar1_Scroll(sender As Object, e As EventArgs) 
      Handles TrackBar1.Scroll
        ' Stop the ring counter if the user manually adjusts the output
        If RingTimer.Enabled Then
            RingTimer.Stop()
            Console.WriteLine("Ring Counter Stopped by TrackBar input.")
        End If
        ' This sends the command whenever the TrackBar position changes.
        SendCommand(TrackBar1.Value)
    End Sub

    Private Sub RingCounterButton_Click(sender As Object, e As EventArgs) 
      Handles RingCounterButton.Click
        RingCounter()
    End Sub

    ' Ensure the port is closed when the form closes
    Private Sub SerialPortForm_FormClosing(sender As Object, e As 
      FormClosingEventArgs) Handles Me.FormClosing
        If SerialPort1.IsOpen Then
            SerialPort1.Close()
        End If
    End Sub


End Class
```