ภาคผนวก ก

ซอร์สโคดโปรแกรมโมดูล

ก.1 ซอร์โคด AVICAP.BAS

```
'Windows API Constant
Public Const WM_USER = &H400
Public Const WM_CAP_START = WM_USER
Public Const WM_CAP_DRIVER_CONNECT = WM_CAP_START + 10
Public Const WM_CAP_DRIVER_DISCONNECT = WM_CAP_START + 11
Public Const WM_CAP_EDIT_COPY = WM_CAP_START + 30
Public Const WM_CAP_DLG_VIDEOSOURCE = WM_CAP_START + 42
Public Const WM_CAP_SET_PREVIEW = WM_CAP_START + 50
Public Const WM_CAP_SET_PREVIEWRATE = WM_CAP_START + 52
Public Const WM_CAP_SET_SCALE = WM_CAP_START + 53
Public Const WM_CAP_GRAB_FRAME = WM_CAP_START + 60
Public Const WM_CAP_GRAB_FRAME_NOSTOP = WM_CAP_START + 61
Public Const WM_CAP_ABORT = WM_CAP_START + 69

Public Const WS_CHILD = &H40000000
Public Const WS_VISIBLE = &H10000000
Public Const SWP_NOMOVE = &H2
Public Const SWP_NOZORDER = &H4
Public Const HWND_BOTTOM = 1

'Call Libraly
Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
   (ByVal hwnd As Long, _
   ByVal wMsg As Long, _
   ByVal wParam As Integer, _
   ByVal lParam As Long) As Long

Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, _
   ByVal hWndInsertAfter As Integer, _
   ByVal x As Integer, _
   ByVal y As Integer, _
   ByVal cx As Integer, _
   ByVal cy As Integer, _
   ByVal wFlags As Integer) As Integer
```

```
Declare Function DestroyWindow Lib "user32" (ByVal hndw As Integer) As Boolean

Declare Function capCreateCaptureWindowA Lib "avicap32.dll" ( _
   ByVal lpszWindowName As String, _
   ByVal dwStyle As Long, _
   ByVal x As Long, ByVal y As Long, _
   ByVal nWidth As Long, ByVal nHeight As Integer, _
   ByVal hWndParent As Long, ByVal nID As Long) As Long
Declare Function capGetDriverDescriptionA Lib "avicap32.dll" ( _
   ByVal wDriver As Integer, _
   ByVal lpszName As String, _
   ByVal cbName As Long, _
   ByVal lpszVer As String, _
   ByVal cbVer As Long) As Boolean
```

ก.2 ซอร์สโคด ANN.BAS

```
Option Base 1
Option Explicit

Const e = 2.7183 'Natural No. const, used in sigmod function

Private Type Dendrite ' Dendrite connects one neuron to another
Weight As Double 'Weight it has
PreviousWeightChanged As Double 'Last changed of wieght
End Type

Private Type Neuron 'The main thing
Dendrites() As Dendrite 'Array of Denrites
DendriteCount As Long 'Number of dendrites
Bias As Double 'The bias
Value As Double 'The value to be passed to next layer of neurons
Delta As Double 'The delta of neuron (used while learning)
PreviousBiasChanged As Double 'Last changed of bias
End Type

Private Type Layer 'Layer contaning number of neurons
Neurons() As Neuron 'Neurons in the layer
NeuronCount As Long 'Number of neurons
End Type

Private Type NeuralNetwork
Layers() As Layer 'Layers in the network
LayerCount As Long 'Number of layers
LearningRate As Double 'The learning rate of the network
```

```
MomentumConst As Double 'The momentum for train network
Err() As Double 'The error of Network
End Type

Dim network As NeuralNetwork ' Our main network
Function SetLearningRate(LearningRate As Double) As Boolean
'Assign new learnig rate
network.LearningRate = LearningRate
SetLearningRate = True
End Function

Function CreateNet(LearningRate As Double, ArrayOfLayers As Variant) As Integer
'0 = Unsuccesful and 1 = Successful
Dim i, j, k As Integer
network.LayerCount = UBound(ArrayOfLayers) 'Init number of layers
If network.LayerCount < 2 Then 'Input and output layers must be there
   CreateNet = 0 'Unsuccessful
   Exit Function
End If
network.LearningRate = LearningRate 'The learning rate
network.MomentumConst = 0.7 'Most used momentum
ReDim network.Layers(network.LayerCount) As Layer 'Redim the layers variable
For i = 1 To UBound(ArrayOfLayers) ' Initialize all layers
DoEvents

   network.Layers(i).NeuronCount = ArrayOfLayers(i)
   ReDim network.Layers(i).Neurons(network.Layers(i).NeuronCount) As Neuron
   For j = 1 To ArrayOfLayers(i) 'Initialize all neurons
   DoEvents
      If i = UBound(ArrayOfLayers) Then 'We will not init dendrites for it because
output layers doesn't have any
         network.Layers(i).Neurons(j).Bias = GetRand
         network.Layers(i).Neurons(j).DendriteCount = ArrayOfLayers(i - 1)
         ReDim
network.Layers(i).Neurons(j).Dendrites(network.Layers(i).Neurons(j).DendriteCount)
As Dendrite 'Redim the dendrite var
         For k = 1 To ArrayOfLayers(i - 1)
            DoEvents
            network.Layers(i).Neurons(j).Dendrites(k).Weight = GetRand * Sqr(3 /
(ArrayOfLayers(i)))  'Set the weight of each dendrite with small random number
relate to  number of inputs
         Next k
      ElseIf i = 1 Then 'Only init dendrites not bias
         DoEvents 'Do nothing coz it is input layer
      Else
         network.Layers(i).Neurons(j).Bias = GetRand
```

```
        network.Layers(i).Neurons(j).PreviousBiasChanged = 0 'Cause it's initializing
bias
        network.Layers(i).Neurons(j).DendriteCount = ArrayOfLayers(i - 1)
        ReDim
network.Layers(i).Neurons(j).Dendrites(network.Layers(i).Neurons(j).DendriteCount)
As Dendrite 'Redim the dendrite var
        For k = 1 To ArrayOfLayers(i - 1)
          DoEvents
          network.Layers(i).Neurons(j).Dendrites(k).Weight = GetRand * Sqr(3 /
(ArrayOfLayers(i)))  'Set the weight of each dendrite with small random number
relate to  number of inputs
          network.Layers(i).Neurons(j).Dendrites(k).PreviousWeightChanged = 0
'Cause It's initalizing dentrite
        Next k
      End If
   Next j
Next i
CreateNet = 1
End Function


Function Run(ArrayOfInputs As Variant) As Variant 'It returns the output inf form of
array
Dim i, j, k As Integer
If UBound(ArrayOfInputs) <> network.Layers(1).NeuronCount Then
   Run = 0
   Exit Function
End If
For i = 1 To network.LayerCount
DoEvents
   For j = 1 To network.Layers(i).NeuronCount
   DoEvents
      If i = 1 Then
        network.Layers(i).Neurons(j).Value = ArrayOfInputs(j) 'Set the value of input
layer
      Else
        network.Layers(i).Neurons(j).Value = 0 'First set the value to zero
        For k = 1 To network.Layers(i - 1).NeuronCount
          DoEvents
          network.Layers(i).Neurons(j).Value = network.Layers(i).Neurons(j).Value
+ network.Layers(i - 1).Neurons(k).Value *
network.Layers(i).Neurons(j).Dendrites(k).Weight 'Calculating the value
        Next k
      network.Layers(i).Neurons(j).Value =
Activation(network.Layers(i).Neurons(j).Value + network.Layers(i).Neurons(j).Bias)
'Calculating the real value of neuron
```

```
      'Network.Layers(i).Neurons(j).Value = tanh(Network.Layers(i).Neurons(j).Value
+ Network.Layers(i).Neurons(j).Bias) 'Calculating the real value of neuron
      End If
   Next j
Next i
ReDim OutputResult(network.Layers(network.LayerCount).NeuronCount) As Double
For i = 1 To (network.Layers(network.LayerCount).NeuronCount)
   DoEvents
   OutputResult(i) = (network.Layers(network.LayerCount).Neurons(i).Value) 'The
array of output result
Next i
Run = OutputResult
End Function

'Back Prop. Learning Rule With GetMostActivation function
Function BackPropTrain1(inputdata As Variant, outputdata As Variant) As Integer
Dim i, j, k As Integer
Dim WeightChanged As Double
Dim Result() As Double
'Check if correct amount of given input
If UBound(inputdata) <> network.Layers(1).NeuronCount Then
   BackPropTrain2 = 0
   Exit Function
End If
'Check if correct amount of given output
If UBound(outputdata) <> network.Layers(network.LayerCount).NeuronCount Then
   BackPropTrain2 = 0
   Exit Function
End If

'Calculate values of all neurons and set the input
Call Run(inputdata)

'Get Network errror sub-process
ReDim network.Err(1)
ReDim Result(UBound(outputdata)) As Double
For i = 1 To UBound(outputdata)
Result(i) = network.Layers(network.LayerCount).Neurons(i).Value
Next
'Error of network for most active node as chosen signal
network.Err(1) = ((GetMostActivation(outputdata) - GetMostActivation(Result)) /
UBound(outputdata))

'Calculate delta's
For i = 1 To network.Layers(network.LayerCount).NeuronCount
DoEvents
```

```
   network.Layers(network.LayerCount).Neurons(i).Delta =
network.Layers(network.LayerCount).Neurons(i).Value * (1 -
network.Layers(network.LayerCount).Neurons(i).Value) * (outputdata(i) -
network.Layers(network.LayerCount).Neurons(i).Value) 'Deltas of Output layer
   For j = network.LayerCount - 1 To 2 Step -1
   DoEvents
      For k = 1 To network.Layers(j).NeuronCount
      DoEvents
         network.Layers(j).Neurons(k).Delta = network.Layers(j).Neurons(k).Value *
(1 - network.Layers(j).Neurons(k).Value) * network.Layers(j +
1).Neurons(i).Dendrites(k).Weight * network.Layers(j + 1).Neurons(i).Delta 'Deltas
of Hidden Layers
      Next k
   Next j
Next i


For i = network.LayerCount To 2 Step -1
DoEvents
   For j = 1 To network.Layers(i).NeuronCount
   DoEvents
      'BiasChanged = network.LearningRate * 1 * network.Layers(I).Neurons(j).Delta
      'network.Layers(I).Neurons(j).Bias = network.Layers(I).Neurons(j).Bias +
BiasChanged + (network.MomentumConst *
network.Layers(I).Neurons(j).PreviousBiasChanged) 'Calculate new bias with
momentum
      network.Layers(i).Neurons(j).Bias = network.Layers(i).Neurons(j).Bias +
(network.LearningRate * 1 * network.Layers(i).Neurons(j).Delta)  'Calculate new bias
      For k = 1 To network.Layers(i).Neurons(j).DendriteCount
      DoEvents
         'network.Layers(i).Neurons(j).Dendrites(k).Weight =
network.Layers(i).Neurons(j).Dendrites(k).Weight + (network.LearningRate *
network.Layers(i - 1).Neurons(k).Value * network.Layers(i).Neurons(j).Delta)
'Calculate new weights without momentum

         WeightChanged = network.LearningRate * network.Layers(i -
1).Neurons(k).Value * network.Layers(i).Neurons(j).Delta +
(network.MomentumConst *
network.Layers(i).Neurons(j).Dendrites(k).PreviousWeightChanged)  'Calculate new
weights with momentum
         network.Layers(i).Neurons(j).Dendrites(k).Weight =
network.Layers(i).Neurons(j).Dendrites(k).Weight + WeightChanged
         network.Layers(i).Neurons(j).Dendrites(k).PreviousWeightChanged =
WeightChanged 'Keep weight change
      Next k
   Next j
Next i
```

```
BackPropTrain2 = 1
End Function
'Back Prop. Learning Rule With Chebyshev Norm
Function BackPropTrain2(inputdata As Variant, outputdata As Variant) As Integer
Dim i, j, k As Integer
Dim WeightChanged As Double
If UBound(inputdata) <> network.Layers(1).NeuronCount Then 'Check if correct
amount of input is given
   BackPropTrain3 = 0
   Exit Function
End If
If UBound(outputdata) <> network.Layers(network.LayerCount).NeuronCount Then
'Check if correct amount of output is given
   BackPropTrain3 = 0
   Exit Function
End If
Call Run(inputdata) 'Calculate values of all neurons and set the input
'Calculate values of all neurons and set the input
Call Run(inputdata)

'Get Network errror sub-process
ReDim network.Err(1)
ReDim Result(UBound(outputdata)) As Double
For i = 1 To UBound(outputdata)
Result(i) = network.Layers(network.LayerCount).Neurons(i).Value
Next
'Error of network for most active node as chosen signal
network.Err(1) = ((GetMostActivation(outputdata) - GetMostActivation(Result)) /
UBound(outputdata))

'Calculate delta's with Chebyshev Norm
For i = 1 To network.Layers(network.LayerCount).NeuronCount
DoEvents
   network.Layers(network.LayerCount).Neurons(i).Delta =
network.Layers(network.LayerCount).Neurons(i).Value * (1 -
network.Layers(network.LayerCount).Neurons(i).Value) * Signum(outputdata(i) -
network.Layers(network.LayerCount).Neurons(i).Value) * Sup(i,
GetMostActivation(outputdata))  'Deltas of Output layer
   For j = network.LayerCount - 1 To 2 Step -1
   DoEvents
      For k = 1 To network.Layers(j).NeuronCount
      DoEvents
         network.Layers(j).Neurons(k).Delta = network.Layers(j).Neurons(k).Value *
(1 - network.Layers(j).Neurons(k).Value) * network.Layers(j +
1).Neurons(i).Dendrites(k).Weight * network.Layers(j + 1).Neurons(i).Delta 'Deltas
of Hidden Layers
      Next k
```

```
   Next j
Next i


For i = network.LayerCount To 2 Step -1
DoEvents
   For j = 1 To network.Layers(i).NeuronCount
   DoEvents
      network.Layers(i).Neurons(j).Bias = network.Layers(i).Neurons(j).Bias +
(network.LearningRate * 1 * network.Layers(i).Neurons(j).Delta)  'Calculate new bias
      For k = 1 To network.Layers(i).Neurons(j).DendriteCount
      DoEvents
         WeightChanged = network.LearningRate * network.Layers(i -
1).Neurons(k).Value * network.Layers(i).Neurons(j).Delta +
(network.MomentumConst *
network.Layers(i).Neurons(j).Dendrites(k).PreviousWeightChanged)  'Calculate new
weights with momentum
         network.Layers(i).Neurons(j).Dendrites(k).Weight =
network.Layers(i).Neurons(j).Dendrites(k).Weight + WeightChanged
         network.Layers(i).Neurons(j).Dendrites(k).PreviousWeightChanged =
WeightChanged 'Keep weight change
      Next k
   Next j
Next i
BackPropTrain3 = 1
End Function
Function NetworkError() As Variant
NetworkError = network.Err
End Function
Function tanh(x As Double) As Double
tanh = (Exp(x) - Exp(-x)) / (Exp(x) + Exp(-x))
End Function

Function Activation(Value As Double) As Double
'To crunch a number between 0 and 1
   Activation = (1 / (1 + Exp(Value * -1))) 'Sigmoid
End Function
'Hard limniter (Threshold function)
Function Signum(f As Double) As Integer
If f < 0 Then
   Signum = -1
Else
   Signum = 1
End If
End Function
'Sup fucntion
Function Sup(ByVal i As Integer, j As Integer) As Integer
```

```
If i = j Then
   Sup = 1
Else
   Sup = 0
End If
End Function

Function GetRand() As Double 'Produces a number between -1 and 1
Randomize
GetRand = 2 - (1 + Rnd + Rnd)
'GetRand = Rnd
End Function

Sub EraseNetwork()
Erase network.Layers
End Sub

Function SaveNet(FilePath As String) As Integer ' 1 = successful, 0 =unsucessful
Dim i, j, k As Integer
Open FilePath For Output As #1
Print #1, "START Learning Rate"
Print #1, network.LearningRate
Print #1, "END Learning Rate"
Print #1, "START Layer Count"
Print #1, network.LayerCount
Print #1, "END Layer Count"
Print #1, "START Input Layer Neuron Count"
Print #1, network.Layers(1).NeuronCount
Print #1, "END Input Layer Neuron Count"
For i = 2 To network.LayerCount
   Print #1, "START Next Layer"
   Print #1, "START Neuron Count"
   Print #1, network.Layers(i).NeuronCount
   Print #1, "END Neuron Count"
   For j = 1 To network.Layers(i).NeuronCount
      Print #1, "START Neuron"
      Print #1, "START Bias"
      Print #1, network.Layers(i).Neurons(j).Bias
      Print #1, "END Bias"
      Print #1, "START Dendrites"
      For k = 1 To network.Layers(i).Neurons(j).DendriteCount
         Print #1, network.Layers(i).Neurons(j).Dendrites(k).Weight
      Next k
      Print #1, "END Dendrites"
      Print #1, "END Neuron"
   Next j
   Print #1, "END Layer"
```

```
Next i
Close #1
SaveNet = 1
End Function

Function LoadNet(FilePath As String) As Integer ' 1 = successful, 0 =unsucessful
Dim Data, DataMain As String
Dim LayerTrack, NeuronTrack As Long 'The variable which tracks the current layer
and current neuron
Dim i As Long
If FileExists(FilePath) = 0 Then
   LoadNet = 0 'File doest not exists
   Exit Function
End If
Open FilePath For Input As #1
Do While Not EOF(1)
   DoEvents
   Line Input #1, Data
   Select Case Data
      Case "START Learning Rate":
         Line Input #1, DataMain
         network.LearningRate = CDbl(DataMain)
      Case "START Layer Count":
         Line Input #1, DataMain
         network.LayerCount = CLng(DataMain)
         ReDim network.Layers(network.LayerCount) As Layer
      Case "START Input Layer Neuron Count": 'Input layer
         LayerTrack = 1
         Line Input #1, DataMain
         network.Layers(1).NeuronCount = CLng(DataMain)
         ReDim network.Layers(1).Neurons(network.Layers(1).NeuronCount) As
Neuron
      Case "START Neuron Count":
         LayerTrack = LayerTrack + 1
         Line Input #1, DataMain
         network.Layers(LayerTrack).NeuronCount = CLng(DataMain)
         ReDim
network.Layers(LayerTrack).Neurons(network.Layers(LayerTrack).NeuronCount) As
Neuron
      Case "START Bias":
         NeuronTrack = NeuronTrack + 1
         Line Input #1, DataMain
         network.Layers(LayerTrack).Neurons(NeuronTrack).Bias = CDbl(DataMain)
         network.Layers(LayerTrack).Neurons(NeuronTrack).DendriteCount =
network.Layers(LayerTrack - 1).NeuronCount
```

```
            ReDim
network.Layers(LayerTrack).Neurons(NeuronTrack).Dendrites(network.Layers(Layer
Track).Neurons(NeuronTrack).DendriteCount) As Dendrite
        Case "START Dendrites":
            For i = 1 To
network.Layers(LayerTrack).Neurons(NeuronTrack).DendriteCount 'All the dendrites
                DoEvents
                Line Input #1, DataMain
                network.Layers(LayerTrack).Neurons(NeuronTrack).Dendrites(i).Weight =
CDbl(DataMain)
            Next i
        Case "END Layer":
            NeuronTrack = 0
        Case Else
            DoEvents
    End Select
Loop
Close #1
LayerTrack = 0
NeuronTrack = 0
LoadNet = 1
End Function

Private Function FileExists(ByVal strPathName As String) As Integer
    Dim intFileNum As Integer

    On Error Resume Next

    '
    'Remove any trailing directory separator character
    '
    If Right$(strPathName, 1) = "\" Then
        strPathName = Left$(strPathName, Len(strPathName) - 1)
    End If
    '
    'Attempt to open the file, return value of this function is False
    'if an error occurs on open, True otherwise
    '
    intFileNum = FreeFile
    Open strPathName For Input As intFileNum

    FileExists = IIf(Err, False, True)

    Close intFileNum

    Err = 0
```

```
End Function
'Find the most active node of output
Function GetMostActivation(Result As Variant) As Integer
Dim MostActiveNode As Integer
Dim i As Integer
MostActiveNode = 1
For i = 1 To UBound(Result)
If Result(MostActiveNode) < Result(i) Then
MostActiveNode = i
End If
Next
GetMostActivation = MostActiveNode
End Function
```

ก.3 ซอร์โคด JOYSTICK.BAS

```
Private Const JOY_BUTTON1 As Long = &H1&
Public Const JOY_BUTTON2 As Long = &H2&
Public Const JOY_BUTTON3 As Long = &H4&
Public Const JOY_BUTTON4 As Long = &H8&

Public Const JOYERR_BASE As Long = 160&
Public Const JOYERR_NOERROR As Long = 0&
Public Const JOYERR_NOCANDO As Long = (JOYERR_BASE + 6&)
Public Const JOYERR_PARMS As Long = (JOYERR_BASE + 5&)
Public Const JOYERR_UNPLUGGED As Long = (JOYERR_BASE + 7&)

Public Const MAXPNAMELEN As Long = 32&
Public Const JOYSTICKID1 As Long = 0&
Public Const JOYSTICKID2 As Long = 1&

Public Type JOYINFO
   x As Long
   y As Long
   Z As Long
   Buttons As Long
End Type

Private Type JOYCAPS
```

```
    wMid As Integer
    wPid As Integer
    szPname As String * MAXPNAMELEN
    wXmin As Long
    wXmax As Long
    wYmin As Long
    wYmax As Long
    wZmin As Long
    wZmax As Long
    wNumButtons As Long
    wPeriodMin As Long
    wPeriodMax As Long
End Type
Declare Function joyGetDevCaps Lib "winmm.dll" Alias "joyGetDevCapsA" (ByVal
id As Long, lpCaps As JOYCAPS, ByVal uSize As Long) As Long
Declare Function joyGetNumDevs Lib "winmm.dll" () As Long
Declare Function joyGetPos Lib "winmm.dll" (ByVal uJoyID As Long, pji As
JOYINFO) As Long

Function GetJoyMax(ByVal joy As Integer, JI As JOYINFO) As Boolean
    Dim jc As JOYCAPS
    If joyGetDevCaps(joy, jc, Len(jc)) <> JOYERR_NOERROR Then
        GetJoyMax = False
    Else
        JI.x = jc.wXmax
        JI.y = jc.wYmax
        JI.Z = jc.wZmax
        JI.Buttons = jc.wNumButtons
        GetJoyMax = True
    End If
End Function

Function GetJoyMin(ByVal joy As Integer, JI As JOYINFO) As Boolean
    Dim jc As JOYCAPS
    If joyGetDevCaps(joy, jc, Len(jc)) <> JOYERR_NOERROR Then
        GetJoyMin = False
    Else
        JI.x = jc.wXmin
        JI.y = jc.wYmin
        JI.Z = jc.wZmin
        JI.Buttons = jc.wNumButtons
        GetJoyMin = True
    End If
End Function

Function GetJoystick(ByVal joy As Integer, JI As JOYINFO) As Boolean
    If joyGetPos(joy, JI) <> JOYERR_NOERROR Then
```

```
    GetJoystick = False
  Else
    GetJoystick = True
  End If
End Function

Function IsJoyPresent(Optional IsConnected As Variant) As Long
  Dim ic As Boolean
  Dim i As Long
  Dim j As Long
  Dim ret As Long
  Dim JI As JOYINFO
  ic = IIf(IsMissing(IsConnected), True, CBool(IsConnected))
  i = joyGetNumDevs
  If ic Then
    j = 0
    Do While i > 0
      i = i - 1
      If joyGetPos(i, JI) = JOYERR_NOERROR Then
        j = j + 1
      End If
    Loop
    IsJoyPresent = j
  Else
    IsJoyPresent = i
  End If
End Function
```

ก.4 ซอร์สโคด cDIBSection.CLS

```
Option Explicit

Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
  lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)

Private Type SAFEARRAYBOUND
  cElements As Long
  lLbound As Long
End Type

Private Type SAFEARRAY2D
  cDims As Integer
  fFeatures As Integer
  cbElements As Long
  cLocks As Long
```

```
    pvData As Long
    Bounds(0 To 1) As SAFEARRAYBOUND
End Type
Private Declare Function VarPtrArray Lib "msvbvm50.dll" Alias "VarPtr" (Ptr() As
Any) As Long

Private Type RGBQUAD
    rgbBlue As Byte
    rgbGreen As Byte
    rgbRed As Byte
    rgbReserved As Byte
End Type
Private Type BITMAPINFOHEADER '40 bytes
    biSize As Long
    biWidth As Long
    biHeight As Long
    biPlanes As Integer
    biBitCount As Integer
    biCompression As Long
    biSizeImage As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
    biClrUsed As Long
    biClrImportant As Long
End Type
Private Type BITMAPINFO
    bmiHeader As BITMAPINFOHEADER
    bmiColors As RGBQUAD
End Type
Private Declare Function CreateCompatibleDC Lib "gdi32" (ByVal hdc As Long) As
Long
Private Declare Function GetDC Lib "user32" (ByVal hwnd As Long) As Long
Private Declare Function GetDesktopWindow Lib "user32" () As Long
' Note - this is not the declare in the API viewer - modify lplpVoid to be
' Byref so we get the pointer back:
Private Declare Function CreateDIBSection Lib "gdi32" _
    (ByVal hdc As Long, _
    pBitmapInfo As BITMAPINFO, _
    ByVal un As Long, _
    lplpVoid As Long, _
    ByVal handle As Long, _
    ByVal dw As Long) As Long
```

```
Private Declare Function BitBlt Lib "gdi32" (ByVal hDestDC As Long, ByVal x As
Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal
hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As
Long) As Long
Private Declare Function SelectObject Lib "gdi32" (ByVal hdc As Long, ByVal
hObject As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal hObject As Long) As
Long
Private Declare Function DeleteDC Lib "gdi32" (ByVal hdc As Long) As Long
Private Declare Function LoadImage Lib "user32" Alias "LoadImageA" (ByVal hInst
As Long, ByVal lpsz As String, ByVal un1 As Long, ByVal n1 As Long, ByVal n2
As Long, ByVal un2 As Long) As Long
Private Const BI_RGB = 0&
Private Const BI_RLE4 = 2&
Private Const BI_RLE8 = 1&
Private Const DIB_RGB_COLORS = 0 '  color table in RGBs

Private Type BITMAP
   bmType As Long
   bmWidth As Long
   bmHeight As Long
   bmWidthBytes As Long
   bmPlanes As Integer
   bmBitsPixel As Integer
   bmBits As Long
End Type
Private Declare Function GetObjectAPI Lib "gdi32" Alias "GetObjectA" (ByVal
hObject As Long, ByVal nCount As Long, lpObject As Any) As Long

Private Declare Function CreateCompatibleBitmap Lib "gdi32" (ByVal hdc As Long,
ByVal nWidth As Long, ByVal nHeight As Long) As Long

' Clipboard functions:

Private Const CF_BITMAP = 2
Private Const CF_DIB = 8

' Handle to the current DIBSection:
Private m_hDIb As Long
```

```
' Handle to the old bitmap in the DC, for clear up:
Private m_hBmpOld As Long
' Handle to the Device context holding the DIBSection:
Private m_hDC As Long
' Address of memory pointing to the DIBSection's bits:
Private m_lPtr As Long
' Type containing the Bitmap information:
Private m_tBI As BITMAPINFO

Public Function CreateDIB( _
      ByVal lHDC As Long, _
      ByVal lWidth As Long, _
      ByVal lHeight As Long, _
      ByRef hDib As Long _
   ) As Boolean
   With m_tBI.bmiHeader
      .biSize = Len(m_tBI.bmiHeader)
      .biWidth = lWidth
      .biHeight = lHeight
      .biPlanes = 1
      .biBitCount = 24
      .biCompression = BI_RGB
      .biSizeImage = BytesPerScanLine * .biHeight
   End With
   hDib = CreateDIBSection( _
         lHDC, _
         m_tBI, _
         DIB_RGB_COLORS, _
         m_lPtr, _
         0, 0)
   CreateDIB = (hDib <> 0)
End Function
Public Function CreateFromPicture( _
      ByRef picThis As StdPicture _
   )
Dim lHDC As Long
Dim lhDCDesktop As Long
Dim lhBmpOld As Long
Dim tBMP As BITMAP

   GetObjectAPI picThis.handle, Len(tBMP), tBMP
```

```
    If (Create(tBMP.bmWidth, tBMP.bmHeight)) Then
      lhDCDesktop = GetDC(GetDesktopWindow())
      If (lhDCDesktop <> 0) Then
        lHDC = CreateCompatibleDC(lhDCDesktop)
        DeleteDC lhDCDesktop
        If (lHDC <> 0) Then
          lhBmpOld = SelectObject(lHDC, picThis.handle)
          LoadPictureBlt lHDC
          SelectObject lHDC, lhBmpOld
          DeleteObject lHDC
        End If
      End If
    End If
End Function
Public Function Create( _
    ByVal lWidth As Long, _
    ByVal lHeight As Long _
  ) As Boolean
  ClearUp
  m_hDC = CreateCompatibleDC(0)
  If (m_hDC <> 0) Then
    If (CreateDIB(m_hDC, lWidth, lHeight, m_hDIb)) Then
      m_hBmpOld = SelectObject(m_hDC, m_hDIb)
      Create = True
    Else
      DeleteObject m_hDC
      m_hDC = 0
    End If
  End If
End Function
Public Property Get BytesPerScanLine() As Long
  ' Scans must align on dword boundaries:
  BytesPerScanLine = (m_tBI.bmiHeader.biWidth * 3 + 3) And &HFFFFFFFC
End Property

Public Property Get Width() As Long
  Width = m_tBI.bmiHeader.biWidth
End Property
Public Property Get Height() As Long
  Height = m_tBI.bmiHeader.biHeight
End Property

Public Sub LoadPictureBlt( _
    ByVal lHDC As Long, _
```

```
        Optional ByVal lSrcLeft As Long = 0, _
        Optional ByVal lSrcTop As Long = 0, _
        Optional ByVal lSrcWidth As Long = -1, _
        Optional ByVal lSrcHeight As Long = -1, _
        Optional ByVal eRop As RasterOpConstants = vbSrcCopy _
    )
    If lSrcWidth < 0 Then lSrcWidth = m_tBI.bmiHeader.biWidth
    If lSrcHeight < 0 Then lSrcHeight = m_tBI.bmiHeader.biHeight
    BitBlt m_hDC, 0, 0, lSrcWidth, lSrcHeight, lHDC, lSrcLeft, lSrcTop, eRop
End Sub
Public Sub PaintPicture( _
        ByVal lHDC As Long, _
        Optional ByVal lDestLeft As Long = 0, _
        Optional ByVal lDestTop As Long = 0, _
        Optional ByVal lDestWidth As Long = -1, _
        Optional ByVal lDestHeight As Long = -1, _
        Optional ByVal lSrcLeft As Long = 0, _
        Optional ByVal lSrcTop As Long = 0, _
        Optional ByVal eRop As RasterOpConstants = vbSrcCopy _
    )
    If (lDestWidth < 0) Then lDestWidth = m_tBI.bmiHeader.biWidth
    If (lDestHeight < 0) Then lDestHeight = m_tBI.bmiHeader.biHeight
    BitBlt lHDC, lDestLeft, lDestTop, lDestWidth, lDestHeight, m_hDC, lSrcLeft,
lSrcTop, eRop
End Sub
Public Property Get DIBSectionBitsPtr() As Long
    DIBSectionBitsPtr = m_lPtr
End Property

Public Sub ClearUp()
    If (m_hDC <> 0) Then
        If (m_hDIb <> 0) Then
            SelectObject m_hDC, m_hBmpOld
            DeleteObject m_hDIb
        End If
        DeleteObject m_hDC
```

```
    End If
    m_hDC = 0: m_hDIb = 0: m_hBmpOld = 0: m_lPtr = 0
End Sub

Public Function Resample( _
        ByVal lNewHeight As Long, _
        ByVal lNewWidth As Long _
    ) As cDIBSection
Dim cDib As cDIBSection
    Set cDib = New cDIBSection
    If cDib.Create(lNewWidth, lNewHeight) Then
      If (lNewWidth <> m_tBI.bmiHeader.biWidth) Or (lNewHeight <>
m_tBI.bmiHeader.biHeight) Then
          ' Change in size, do resample:
          ResampleDib cDib
      Else
          ' No size change so just return a copy:
          cDib.LoadPictureBlt m_hDC
      End If
      Set Resample = cDib
    End If
End Function

Private Function ResampleDib(ByRef cdibto As cDIBSection) As Boolean
Dim bDibFrom() As Byte
Dim bDibTo() As Byte

Dim tSAFrom As SAFEARRAY2D
Dim tSAto As SAFEARRAY2D

    ' Get the bits in the from DIB section:
    With tSAFrom
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = m_tBI.bmiHeader.biHeight
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = BytesPerScanLine()
      .pvData = m_lPtr
    End With
    CopyMemory ByVal VarPtrArray(bDibFrom()), VarPtr(tSAFrom), 4
```

```
' Get the bits in the to DIB section:
With tSAto
  .cbElements = 1
  .cDims = 2
  .Bounds(0).lLbound = 0
  .Bounds(0).cElements = cdibto.Height
  .Bounds(1).lLbound = 0
  .Bounds(1).cElements = cdibto.BytesPerScanLine()
  .pvData = cdibto.DIBSectionBitsPtr
End With
CopyMemory ByVal VarPtrArray(bDibTo()), VarPtr(tSAto), 4

Dim xScale As Single
Dim yScale As Single

Dim x As Long, y As Long, xEnd As Long, xOut As Long

Dim fX As Single, fY As Single
Dim ifY As Long, ifX As Long
Dim dX As Single, dy As Single
Dim r As Long, r1 As Single, r2 As Single, r3 As Single, r4 As Single
Dim g As Long, g1 As Single, g2 As Single, g3 As Single, g4 As Single
Dim b As Long, b1 As Single, b2 As Single, b3 As Single, b4 As Single
Dim ir1 As Long, ig1 As Long, ib1 As Long
Dim ir2 As Long, ig2 As Long, ib2 As Long

xScale = (Width - 1) / cdibto.Width
yScale = (Height - 1) / cdibto.Height
xEnd = cdibto.Width - 1

For y = 0 To cdibto.Height - 1

  fY = y * yScale
  ifY = Int(fY)
  dy = fY - ifY
```

```
For x = 0 To xEnd
    fX = x * xScale
    ifX = Int(fX)
    dX = fX - ifX

    ifX = ifX * 3
    ' Interpolate using the four nearest pixels in the source
    b1 = bDibFrom(ifX, ifY): g1 = bDibFrom(ifX + 1, ifY): r1 = bDibFrom(ifX + 2,
ifY)
    b2 = bDibFrom(ifX + 3, ifY): g2 = bDibFrom(ifX + 4, ifY): r2 = bDibFrom(ifX
+ 5, ifY)
    b3 = bDibFrom(ifX, ifY + 1): g3 = bDibFrom(ifX + 1, ifY + 1): r3 =
bDibFrom(ifX + 2, ifY + 1)
    b4 = bDibFrom(ifX + 3, ifY + 1): g4 = bDibFrom(ifX + 4, ifY + 1): r4 =
bDibFrom(ifX + 5, ifY + 1)

    ' Interplate in x direction:
    ir1 = r1 * (1 - dy) + r3 * dy: ig1 = g1 * (1 - dy) + g3 * dy: ib1 = b1 * (1 - dy) + b3 *
dy
    ir2 = r2 * (1 - dy) + r4 * dy: ig2 = g2 * (1 - dy) + g4 * dy: ib2 = b2 * (1 - dy) + b4 *
dy
    ' Interpolate in y:
    r = ir1 * (1 - dX) + ir2 * dX: g = ig1 * (1 - dX) + ig2 * dX: b = ib1 * (1 - dX) +
ib2 * dX

    ' Set output:
    If (r < 0) Then r = 0
    If (r > 255) Then r = 255
    If (g < 0) Then g = 0
    If (g > 255) Then g = 255
    If (b < 0) Then b = 0
    If (b > 255) Then
      b = 255
    End If
    xOut = x * 3
    bDibTo(xOut, y) = b
```

```
            bDibTo(xOut + 1, y) = g
            bDibTo(xOut + 2, y) = r

        Next x

    Next y

    ' Clear the temporary array descriptor
    CopyMemory ByVal VarPtrArray(bDibFrom), 0&, 4
    CopyMemory ByVal VarPtrArray(bDibTo), 0&, 4

End Function


Private Sub Class_Terminate()
    ClearUp
End Sub
```

ก.5 cImageProcess.CLS

```
Option Explicit
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
    lpvDest As Any, lpvSource As Any, ByVal cbCopy As Long)

Private Type SAFEARRAYBOUND
    cElements As Long
    lLbound As Long
End Type
Private Type SAFEARRAY2D
    cDims As Integer
    fFeatures As Integer
    cbElements As Long
    cLocks As Long
    pvData As Long
    Bounds(0 To 1) As SAFEARRAYBOUND
End Type
Private Declare Function VarPtrArray Lib "msvbvm50.dll" Alias "VarPtr" (Ptr() As
Any) As Long


Public Event InitProgress(ByVal lMax As Long)
```

```
Public Sub GrayScale( _
      ByRef cTo As cDIBSection _
   )
' Gray scale using standard intensity components.
' see http://www.dcs.ed.ac.uk/~mxr/gfx/faqs/colourspace.faq for details.
'
Dim bDib() As Byte
Dim x As Long, y As Long
Dim xMax As Long, yMax As Long
Dim bContinue As Boolean
Dim lB As Long, lG As Long, lR As Long
Dim lGray As Long
Dim lTIme As Long
Dim tSA As SAFEARRAY2D


   ' have the local matrix point to bitmap pixels
   With tSA
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = cTo.Height
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = cTo.BytesPerScanLine
      .pvData = cTo.DIBSectionBitsPtr
   End With
   CopyMemory ByVal VarPtrArray(bDib), VarPtr(tSA), 4

   yMax = cTo.Height - 1
   xMax = cTo.Width - 1

   For x = 0 To (xMax * 3) Step 3
      For y = 0 To yMax
         lB = bDib(x, y)
         lG = bDib(x + 1, y)
```

```
        lR = bDib(x + 2, y)

        'But now all people *should* use the most accurate, it means ITU standard:
        lGray = (222 * lR + 707 * lG + 71 * lB) / 1000

        bDib(x, y) = lGray
        bDib(x + 1, y) = lGray
        bDib(x + 2, y) = lGray
      Next y
    Next x

    ' clear the temporary array descriptor
    ' without destroying the local temporary array
    CopyMemory ByVal VarPtrArray(bDib), 0&, 4


End Sub
```

ภาคผนวก ข

ซอร์สโค๊ดโปรแกรมแกน

ข.1 ซอร์สโคด **TrainRecord**

```
Option Explicit

'LPT Port Interface DLL
Private Declare Sub Out Lib "inpout32.dll" Alias "Out32" (ByVal PortAddress As
Integer, ByVal Value As Integer)

'Use ImageProcessDIB Class Module
Private WithEvents m_cImage As cImageProcessDIB
Private m_cDib As New cDIBSection            'For Manage Picture
Private m_cDibBuffer As New cDIBSection      'Picture Data Buffer

'Use CCRP Timer Instead of VB Timer
Private WithEvents Timer1 As ccrpTimer
Private WithEvents Timer3 As ccrpTimer

Private c As Double      'Control Direction Var
Private pwrite As Integer   'Port Adress Var
Private lwndC As Long       'Handle to the Capture Windows
Const iDevice = 0          'Capture Devive No.0

Private Sub Form_Load()
Dim CapDevName As String * 100
Dim CapDevVer As String * 100

'Set up Timer
'Timer1 for Control via Joystick
Set Timer1 = New ccrpTimer
With Timer1
   .Stats.Frequency = 33
   .Interval = 33
   .Enabled = True
End With

'Timer2 for GrabFrame by Capture Device
```

```
Timer2.Enabled = False

'Timer3 for Counter Clock
Set Timer3 = New ccrpTimer
With Timer3
   .Stats.Frequency = 33
   .Interval = 33
   .Enabled = False
End With

'Image Processing Class Module for Process Grabed Image
Set m_cImage = New cImageProcessDIB

'Prepare Capture Device
capGetDriverDescriptionA 0, CapDevName, 100, CapDevVer, 100
lwndC = capCreateCaptureWindowA(CapDevName, WS_CHILD Or WS_VISIBLE,
0, 0, 189, 149, Picture1.hwnd, 0)

'Prepare LPT Port Interface
pwrite = &H378
Out pwrite, &H0

'Prepare Com Port Interface
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"

'Neutral Status
Timer1.Enabled = False
MSComm1.InputMode = comInputModeText
Shape1.BackColor = vbWhite
End Sub

Private Sub HWConnect_Click()
Timer1.Enabled = True
MSComm1.PortOpen = True
Shape1.BackColor = vbRed
'Connect to Device
SendMessage lwndC, WM_CAP_DRIVER_CONNECT, iDevice, 0
'Set Scaleable Preview
SendMessage lwndC, WM_CAP_SET_SCALE, True, 0
'Set Preview Rate
SendMessage lwndC, WM_CAP_SET_PREVIEWRATE, 66, 0
'Start Preview
SendMessage lwndC, WM_CAP_SET_PREVIEW, True, 0
End Sub

Private Sub HWDisconnect_Click()
```

```
MSComm1.PortOpen = False
Timer1.Enabled = False
Shape1.BackColor = vbWhite
Text6.Text = ""
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text15.Text = ""
'Disconnect Capture Device
SendMessage lwndC, WM_CAP_DRIVER_DISCONNECT, iDevice, 0
DestroyWindow Picture1
End Sub
'Start Capture
Private Sub Start_Click()
Timer2.Enabled = True
End Sub
'Stop Capture
Private Sub Stop_Click()
Timer3.Enabled = False
Timer2.Enabled = False
End Sub
' Joystick Timmer
Private Sub Timer1_Timer(ByVal Milliseconds As Long)
Dim JInfo As JOYINFO

'Recieve Invitation Message
On Error Resume Next
   MSComm1.DTREnable = False
   MSComm1.DTREnable = True
   MSComm1.InputLen = 0
Text6.Text = Text6.Text & MSComm1.Input

GetJoystick JOYSTICKID1, JInfo
   'Steering Direction
   Text1.Text = JInfo.x
   Text3.Text = Int(Int(JInfo.x / 257) / 8.5)
   c = Val(Text3.Text)
   Text2.Text = "&H" & Hex$(Text3.Text)
   Text4.Text = Chr(Text2.Text)
   MSComm1.Output = Chr(Text2.Text)

   'Forward or Pause
   If JInfo.Buttons = 8 Then
      Out pwrite, &H4
      Text15.Text = "Forward"
      Shape2.BackColor = vbRed
```

```
    Else
        Out pwrite, &H0
        Text15.Text = "Pause"
        Shape2.BackColor = vbWhite
    End If
End Sub
'Image Processor Timer
Private Sub Timer2_Timer()

'Enable Timer3 for Counter Clock
Timer3.Enabled = True

'Grab Frame Timmer
SendMessage lwndC, WM_CAP_GRAB_FRAME_NOSTOP, 0, 0
SendMessage lwndC, WM_CAP_EDIT_COPY, 0, 0

'Process Grabed Image
m_cDib.CreateFromPicture Clipboard.GetData
m_cDibBuffer.Create m_cDib.Width, m_cDib.Height
GrayScale
Resample 30, 32
'Save Processed Image

If c < 10 Then
    SavePicture Picture2.Image, App.Path & "/" & Label12.Caption & "(0" &
Str(Text3.Text) & ")" & ".bmp"
Else
    SavePicture Picture2.Image, App.Path & "/" & Label12.Caption & "(" &
Str(Text3.Text) & ")" & ".bmp"
End If
End Sub
'Time Counter for Refernce
Private Sub Timer3_Timer(ByVal Milliseconds As Long)
Static Ticks As Long

Ticks = Ticks + (Milliseconds / 100)
Label12.Caption = Format(Ticks, "#,##0")
End Sub
'For Tuning Capture Device
Private Sub TuneCap_Click()
SendMessage lwndC, WM_CAP_DLG_VIDEOSOURCE, 0, 0
End Sub
'Rebuilt Processed Picture
Private Sub Render()
Picture2.Width = m_cDib.Width * Screen.TwipsPerPixelX
Picture2.Height = m_cDib.Height * Screen.TwipsPerPixelY
m_cDib.PaintPicture Picture2.hdc
```

```
End Sub
'For Process Grabed Picture As Gray Scale
Private Sub GrayScale()
m_cImage.GrayScale m_cDib
'Render
End Sub
'Resizing Picture
Private Sub Resample(ByVal lW As Long, ByVal lH As Long)
Dim cDib As New cDIBSection
   If (lW <> m_cDib.Width) Or (lH <> m_cDib.Height) Then
      Set cDib = m_cDib.Resample(lH, lW)
      Set m_cDib = cDib
      m_cDibBuffer.Create m_cDib.Width, m_cDib.Height
      Render
   End If
End Sub
```

ข.2 ซอร์สโคด ExtracData

```
Option Explicit
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (lpvDest
As Any, lpvSource As Any, ByVal cbCopy As Long)
Private Declare Function VarPtrArray Lib "msvbvm50.dll" Alias "VarPtr" (Ptr() As
Any) As Long
Private Type SAFEARRAYBOUND
   cElements As Long
   lLbound As Long
End Type
Private Type SAFEARRAY2D
   cDims As Integer
   fFeatures As Integer
   cbElements As Long
   cLocks As Long
   pvData As Long
   Bounds(0 To 1) As SAFEARRAYBOUND
End Type

Private m_cDib As New cDIBSection
Private m_cDibBuffer As New cDIBSection
Private BitsByte() As Byte

Private Sub Command1_Click()
CommonDialog1.Filter = "Training Picture | *.bmp"
CommonDialog1.ShowOpen
If CommonDialog1.filename <> "" Then
Image1.Stretch = True
```

```
Image1.Picture = LoadPicture(CommonDialog1.filename)
Picture1 = LoadPicture(CommonDialog1.filename)
m_cDib.CreateFromPicture Picture1
m_cDibBuffer.Create m_cDib.Width, m_cDib.Height
Text1.Text = Right(CommonDialog1.filename, 7)
Text1.Text = Left(Text1.Text, 2)
Text2.Text = CommonDialog1.FileTitle
End If
End Sub

Private Sub Command2_Click()
Dim GetBits(2880) As Byte 'Picture Represented Bits Data
Dim GetData(960) As Byte 'Get the inputs data
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim x As Integer
Dim no_white As Integer
Dim left_count As Integer
Dim right_count As Integer
Dim target As Byte
'Dim TempBB() As Byte
Dim output(30) As Byte
Dim filename As String

For i = 1 To 30
   output(i) = 0
Next
output(Val(Text1.Text)) = 1

k = 1
i = 0
j = 0
x = 1
GetbitsAsByteArray
   For j = 0 To 31
   For i = 0 To 89
      GetBits(k) = BitsByte(i, j)
      k = k + 1
   Next
   Next
   For i = 1 To 960
      GetData(i) = GetBits(x)
      x = x + 3
   Next
```

```
Text1.Text = target 'Show corrected target result
output(Val(Text1.Text)) = 1 'Set activation node
Text3.Text = Text3.Text + 1 'Set file Serial No.

'Set Serial in 4 digits of inteber
If Val(Text3.Text) < 10 Then 'for less than 10 serials
   Text3.Text = "00" & Text3.Text
End If

If (Val(Text3.Text) >= 10) And (Val(Text3.Text) < 100) Then 'for more than 10 but
less than 100
   Text3.Text = "0" & Text3.Text
End If

If (Val(Text3.Text) >= 100) And Val(Val(Text3.Text) < 1000) Then 'for more than
100 but leszs than 1000
   Text3.Text = "0" & Text3.Text
End If

'Save training data  and Kwon target into NND file
filename = App.Path & "\" & Text3.Text & ".nnd"
Open filename For Output As #1
   For i = 1 To 960
      Print #1, (GetData(i) / 255)
   Next
   For i = 1 To 30
      Print #1, Gaussian(i, target)
   Next
Close #1

End Sub

Private Sub GetbitsAsByteArray()
Dim tSA As SAFEARRAY2D
Dim bDib() As Byte
'Get The bits in DIB Section
   With tSA
      .cbElements = 1
      .cDims = 2
      .Bounds(0).lLbound = 0
      .Bounds(0).cElements = m_cDib.Height
      .Bounds(1).lLbound = 0
      .Bounds(1).cElements = m_cDib.BytesPerScanLine
      .pvData = m_cDib.DIBSectionBitsPtr
   End With
   CopyMemory ByVal VarPtrArray(bDib()), VarPtr(tSA), 4
   BitsByte = bDib
```

```
End Sub

Private Function Gaussian(i As Integer, target As Byte) As Double
Gaussian = Exp(-0.1 * ((i - target) ^ 2))
End Function
```

ข.3 ซอร์สโคด ANNTrainer

```
Option Base 1
Option Explicit
Private Type Data
  Data() As Double
  Target() As Double
End Type
'Train variable
Dim TrainData As Data
'Test variable
Dim TestData As Data

'ANN here
Dim LearningRate As Double
Dim Train As Boolean
Const PicH = 32
Const PicW = 30
Const HiddenNode = 5
Const OutNode = 30
Const theta = 0.2 'Momentum constant
Dim Inp(PicH * PicW) As Double 'Input Value
Dim Hidden(HiddenNode) As Double 'Hidden node result
Dim Out(OutNode) As Double 'Output result
Dim HideWeight(PicH * PicW, HiddenNode) As Double ' Hidden Layer nodes's
weights
Dim OutWeight(HiddenNode, OutNode) As Double 'Output Layer nodes's weights
Dim DelHideWeight(PicH * PicW, HiddenNode) As Double 'Weighting Change for
Momentum
Dim DelOutWeight(HiddenNode, OutNode) As Double
Dim OutBeta(OutNode) As Double
Dim HideBeta(HiddenNode) As Double
'Load TrainData to Input Array
Function LoadData(FilePath As String) As Integer
Dim i As Integer
```

```
Dim Data() As String
Dim Target() As String
ReDim TrainData.Data(PicH * PicW)
ReDim TrainData.Target(OutNode)
ReDim Data(PicW * PicH)
ReDim Target(OutNode)
Open FilePath For Input As #1
i = 1
For i = 1 To (PicW * PicH)
   Line Input #1, Data(i)
Next
For i = ((PicW * PicH) + 1) To ((PicW * PicH) + OutNode)
   Line Input #1, Target(i - (PicW * PicH))
Next
Close #1
For i = 1 To (PicW * PicH)
TrainData.Data(i) = Val(Data(i))
Next
For i = 1 To OutNode
TrainData.Target(i) = Val(Target(i))
Next

End Function
'Load TrainData to Input Array
Function LoadTest(FilePath As String) As Integer
Dim i As Integer
Dim Data() As String
Dim Target() As String
ReDim TestData.Data(PicH * PicW)
ReDim TestData.Target(OutNode)
ReDim Data(PicH * PicW)
ReDim Target(OutNode)

Open FilePath For Input As #1
i = 1
For i = 1 To (PicH * PicW)
   Line Input #1, Data(i)
Next
For i = ((PicH * PicW) + 1) To ((PicH * PicW) + OutNode)
```

```
   Line Input #1, Target(i - (PicH * PicW))
Next
Close #1
For i = 1 To (PicH * PicW)
TestData.Data(i) = Val(Data(i))
Next
For i = 1 To OutNode
TestData.Target(i) = Val(Target(i))
Next

End Function
'Create Network
Private Sub Command2_Click()

LearningRate = InputBox("How many LearningRate?", , "1.0")
Text6.Text = LearningRate
InitNN
Command4.Enabled = True

End Sub
'Train ANN here!
Private Sub Command4_Click()
Dim i As Long
Dim Train_Err As Double
Dim Test_Err As Double
Dim Graph As Form2
Set Graph = New Form2
Graph.Show

'Train until reach an assigned epoch
For i = 1 To Val(Text3.Text)
'Open file processes
'Call a traing data file
Gen_Train_Name
'If files did not existed, then try another random
Do While Not FileExists(App.Path & "\" & Text1.Text & ".nnd")
Gen_Train_Name
Text2.Text = FileExists(App.Path & "\" & Text1.Text & ".nnd")
Loop
```

```
'Now load Training Data
LoadData (App.Path & "\" & Text1.Text & ".nnd")
RunNN
TrainNN
Train_Err = (GetMostActivation(TrainData.Target) - GetMostActivation(Out)) /
OutNode

'Adaptive Learning rate
If (i Mod 2000) = 0 Then
LearningRate = LearningRate * 0.5
Text6.Text = LearningRate
End If
'Plot Average Sum Sqaure Error
  If i < 501 Then
    Graph.AddData1 1, i, Abs(Train_Err)
  Else
     Graph.AddData2 1, i, Abs(Train_Err)
  End If

'Complete epoch here
Text3.Text = Val(Text3.Text) - 1
Next

'When Finish, Ask for save network weights file
MsgBox "Training complete"

CommonDialog1.FileName = ""
CommonDialog1.Filter = "Trained ANN Weights|*.net"
CommonDialog1.ShowSave
If CommonDialog1.FileName <> "" Then
Call SaveNet(CommonDialog1.FileName)
End If

End Sub
'For Load a trained NN weights file
Private Sub Command5_Click()
CommonDialog1.Filter = " Network Weights Data File| *.net"
```

```
CommonDialog1.ShowOpen
Call LoadNet(CommonDialog1.FileName)
End Sub
'When Form load
Private Sub Form_Load()
Train = True
Check1.Value = Checked
Check1.Enabled = False
Option3.Value = True
End Sub
'Train Only
Private Sub Option1_Click()
Check1.Value = Unchecked
Check1.Enabled = False
Text1.Enabled = True
'Text1.Text = "200"
Train = True
End Sub
'Test Only
Private Sub Option2_Click()
Train = False
Text1.Text = ""
Text1.Enabled = False
Command6.Enabled = True
Check1.Value = Checked
End Sub
'Test/Train
Private Sub Option3_Click()
Check1.Value = Checked
Text1.Enabled = True
'Text1.Text = "200"
Train = True
End Sub
'To Generate a randomized ing event by using a randomized picture
Private Sub Gen_Train_Name()
Dim Digit(3) As Byte
```

```
'Randomize
Digit(1) = Int(Rnd * 10)
Digit(2) = Int(Rnd * 10)
Digit(3) = Int(Rnd * 10)
'Digit(4) = Int(Rnd * 10)
Text1.Text = Digit(1) & Digit(2) & Digit(3) '& Digit(4)
End Sub
'Choose a query test file randonly
Private Sub Gen_Test_Name()
Dim Digit(3)
Digit(1) = Int(Rnd * 10)
Digit(2) = Int(Rnd * 10)
Digit(3) = Int(Rnd * 10)
'Digit(4) = Int(Rnd * 10)
Text4.Text = Digit(1) & Digit(2) & Digit(3) '& Digit(4)
End Sub
'Find the most active node of output
Function GetMostActivation(Result As Variant) As Integer
Dim MostActiveNode As Integer
Dim i As Integer
MostActiveNode = 1
For i = 1 To UBound(Result)
If Result(MostActiveNode) < Result(i) Then
MostActiveNode = i
End If
Next
GetMostActivation = MostActiveNode
End Function
Private Function FileExists(ByVal strPathName As String) As Integer
Dim intFileNum As Integer

On Error Resume Next
'Remove any trailing directory separator character
   If Right$(strPathName, 1) = "\" Then
      strPathName = Left$(strPathName, Len(strPathName) - 1)
   End If

'Attempt to open the file, return value of this function is False
'if an error occurs on open, True otherwise
```

```
    intFileNum = FreeFile
    Open strPathName For Input As intFileNum
    FileExists = IIf(Err, False, True)
    Close intFileNum

    Err = 0
End Function
Function Sigmoid(Value As Double) As Double
'To crunch a number between 0 and 1
    Sigmoid = (1 / (1 + Exp(Value * -1))) 'Unipolar Sigmoid
End Function
 Function GetRand() As Double 'Produces a number between -1 and 1
Randomize
GetRand = 2 - (1 + Rnd + Rnd)
'GetRand = Rnd
End Function
'Initiate Neural Network Weights
Private Sub InitNN()
Dim i As Integer
Dim j As Integer

'Apply weights to all hidden nodes randomly
For i = 1 To HiddenNode
  For j = 1 To (PicH * PicW)
     HideWeight(j, i) = GetRand
   Next
Next

'Apply weights to all output nodes randomly
For i = 1 To OutNode
   For j = 1 To HiddenNode
      OutWeight(j, i) = GetRand
   Next
 Next
End Sub
'Load ANN weights file for continous training
Private Sub LoadNN()

End Sub
'Run an ANN here
Private Sub RunNN()
Dim i As Integer
```

```
Dim j As Integer

'Apply query picture to input array
For i = 1 To (PicW * PicH)
   Inp(i) = TrainData.Data(i)
Next
'Sum all dendrite signals in hidden layer
For j = 1 To HiddenNode
  Hidden(j) = 0
  For i = 1 To (PicW * PicH)
    Hidden(j) = Hidden(j) + (Inp(i) * HideWeight(i, j))
  Next
Next
'Calculate Activation of each node in hidden layer
For i = 1 To HiddenNode
  Hidden(i) = Sigmoid(Hidden(i))
Next

'Sum all dendrite signals in output layer
For j = 1 To OutNode
   Out(j) = 0
   For i = 1 To HiddenNode
      Out(j) = Out(j) + (Hidden(i) * OutWeight(i, j))
   Next
Next
'calculate Activation of each node in output layer
For i = 1 To OutNode
   Out(i) = Sigmoid(Out(i))
Next

End Sub
'BackPropagation learning rule
Private Sub TrainNN()
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim Error As Double
Dim OldWeight As Double
Dim WeightD As Double
Dim hBetaD As Double
Dim OutputD As Double
Dim dNowWeight As Double
```

```
'Beta(s) of Output nodes
For j = 1 To OutNode
   If j = GetMostActivation(TrainData.Target) Then
      OutBeta(j) = Out(j) - 1
   Else
      OutBeta(j) = Out(j)
   End If
Next

'Beta(s) of hidden nodes
For j = 1 To HiddenNode
   hBetaD = 0
   For k = 1 To OutNode
      OutputD = Out(k)
      hBetaD = hBetaD + (OutWeight(j, k) * OutputD * (1 - OutputD) * OutBeta(k))
   Next
   HideBeta(j) = hBetaD
Next

'Calculate error here!, i.e. Sum square error
For j = 1 To OutNode
   Error = Error + (OutBeta(j) * OutBeta(j))
   Error = Error / 2
Next

'Calculate new weights for hidden layer
For i = 1 To (PicH * PicW)
   For j = 1 To HiddenNode
      OldWeight = HideWeight(i, j)
      WeightD = HideWeight(i, j)
      dNowWeight = -1 * (LearningRate * Inp(i) * Hidden(j) * (1 - Hidden(j)) *
HideBeta(j))
      HideWeight(i, j) = WeightD + dNowWeight + (theta * DelHideWeight(i, j))
      DelHideWeight(i, j) = dNowWeight
   Next
Next
'Calculate new weights for output layer
For i = 1 To HiddenNode
   For j = 1 To OutNode
```

```
        WeightD = OutWeight(i, j)
        dNowWeight = -1 * (LearningRate * Hidden(1) * Out(j) * (1 - Out(j)) *
OutBeta(j))
        OutWeight(i, j) = WeightD + dNowWeight + (theta * DelOutWeight(i, j))
        DelOutWeight(i, j) = dNowWeight

    Next
Next
End Sub
```

ข.4 ซอร์ซโคด ANNContrller

```
Option Explicit

'LPT Port Interface DLL
Private Declare Sub Out Lib "inpout32.dll" Alias "Out32" (ByVal PortAddress As
Integer, ByVal Value As Integer)

'DLL Set For CopyMemory From DIB to Gets Bit Value
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (lpvDest
As Any, lpvSource As Any, ByVal cbCopy As Long)
Private Declare Function VarPtrArray Lib "msvbvm50.dll" Alias "VarPtr" (Ptr() As
Any) As Long

'Use ImageProcessDIB Class Module
Private WithEvents m_cImage As cImageProcessDIB
Private m_cDib As New cDIBSection            'For Manage Picture
Private m_cDibBuffer As New cDIBSection      'Picture Data Buffer

'Use CCRP Timer Instead of VB Timer
Private WithEvents Timer2 As ccrpTimer

'Var. Type For Gets Bits Value
Private Type SAFEARRAYBOUND
    cElements As Long
    lLbound As Long
End Type
Private Type SAFEARRAY2D
    cDims As Integer
    fFeatures As Integer
    cbElements As Long
    cLocks As Long
    pvData As Long
    Bounds(0 To 1) As SAFEARRAYBOUND
End Type
```

```
Private c() As Double       'Control Direction Var
Private pwrite As Integer   'Port Adress Var
Private lwndC As Long       'Handle to the Capture Windows
Const iDevice = 0           'Capture Devive No.0
Private MyInput(960) As Double ' As Input For ANN

Private Sub ANNload_Click()
CommonDialog1.Filter = "Trained Network Weights File|*.net"
CommonDialog1.ShowOpen
If CommonDialog1.FileName <> "" Then
Call LoadNet(CommonDialog1.FileName)
End If
Start.Enabled = True
End Sub
Private Sub Form_Load()
Dim CapDevName As String * 100
Dim CapDevVer As String * 100

'Set up Timer
'Timer1 for Control via ANN
Timer1.Enabled = False

'Timer2 for Counter Clock
Set Timer2 = New ccrpTimer
With Timer2
  .Stats.Frequency = 33
  .Interval = 33
  .Enabled = False
End With

'Image Processing Class Module for Process Grabed Image
Set m_cImage = New cImageProcessDIB

'Prepare Capture Device
capGetDriverDescriptionA 0, CapDevName, 100, CapDevVer, 100
```

```
lwndC = capCreateCaptureWindowA(CapDevName, WS_CHILD Or WS_VISIBLE,
0, 0, 189, 149, Picture1.hwnd, 0)

'Prepare LPT Port Interface
pwrite = &H378
Out pwrite, &H0

'Prepare Com Port Interface
MSComm1.CommPort = 1
MSComm1.Settings = "9600,N,8,1"

'Prepare ANN
Call CreateNet(1, Array(960, 5, 30))

'Neutral Status
Timer1.Enabled = False
MSComm1.InputMode = comInputModeText
Shape1.BackColor = vbWhite
Start.Enabled = False
HWDisconnect.Enabled = False
End Sub

Private Sub HWConnect_Click()
MSComm1.PortOpen = True
Shape1.BackColor = vbRed
'Connect to Device
SendMessage lwndC, WM_CAP_DRIVER_CONNECT, iDevice, 0
'Set Scaleable Preview
SendMessage lwndC, WM_CAP_SET_SCALE, True, 0
'Set Preview Rate
SendMessage lwndC, WM_CAP_SET_PREVIEWRATE, 66, 0
'Start Preview
SendMessage lwndC, WM_CAP_SET_PREVIEW, True, 0

'Recieve Invitation Messages From Com Port
On Error Resume Next
   MSComm1.DTREnable = False
   MSComm1.DTREnable = True
```

```
   MSComm1.InputLen = 0
Text6.Text = Text6.Text & MSComm1.Input

HWDisconnect.Enabled = True
End Sub

Private Sub HWDisconnect_Click()
MSComm1.PortOpen = False
Timer1.Enabled = False
Shape1.BackColor = vbWhite
Text6.Text = ""
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text15.Text = ""
'Disconnect Capture Device
SendMessage lwndC, WM_CAP_DRIVER_DISCONNECT, iDevice, 0
DestroyWindow Picture1

End Sub
'Start Operation
Private Sub Start_Click()
'Timer2.Enabled = True
'Enable Timer3 for Counter Clock
Timer2.Enabled = True
Timer1.Enabled = True 'Activate AI
End Sub
'Stop Operation
Private Sub Stop_Click()
'Pause Car
Out pwrite, &H0
Shape2.BackColor = vbWhite
Text15.Text = "Pause"

Timer2.Enabled = False
```

```
Timer1.Enabled = False

'Make Sure That All Movement is Stoped
Out pwrite, &H0
Shape2.BackColor = vbWhite
Text15.Text = "Pause"
End Sub

Private Sub Timer1_Timer()

'Image Processor Timer
'Grab Frame Timmer
SendMessage lwndC, WM_CAP_GRAB_FRAME_NOSTOP, 0, 0
SendMessage lwndC, WM_CAP_EDIT_COPY, 0, 0

'Process Grabed Image
m_cDib.CreateFromPicture Clipboard.GetData
m_cDibBuffer.Create m_cDib.Width, m_cDib.Height
GrayScale
Resample 30, 32
GetbitsAsByteArray

' ANN Timmer
'Query of ANN
 c() = Run(MyInput)
Text1.Text = GetMostActivation(c)
Text3.Text = Int(GetMostActivation(c))
Text2.Text = "&H" & Hex$(Text3.Text)
Text4.Text = Chr(Text2.Text)
MSComm1.Output = Chr(Text2.Text)

'Let's Move Forward
Out pwrite, &H4
Text15.Text = "Forward"
Shape2.BackColor = vbRed

End Sub
```

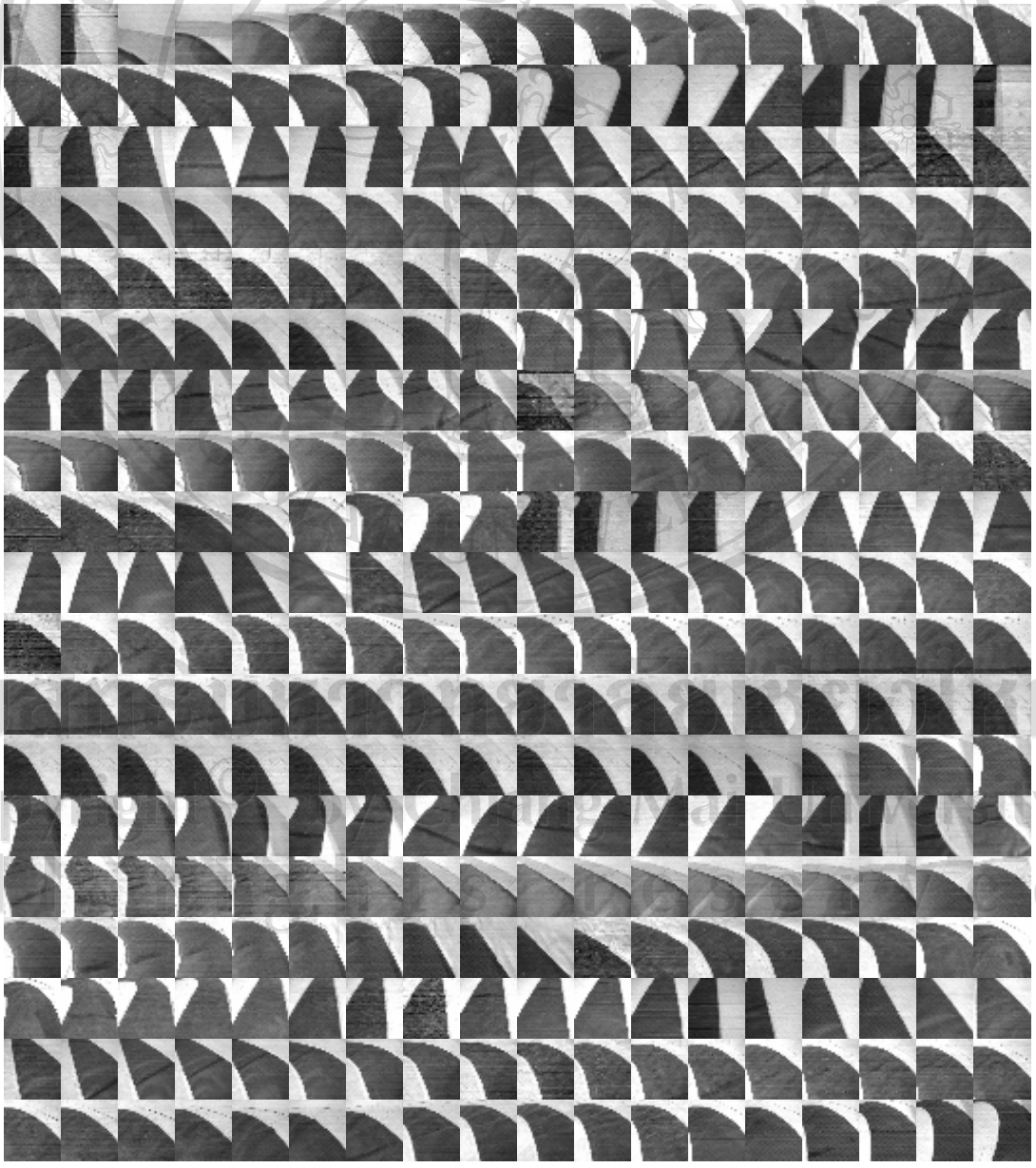
```
'Time Counter for Refernce
Private Sub Timer2_Timer(ByVal Milliseconds As Long)
Static Ticks As Long

Ticks = Ticks + (Milliseconds / 100)
Label12.Caption = Format(Ticks, "#,##0")
End Sub
'For Tuning Capture Device
Private Sub TuneCap_Click()
SendMessage lwndC, WM_CAP_DLG_VIDEOSOURCE, 0, 0
End Sub
'Rebuilt Processed Picture
Private Sub Render()
Picture2.Width = m_cDib.Width * Screen.TwipsPerPixelX
Picture2.Height = m_cDib.Height * Screen.TwipsPerPixelY
m_cDib.PaintPicture Picture2.hdc
End Sub
'For Process Grabed Picture As Gray Scale
Private Sub GrayScale()
m_cImage.GrayScale m_cDib
'Render
End Sub
'Resizing Picture
Private Sub Resample(ByVal lW As Long, ByVal lH As Long)
Dim cDib As New cDIBSection
   If (lW <> m_cDib.Width) Or (lH <> m_cDib.Height) Then
      Set cDib = m_cDib.Resample(lH, lW)
      Set m_cDib = cDib
      m_cDibBuffer.Create m_cDib.Width, m_cDib.Height
      Render
   End If
End Sub
Private Sub GetbitsAsByteArray()
Dim tSA As SAFEARRAY2D
Dim bDib() As Byte
Dim BitsByte() As Byte
Dim i As Integer
Dim j As Integer
Dim k As Integer

'Get The bits in DIB Section
   With tSA
```

```
        .cbElements = 1
        .cDims = 2
        .Bounds(0).lLbound = 0
        .Bounds(0).cElements = m_cDib.Height
        .Bounds(1).lLbound = 0
        .Bounds(1).cElements = m_cDib.BytesPerScanLine
        .pvData = m_cDib.DIBSectionBitsPtr
    End With
    CopyMemory ByVal VarPtrArray(bDib()), VarPtr(tSA), 4
    BitsByte = bDib

k = 1
i = 0
j = 0
    For j = 0 To 31
    For i = 0 To 29
      MyInput(k) = (BitsByte(i, j) / 255)
      k = k + 1
    Next
    Next
End Sub

'Find the most active node of output
Function GetMostActivation(Result As Variant) As Integer
Dim MostActiveNode As Integer
Dim i As Integer
MostActiveNode = 1
For i = 1 To UBound(Result)
If Result(MostActiveNode) < Result(i) Then
MostActiveNode = i
End If
Next
GetMostActivation = MostActiveNode
End Function
```

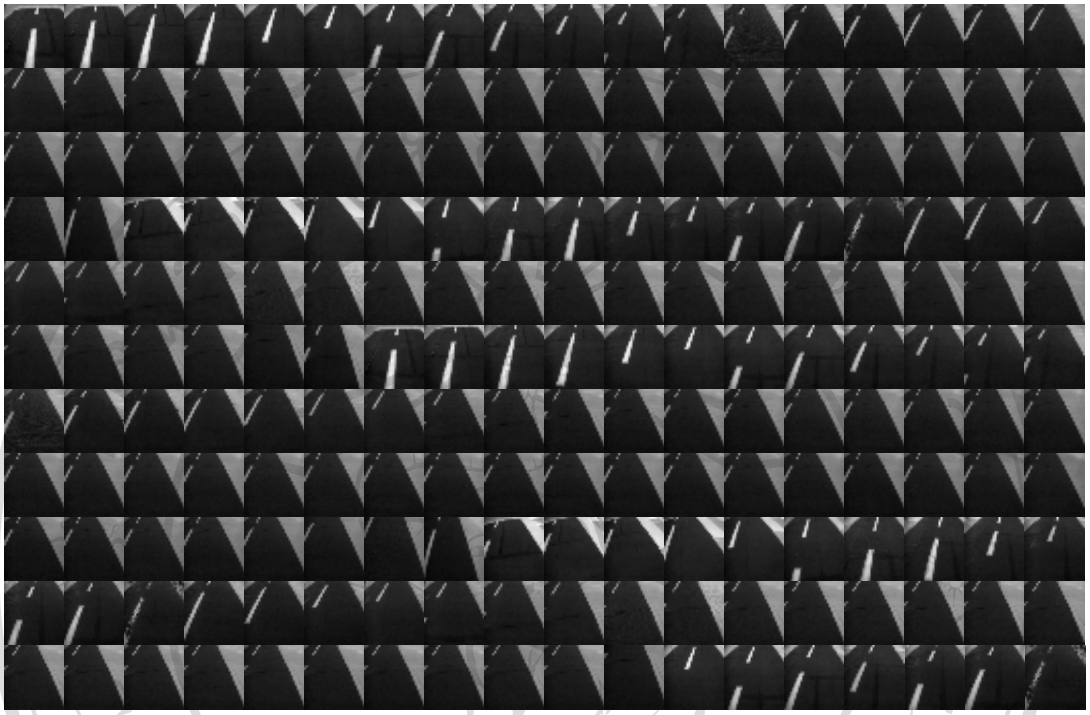
ภาคผนวก ค

ตัวอย่างภาพสำหรับการสอนระบบเครือข่ายประสาทเทียม

ค.1 ตัวอย่างของภาพที่ใช้สอนระบบเครือข่ายประสาทเทียม สำหรับตามเส้นทาง และหลีกเลี่ยงสิ่งกีดขวาง



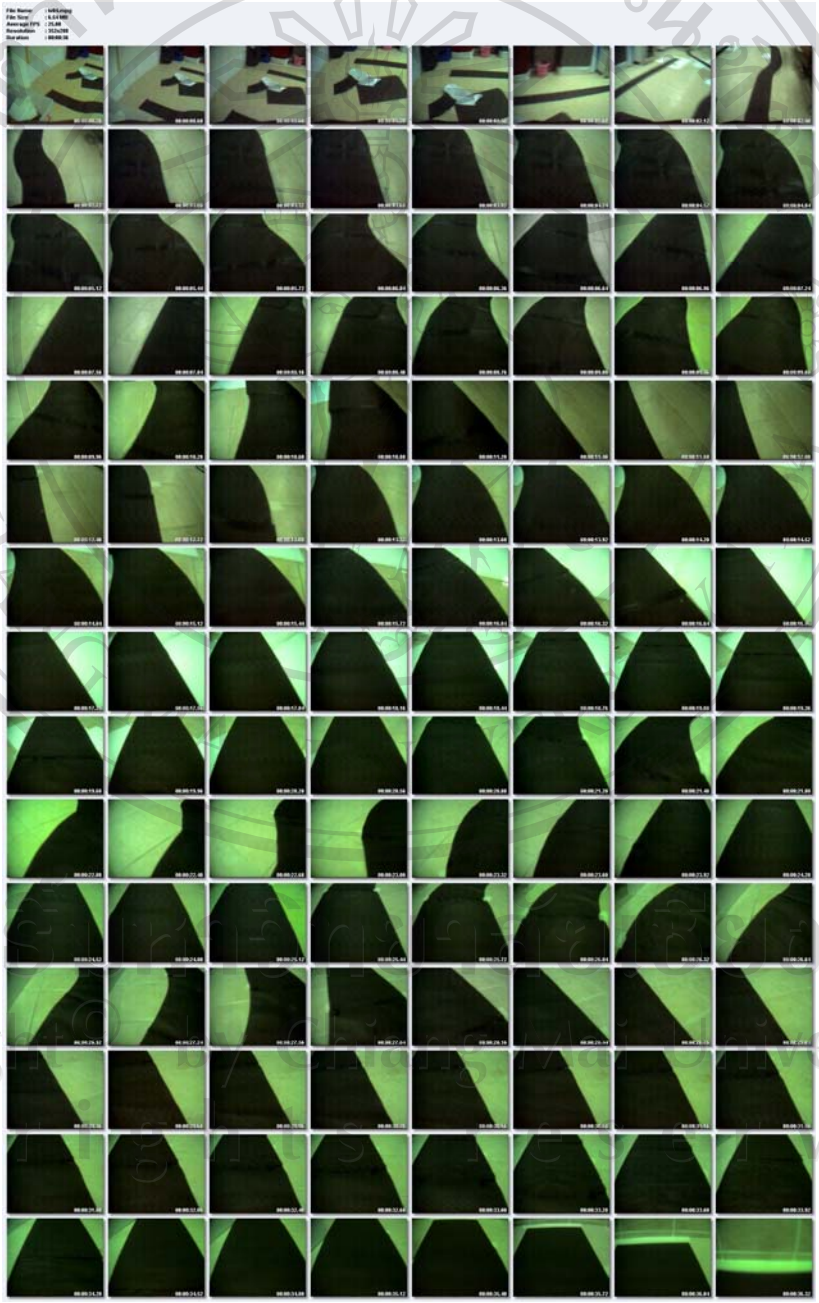
ค.2 ภาพตัวอย่างที่ใช้สอนเพื่อการติดตามถนนจำลองแบบสองช่องทางวิ่ง



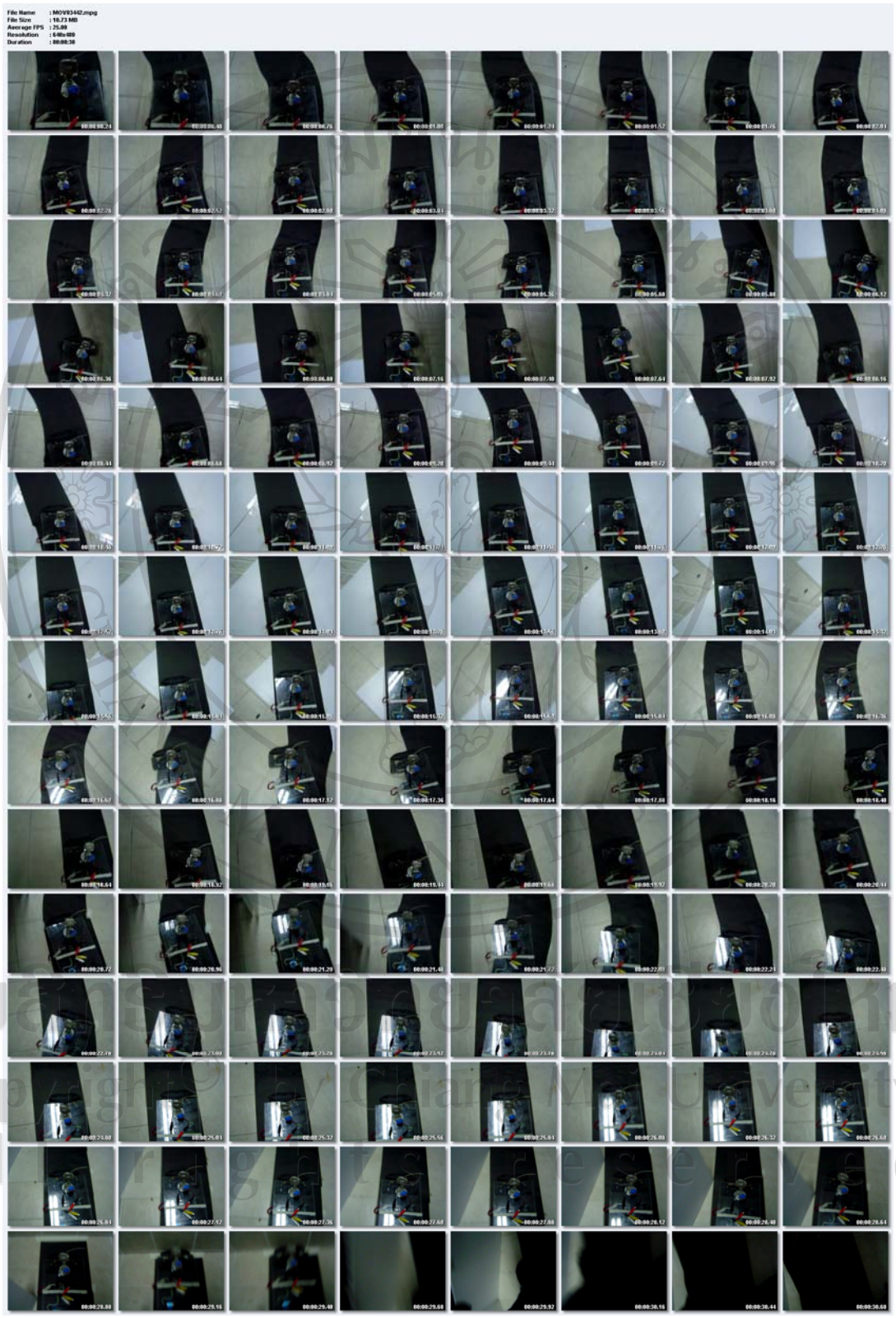
ภาคผนวก ง

ภาพจากวีดิทัศน์บันทึกการทดสอบระบบยานยนต์

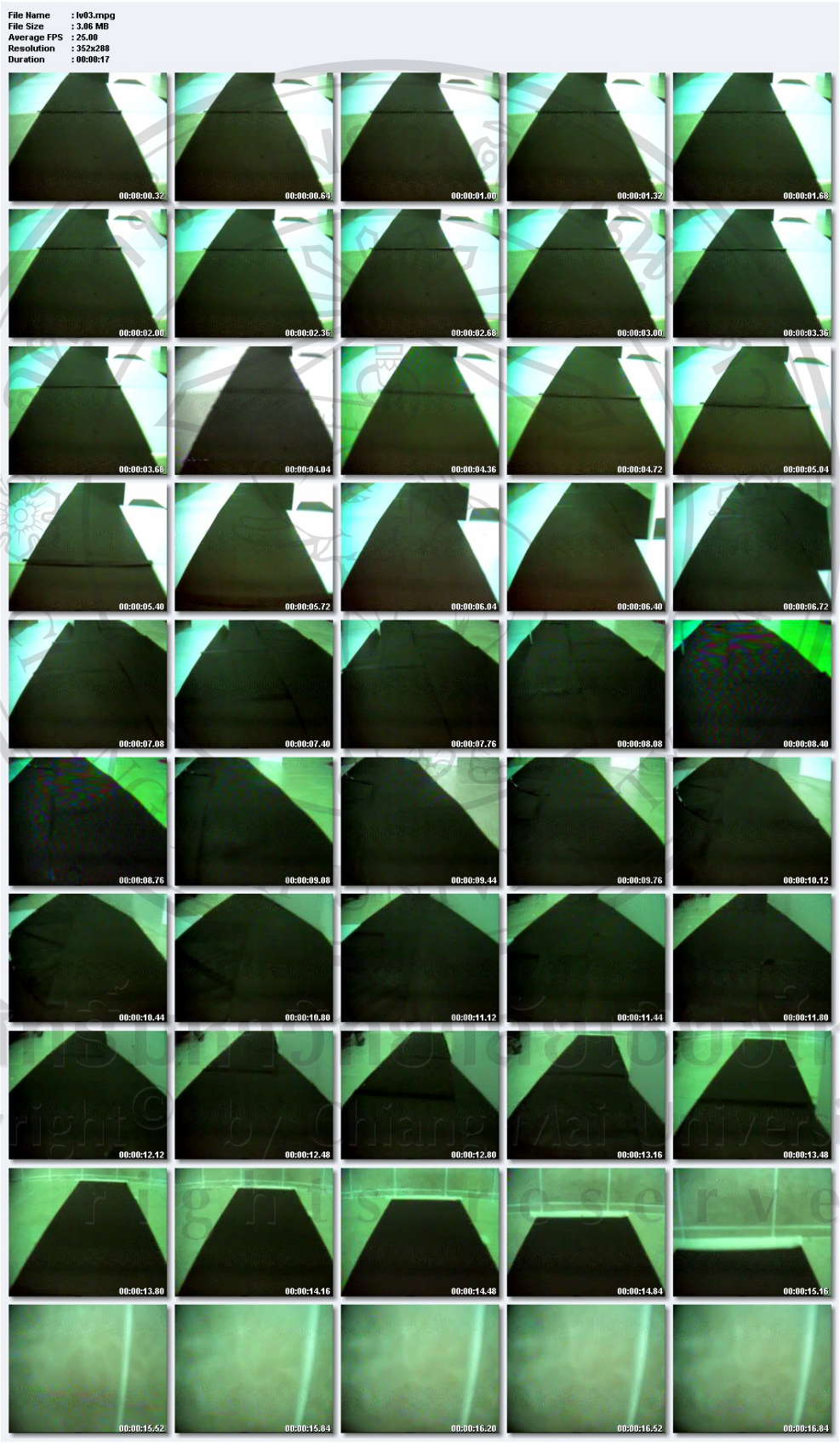
ง.1 ภาพชุดผลการทดสอบการติดตามเส้นทางเมื่อมองจากหน้ารถ



ง.2 ภาพชุดผลการทดสอบการติดตามเส้นทางเมื่อมองจากด้านบนรถ

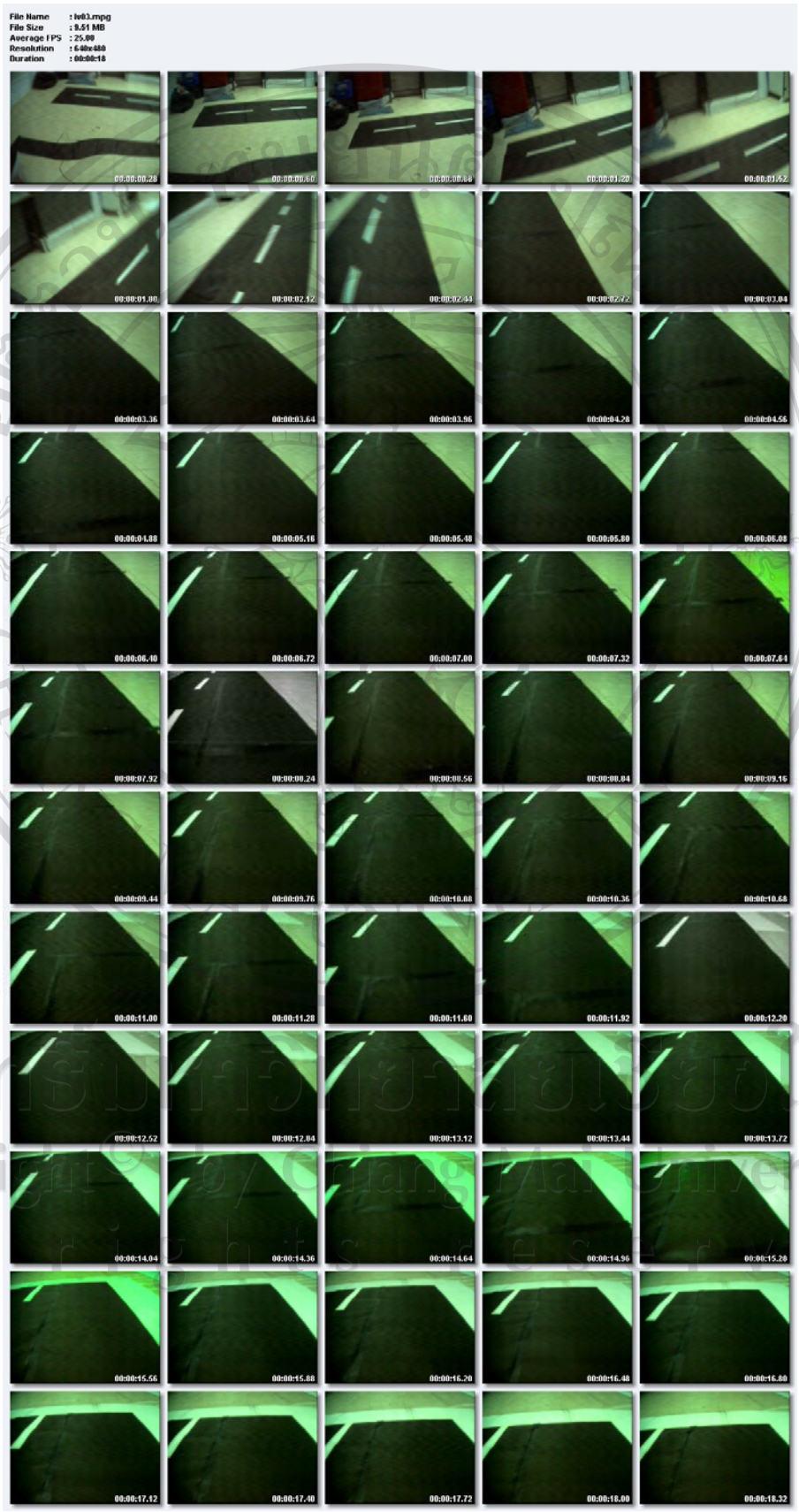


ง.3 ภาพชุดผลการทดสอบการหลบเลี่ยงสิ่งกีดขวางเมื่อมองจากหน้ารถ

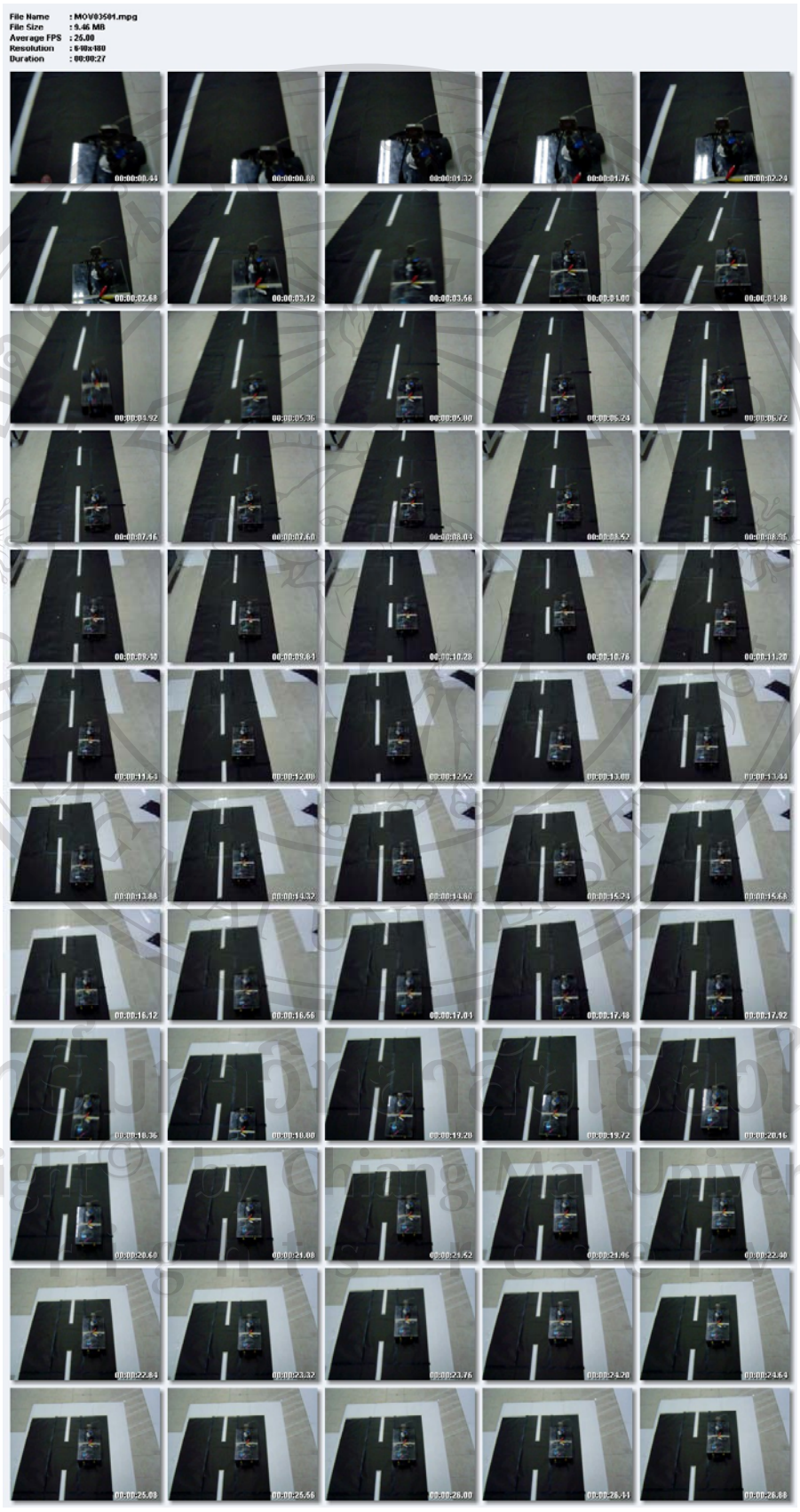


ง. 4 ภาพชุดผลการทดสอบหลบเลี่ยงสิ่งกีดขวางเมื่อมองจากด้านบนรถ

ง.5 ภาพชุดผลการทดสอบการติดตามถนน 2 เลนเมื่อมองจากหน้ารถ



ง.6 ภาพชุดผลการทดสอบการติดตามถนน 2 เลนเมื่อมองจากด้านบนรถ



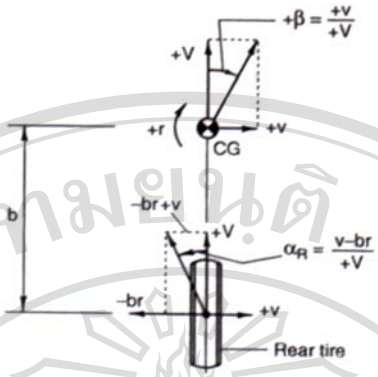
ภาคผนวก จ

ระบบพลศาสตร์ของยานยนต์และสมการการเคลื่อนที่

ในการวิเคราะห์พลศาสตร์ของยานยนต์นี้ จะเน้นหนักไปในเรื่องของการควบคุมการเข้าโค้ง ซึ่งหากจะทำการวิเคราะห์ตลอดกระบวนการเข้าโค้งนั้นเป็นสิ่งที่ยุ่งยากและซับซ้อนมากเกินไป และหากคำนึงตามกายภาพตามความเป็นจริงแล้ว เราจะสามารถแบ่ง การเข้าโค้งได้เป็น 3 ช่วง กล่าวคือ ในขณะที่ รถมีความเร็วและเดินทางจากเส้นตรง ในขั้นตอนแรกนั้น ความเร็วหันเห (Yawing velocity) และ ความเร็วด้านข้าง (Lateral velocity) จะเริ่มเพิ่มขึ้นจากศูนย์ (ซึ่งเป็นการวิ่งในแนวตรง) ขึ้นเรื่อยๆ จนกระทั่ง มีค่าตามการเลี้ยวแบบอยู่ตัว (Steady turn) ซึ่งในส่วนนี้จะเรียกว่า ช่วง เข้าสู่โค้ง ในภาวะชั่วคราว (Transient turn-entry) ช่วงนี้ความเร็วหันเหและความเร็วด้านข้างจะเปลี่ยนแปลงกับเวลา ช่วงที่สอง คือช่วงเข้าโค้งแบบอยู่ตัว (Steady-state cornering) ซึ่งความเร็วหันเห ความเร็วด้านข้าง และมุมไถลของยาง มีค่าคงที่ และตัวยานยนต์เองจะเคลื่อนที่ตามเส้นทางที่มีความโค้ง ไม่เปลี่ยนแปลง และช่วงสุดท้าย คือช่วงออกจากโค้ง ในภาวะชั่วคราว (Transient turn-exit) ในช่วงนี้ก็จะเป็นเช่นเดียวกับช่วงแรกกล่าว คือความเร็วหันเห และความเร็วด้านข้าง เปลี่ยนแปลงตามเวลา จนกระทั่งเข้าสู่ ศูนย์ สำหรับการวิ่งในแนวตรง

จากการอธิบายตามข้างต้นอาจสรุปได้ว่า ในขบวนการเข้าโค้งนั้นสามารถแบ่งออก เป็นสองลักษณะ คือช่วงอยู่ตัวและช่วงขึ้นกับเวลา การระบุช่วงแบบนี้ มีประโยชน์มาก สำหรับทุกตัวอย่างของโค้งในถนน ตามความจริงแล้วในแต่ละโค้งนั้น มีความแตกต่างแตกต่างกันในเรื่องช่วงแต่ละช่วงอย่างมาก เช่นโค้งที่กวาดออกกว้าง (Sweeper) จะมีช่วงอยู่ตัวที่นานมากว่าเมื่อเทียบกับช่วงขึ้นกับเวลา ในขณะเดียวกัน โค้งที่มีลักษณะปลายแคบ เช่น โค้งรูปตัว U (Hairpin) จะมี ช่วงอยู่ตัวที่สั้นมากๆ หรือในการเปลี่ยนแนว (Lane) ซ้าย-ขวา อย่างรวดเร็ว อาจจะส่งผลรวมให้ ช่วงอยู่ตัว หายไปเลยก็ได้

สมการการเคลื่อนที่จะทำการพัฒนาดังข้อจำกัดความที่กล่าวมาข้างต้น จะสามารถสร้างต้นแบบ ที่มีระดับความอิสระ เท่ากับ 2 โดยมีตัวแปรของการเคลื่อนที่ เป็นฟังก์ชั่นของแรงและโมเมนต์ ที่กระทำบนยานยนต์ เมื่อพิจารณา ล้อหลังดังรูป จ.1 จะสามารถแสดงเป็นสมการคือ [10]



รูป จ.1 ความเร็วต่างๆที่ล้อหลัง [Milliken W. F., Milliken D. L., 1995]

$$\alpha_R = \beta - \frac{br}{V} \qquad \text{(จ.1)}$$

โดยที่

α_R คือ มุมไถลของยางที่ล้อหลัง

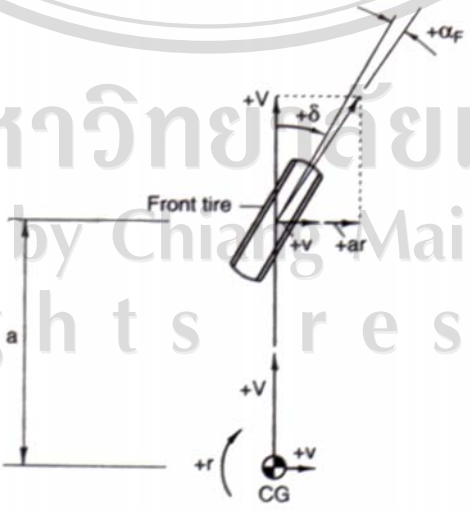
β คือ มุมไถลของยานยนต์ที่ จุดศูนย์กลางศูนย์ถ่วง

b คือ ตำแหน่งของจุดศูนย์ถ่วงถึงจุดศูนย์กลางล้อหลัง

r คือ ความเร็วหันเห(หน่วย : เรเดียน/วินาที)

V คือ ความเร็วของยานยนต์

ทำนองเดียวกันหากพิจารณา ล้อหน้าตามรูป จ.2 ก็จะสามารถสร้างสมการได้คือ



รูป จ.2 ความเร็วต่างที่ล้อหน้า [Milliken W. F., Milliken D. L., 1995]

$$\alpha_F = \beta + \frac{ar}{V} - \delta \tag{จ.2}$$

โดยที่

α_F คือ มุมไถลของยางที่ล้อหน้า

a คือ ตำแหน่งของจุดศูนย์ถ่วงถึงจุดศูนย์กลางล้อหน้า

δ คือ มุมเลี้ยวล้อหน้า

หากสมมติว่า แรงด้านข้างของยางจะมีเปลี่ยนแปลงแบบเชิงเส้นกับมุมไถล แรงด้านข้างของยางด้านหน้าและหลัง ก็สามารถแสดงได้ดังสมการ (3) และ สมการ (4)

$$Y_F = C_F\left(\beta + \frac{ar}{V} - \delta\right) = C_F\beta + C_F\left(\frac{ar}{V}\right) - C_F\delta \tag{จ.3}$$

$$Y_R = C_R\left(\beta - \frac{br}{V}\right) = C_R\beta - C_R\left(\frac{br}{V}\right) \tag{จ.4}$$

โดยที่

C_F คือ ความยืดหยุ่นการเข้าโค้ง (Cornering stiffness) ของยางล้อหน้า

C_R คือ ความยืดหยุ่นการเข้าโค้งของยางล้อหลัง

แรงด้านข้างรวม Y ก็จะเป็น

$$Y = Y_F + Y_R = (C_F + C_R)\beta + \frac{1}{V}(aC_F - bC_R)r - C_F\delta \tag{จ.5}$$

และโมเมนต์หันเหรวม ขอบแกน Z ที่จุดศูนย์ถ่วง N คือ

$$N = Y_F a - Y_R b = (aC_F - bC_R)\beta + \frac{1}{V}(a^2C_F + b^2C_R)r - aC_F\delta \tag{จ.6}$$

จากสมการข้างต้นจะเห็นได้ว่ามีความยุ่งยากและยาวเกินไป ในที่นี้จะนำการเขียนย่อแบบอนุพันธ์ (Delivative notation) มาใช้ สมการจะจัดรูปใหม่ได้เป็น

$$Y = f(\beta, r, \delta) = \left(\frac{\partial Y}{\partial \beta}\right)\beta + \left(\frac{\partial Y}{\partial r}\right)r + \left(\frac{\partial Y}{\partial \delta}\right)\delta$$
$$Y = Y_\beta\beta + Y_r r + Y_\delta\delta \tag{จ.7}$$

$$N = f(\beta, r, \delta) = \left(\frac{\partial N}{\partial \beta}\right)\beta + \left(\frac{\partial N}{\partial r}\right)r + \left(\frac{\partial N}{\partial \delta}\right)\delta$$
$$N = N_\beta\beta + N_r r + N_\delta\delta \tag{จ.8}$$

และเมื่อ

$$I_Z \frac{dr}{dt} = I_Z \dot{r} \qquad \text{(จ.9)}$$

โดยที่

I_Z คือ โมเมนต์ของความเฉื่อยรอบแกน Z

และความเร่งด้านข้าง a_y สามารถเขียนได้ว่า

$$a_y = Vr + V\dot{\beta} = V\left(r + \dot{\beta}\right) \qquad \text{(จ.10)}$$

และเมื่อรวมสมการทั้ง 2 ชุด 4 สมการ คือสมการ (จ.7) ถึง (จ.10) เข้าด้วยกันจะได้ สมการการเคลื่อนที่ ที่สมบูรณ์ ออกมาดังสมการ (จ.11)

$$\begin{aligned} I_Z \dot{r} &= N_\beta \beta + N_r r + N_\delta \delta \\ mV\left(r + \dot{\beta}\right) &= Y_\beta \beta + Y_r r + Y_\delta \delta \end{aligned} \qquad \text{(จ.11)}$$

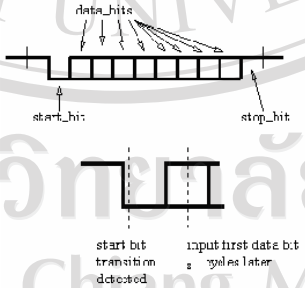
ภาคผนวก ฉ

การใช้งานไมโครคอนโทรลเลอร์ แบบ MCS-51

ภายในตัวชิพของไมโครคอนโทรลเลอร์เองนั้นประกอบไปด้วย ความจำ พอร์ท และหน่วยประมวลผล ภายในชิพเดียว ซึ่งกล่าวได้ว่า ไมโครคอนโทรลเลอร์คือคอมพิวเตอร์ชิพเดียว สำหรับไมโครคอนโทรลเลอร์ตระกูล MCS-51 เป็นไมโครคอนโทรเลอร์ขนาด 8 บิต ซึ่งถูกพัฒนาขึ้นโดยบริษัท Intel และมีการนำไปดัดแปลงเพิ่มเติมจนเกิดชิพหลายๆรหัสด้วยกัน แต่อย่างไรก็ตามทุกๆชิพของไมโครคอนโทลเลอร์ ตระกูล MCS-51 จะมีคำสั่งพื้นฐาน 256 คำสั่ง ตามมาตรฐานของ Intel แต่ในส่วนของรายละเอียด อื่นๆเกี่ยวกับข้อมูลขา พอร์ท หน่วยความจำภายใน ไทเมอร์ และรีจิสเตอร์พิเศษ นั้นให้อ้างอิงจากคู่มือของชิพรหัสนั้นๆ ซึ่งในแต่ละผู้ผลิตจะมีความต่างกับออกไป

ฉ.1 การรับส่งข้อมูลแบบอนุกรมกับไมโครคอนโทรลเลอร์แบบ MCS-51

ในงานวิจัยนี้จะทำการพัฒนาโปรแกรม เพื่อใช้ควบคุมไมโครคอนโทรเลอร์ด้วยภาษา Assembly ตามมาตรฐาน Intel และภายในชิพ MCS-51 ทุกๆรหัสจะมี UART (Universal Asynchronous receiver transmitter) ที่ทำหน้าที่เป็น รีจิสเตอร์เลื่อน (Shift register) แปลงจาก การรับ/ส่งข้อมูลแบบขนาน สู่ การรับ/ส่งข้อมูลแบบอนุกรม ติดตั้งไว้ภายในอยู่แล้ว



รูป ฉ.1 การส่งข้อมูลแบบขนาน

ซึ่งพอร์ทอนุกรมของไมโครคอนโทรเลอร์แบบ MCS-51 นี้โดยส่วนมากจะอยู่ที่ขารหัส P3.0 เป็นขา RXD (ขารับสัญญาณ) และ P3.1 เป็นขา TXD (ขาส่งสัญญาณ) การติดต่อรับส่งข้อมูลของไมโครคอนโทรลเลอร์แบบ MCS-51 นี้มีรีจิสเตอร์ที่สำคัญ 2 ตัวคือ SBUF และ SCON โดยมี

ตำแหน่งที่ 99h และ 98h ตามลำดับ โดยรีจิสเตอร์ SBUF จะทำหน้าที่ส่งข้อมูลออกไปทางพอร์ทอนุกรมถ้าหากมีการเขียนข้อมูลใดๆลงบนรีจิสเตอร์นี้และจะทำหน้าที่รับข้อมูลหากมีการอ่านข้อมูลจากรีจิสเตอร์นี้ สำหรับรีจิสเตอร์ SCON เป็นรีจิสเตอร์ที่เข้าถึงข้อมูลระดับบิตได้ รีจิสเตอร์นี้จะทำหน้าที่ควบคุมและบอกสถานะต่างๆของการรับส่งข้อมูลแบบอนุกรม สำหรับบอดเรทสามารถหาได้จากการหารสัญญาณนาฬิกาที่ใช้กับ MCS-51 ซึ่งจะกล่าวต่อไปในภายหลัง

ตาราง ฉ.1 บิตต่างๆ ของรีจิสเตอร์ SCON

บิต	ชื่อ	ตำแหน่ง	ความหมาย
SCON.7	SM0	9Fh	บิตเลือกโหมดการทำงานบิต 0
SCON.6	SM1	9Eh	บิตเลือกโหมดการทำงานบิต 1
SCON.5	SM2	9Dh	บิตเลือกโหมดการทำงานบิต 2
SCON.4	REN	9Ch	บิตแฟลกกำหนดยอมให้มีการรับข้อมูล
SCON.3	TB8	9Bh	ค่าของบิต 9 สำหรับการรับส่งข้อมูลในโหมด 2 และ 3 สามารถ Set และ Clear ได้โดย ซอฟต์แวร์
SCON.2	RB8	9Ah	ค่าของบิต 9 เมื่อรับข้อมูลเข้ามา
SCON.1	TI	99h	บิตแฟลกแสดงการอินเทอร์รัพท์ภายหลังการส่งข้อมูลออกไป โดยจะ Set เมื่อส่งข้อมูลออกไปหมดแล้วและสามารถ Clear ได้โดยซอฟต์แวร์
SCON.0	RI	98h	แฟลกแสดงการอินเทอร์รัพท์ภายหลังรับข้อมูลเข้ามาสามารถ Clear ได้ด้วยซอฟต์แวร์

ตาราง ฉ.2 แสดงโหมดต่างๆของการรับส่งแบบอนุกรม

SM0	SM1	โหมดทำงาน	ความหมาย	บอดเรท
0	0	0	รีจิสเตอร์เลื่อน	เปลี่ยนแปลงไม่ได้
0	1	1	8 บิต-UART	สามารถเปลี่ยนแปลงได้โดยกำหนดTimer
1	0	2	9 บิต-UART	เปลี่ยนแปลงไม่ได้
1	1	3	9 บิต-UART	สามารถเปลี่ยนแปลงได้โดยกำหนดTimer

ในงานวิจัยนี้ จะกล่าวถึงการใช้การติดต่อแบบอนุกรมโหมดการทำงานแบบ 1 เท่านั้น ในโหมดการทำงานนี้จะเป็นการรับส่งข้อมูลแบบ 10 บิต ซึ่งประกอบไปด้วยบิตเริ่มต้น (เป็น “0”) ข้อมูล 8 บิต และบิตจบ (เป็น “1”) นอกจากนี้ยังสามารถกำหนด บอดเรท ได้โดยค่า บอดเรทนี้จะแปรตามไทเมอร์ตัวที่ 1 ค่าบอดเรทนี้จะกำหนดโดยค่าที่กำหนดให้ไทเมอร์ หารด้วย 32 และเมื่อทำการรับ หรือ ส่งแล้ว บิตที่เกี่ยวข้องกับการ อินเทอร์รัพท์ (Interrupt) จะถูกกำหนดค่าให้เป็น “1” ซึ่งจะต้องทำการเคลียร์ค่านี้ในภายหลัง

ฉ.2 การใช้งานไทเมอร์ ภายใน MCS-51

จะเห็นได้ว่าการใช้งานติดต่อกับพอร์ทอนุกรมด้วยไมโครคอนโทรเลอร์ตระกูล MCS-51 ในโหมดการทำงานที่ 1 จำเป็นต้องใช้ งานไทเมอร์ และก่อนที่จะกล่าวถึงการใช้งานไทเมอร์ในการกำหนด บอดเรทให้แก่การรับส่งแบบอนุกรมนั้น จำเป็นต้องทราบถึง รีจิสเตอร์พิเศษที่เกี่ยวข้องกันกับการกำหนดการทำงานของไทเมอร์ก่อน

ตาราง ฉ.3 รีจิสเตอร์ที่ใช้ในไทเมอร์

รีจิสเตอร์	หน้าที่	ตำแหน่ง	สามารถอ้างอิง
TCON	Control	88h	ได้
TMOD	Mode	89h	ไม่ได้
TL0	Timer 0 Low -byte	8Ah	ไม่ได้
TL1	Timer 1 Low - bye	8Bh	ไม่ได้
TH0	Timer 0 High – byte	8Ch	ไม่ได้
TH1	Timer 1 High – byte	8Dh	ไม่ได้
T2CON*	Timer 2 Control	C8h	ได้
RCAP2L*	Timer 2 Low – byte Capture	CAh	ไม่ได้
RCAP2H*	Timer 2 High – byte Capture	CBh	ไม่ได้
TL2*	Timer 2 Low – byte	CCh	ไม่ได้
TH2*	Timer 2 High – byte	CDh	ไม่ได้

*มีให้ใช้งานในรหัส 8032/8052 เท่านั้น

ตามมาตรฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะมีไทเมอร์ ทั้งหมดอย่างน้อยสองตัวขึ้นอยู่กับรหัสของชิพแต่ละตัว ไทเมอร์ในส่วนของไมโครคอนโทรลเลอร์นั้น อาจจะกล่าวได้ว่าเป็น ตัวฟลิบฟลอบ (Flip-flop) มาต่อเรียงกันโดยมีสัญญาณนาฬิกาเป็นอินพุต ในการใช้งาน

ไทเมอร์นั้น ต้องทำความเข้าใจรีจิสเตอร์สองตัว คือ TMOD และ TCON ซึ่งเป็นรีจิสเตอร์ ที่ทำหน้าที่ควบคุมไทเมอร์ ตัวรีจิสเตอร์ TMOD จะแบ่งออกเป็น 2 กลุ่ม กลุ่มละ 4 บิตโดย 4 บิตบนจะทำการควบคุมไทเมอร์ตัวที่ 2 (Timer 0) ส่วน 4 บิตล่างจะเป็นการควบคุม ไทเมอร์ตัวที่ 1 (Timer 1) ความหมายของแต่ละบิตดังแสดงไว้ในตาราง ฉ.4 ซึ่งตัวรีจิสเตอร์นี้เป็นตัวเลือกการทำงานว่าจะให้ตัว ไทเมอร์ทำงานในโหมดใดและเป็น ตัวจับเวลา หรือ ตัวนับเวลา (Counter) รีจิสเตอร์ TCON ไม่สามารถจะโปรแกรมเข้าไปในระดับบิตได้ (Not bit-addressable) ซึ่งการใช้งานมักจะโปรแกรมเข้าไปครั้งเดียวในตำแหน่งเริ่มต้นของโปรแกรม

ตาราง ฉ.4 รีจิสเตอร์ TMOD (Timer mode)

บิต	ชื่อ	Timer	ความหมาย
7	GATE	1	Gate bit ถ้าบิตนี้เซต (Set) วงจรจะทำงานเมื่อ INT1 เป็น High
6	C/T	1	เลือกการทำงาน 1 คือ ตัวนับเวลา, 2 คือ ตัวจับเวลา
5	M1	1	Mode bit 1
4	M0	1	Mode bit 0
3	GATE	0	Gate bit ของไทเมอร์ 0
2	C/T	0	เลือกการทำงาน 1 คือ ตัวนับเวลา, 2 คือ ตัวจับเวลา
1	M1	0	Mode bit 1
0	M0	0	Mode bit 0

ตาราง ฉ.5 การใช้โหมดต่างๆของไทเมอร์

M1	M0	โหมด	ความหมาย
0	0	0	ใช้เป็นไทเมอร์แบบ 13 บิต
0	1	1	ใช้เป็นไทเมอร์แบบ 16 บิต
1	0	2	ใช้เป็นไทเมอร์ แบบ 8 บิต Auto-reload
1	1	3	ใช้ไทเมอร์ ตัวที่ 1 แยกออกเป็น 2 ตัว ตัวละ 8 บิตคือ TL0 และ TH0 โดยไม่ใช้ไทเมอร์ตัวที่ 2

ตาราง ฉ.6 ความหมายแต่ละบิตของรีจิสเตอร์ TCON

บิต	ชื่อ	ตำแหน่งบิต	ความหมาย
TCON.7	TF1	8Fh	บิตแฟลกแสดงการโอเวอร์โฟลว์ (Over flow) จะทำการ เซต โดย ฮาร์ดแวร์ และทำการ ปรับเป็น 0 โดย ซอฟต์แวร์ ของไทเมอร์ตัวที่2
TCON.6	TR1	8Eh	บิตควบคุมการปิด/เปิด ไทเมอร์ของไทเมอร์ ตัวที่ 2 จะทำการ เซต โดย ฮาร์ดแวร์ และทำการ ปรับเป็น 0 โดย ซอฟต์แวร์
TCON.5	TF0	8Dh	บิตแฟลกแสดงการโอเวอร์โฟลว์ของไทเมอร์ตัวที่ 1
TCON.4	TR0	8Ch	บิตควบคุมการปิด/เปิด ไทเมอร์ของไทเมอร์ตัวที่ 1
TCON.3	IE1	8Bh	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT 1 จะทำการ เซต โดย ฮาร์ดแวร์ และทำการ ปรับเป็น 0 โดย ซอฟต์แวร์
TCON.2	IT1	8Ah	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT1 จะทำการ เซต โดย ฮาร์ดแวร์ และทำการ ปรับเป็น 0 โดย ซอฟต์แวร์
TCON.1	IE0	89h	บิตแฟลกแสดงการอินเทอร์รัพท์จาก INT 0
TCON.0	IT0	88h	บิตเลือกชนิดของสัญญาณอินเทอร์รัพท์จากอินเทอร์รัพท์ภายนอก INT0

จากตารางการทำงานข้างต้น การใช้งานไทเมอร์เพื่อกำหนด บอดเรท ให้แก่การติดต่อพอร์ทอนุกรม จะใช้ ไทเมอร์ใน โหมด 2 คือ ไทเมอร์แบบ 8 บิต Auto-reload กล่าวคือ ไทเมอร์จะใช้ไบต์ต่ำในการรับและเมื่อเกิดการ โอเวอร์โฟลว์ขึ้น ไทเมอร์จะทำการโหลดเอาค่าที่เก็บไว้ในไบต์สูง มาเก็บไว้ในไบต์ต่ำเพื่อทำการเริ่มต้นการนับต่อไป ด้วยลักษณะเช่นนี้ โหมดการทำงานนี้จึงเป็นโหมดที่นิยมสร้างขึ้นเป็นฐานเวลาได้ จึงสามารถนำไทเมอร์นี้มาใช้เพื่อกำหนดบอดเรทได้ด้วย ตัวอย่างเช่นกำหนดให้ไทเมอร์ ตัวที่ 2 ทำหน้าที่ในโหมด 2 จะต้องกำหนดค่าเข้าสู่ ริจิสเตอร์ TCON ดังตัวอย่าง

MOV TMOD,#0010XXXXB

ค่า X หมายถึงค่าอะไรก็ได้เพราะ ไม่ได้เกี่ยวข้องกับไทเมอร์ตัวที่ 1

รูปแบบทั่วไปในการหาบอดเรท ในโหมด 1 สามารถหาได้ตามสูตร

BAUD RATE = TIMER1 OVERFLOW RATE / 32

ตัวอย่างเช่น ถ้าต้องการ บอดเรท เท่ากับ 1200 บิตต่อวินาที สามารถคำนวณค่าความถี่ โอเวอร์โฟลว์ของไทเมอร์ ได้ดังนี้

1200 = Timer 1 Overflow Rate / 32

จะได้ Timer Overflow เท่ากับ 38.4 kHz

ถ้าระบบ MCS-51 ใช้ความถี่สัญญาณนาฬิกาจากคริสตัล เท่ากับ 12 MHz ตัวไทมเมอร์ตัวที่ 2 จะได้รับสัญญาณ นาฬิกา เท่ากับ 1 MHz หรือ 1000 kHz ถ้าเราต้องการ ไทเมอร์โอเวอร์โฟลว์ เท่ากับ 38.4 KHz ดังนั้นอัตราโอเวอร์โฟลว์ มีค่าเท่ากับ 1000 / 38.4 = 26.04 สัญญาณนาฬิกา โดยค่า โอเวอร์โฟลว์ จะเกิดขึ้นเมื่อเกิดการเปลี่ยนแปลง FFh เป็น 00h ดังนั้นจะต้องให้ไทเมอร์ นับไป 26 ครั้ง ดังนั้นค่าที่จะให้รีจิสเตอร์ TH1 มีค่าเท่ากับ -26 ซึ่งใช้เป็นค่าโหลดเข้าสู่ไบต์สูง

```
MOV  TH1, #-26
```

ฉ.3 การอินเทอร์รัพท์

ในการทำงานของระบบคอมพิวเตอร์ ซึ่งรวมไปถึงไมโครคอนโทรลเลอร์ด้วยนั้น โดยทั่วไปมักมีอุปกรณ์ต่อพ่วงร่วมอยู่ด้วย ถ้าคอมพิวเตอร์ต้องการทำงานกับอุปกรณ์ภายนอกต้องทำการตรวจสอบอุปกรณ์ภายนอกเหล่านั้นเสมอ ตัวอย่างเช่นถ้าต้องการให้คอมพิวเตอร์พอร์ทหนึ่งต่ออยู่กับหลอด LED 7 ส่วน อีกพอร์ทหนึ่งต่อกับสวิตช์ ถ้าระบบของเราทำงานเป็นนาฬิกาแสดงทาง LED ถ้ามีการกดสวิตช์ ให้นาฬิกาหยุดเดิน เราอาจจะเขียนโปรแกรมให้ทำงานเป็นนาฬิการะหว่างที่นาฬิกาเดินไปให้คอยตรวจสอบสวิตช์ด้วยว่ามีการกดหรือยัง การทำงานแบบนี้เรียกกว่าวิธีการแบบโพลลิ่ง (Polling method) คือตัวไมโครโปรเซสเซอร์จะต้องคอยตรวจสอบอุปกรณ์อินพุตตลอดเวลา ว่ามีข้อมูลเข้ามาหรือยัง การทำงานแบบนี้ถ้ามีอุปกรณ์ภายนอกหลายตัวระบบต้องตรวจสอบอุปกรณ์ภายนอกทั้งหมดนั้นซึ่งจะทำให้เสียเวลาในการทำงานหลักไป นอกจากนี้ยังเป็นผลให้หน่วยประมวลผลกลางทำงานหนักอีกด้วย ในทางกลับกัน หากมีการกดสวิตช์เมื่อไหร่ให้นาฬิกาหยุดเดินทันที การทำงานแบบนี้นอกจากไม่เสียเวลาในการทำงานในการตรวจสอบอุปกรณ์ถ้าอุปกรณ์ภายนอกต้องการติดต่อกับหน่วยประมวลผล อุปกรณ์ภายนอกจะส่งสัญญาณมาบอกหน่วยประมวลผลเอง ระบบการทำงานแบบนี้เรียกว่า การอินเทอร์รัพท์

ถ้าหากคอมพิวเตอร์กำลังทำงานหลักอยู่ เมื่อมีการอินเทอร์รัพท์เข้ามาคอมพิวเตอร์จะละทิ้งโปรแกรมหลัก แต่ไปทำงานโปรแกรมตอบสนองการอินเทอร์รัพท์ (Interrupt service routine) เมื่อทำโปรแกรมตอบสนองอินเทอร์รัพท์เสร็จสิ้นแล้ว คอมพิวเตอร์จะกลับมาทำงานตามเดิม

แหล่งกำเนิดสัญญาณอินเทอร์รัพท์ที่ใช้กับ MCS-51 มีสองชนิดคือ อินเทอร์รัพท์ภายนอก และ อินเทอร์รัพท์ภายใน โดยการอินเทอร์รัพท์ภายในจะเกิดขึ้นภายในตัว MCS-51 เอง ได้แก่

สัญญาณไทเมอร์แฟลก 0 (TF0) ไทเมอร์แฟลก 1 (TF1) และพอร์ทอนุกรม ซึ่งในงานวิจัยนี้จะใช้งานเพียงการอินเทอร์รัพท์จากภายในเท่านั้น อนึ่งการที่จะทำให้ MCS-51 รองรับการอินเทอร์รัพท์นั้นต้องมีการกำหนดค่าให้แก่รีจิสเตอร์ IE (Interrupt enable) และหากว่าต้องการจัดลำดับว่าจะให้การอินเทอร์รัพท์ใดเกิดก่อน สามารถกำหนดได้โดยกำหนดค่าได้ในรีจิสเตอร์ IP (Interrupt priority) โดยริจิสเตอร์ทั้งสองตัวมีรายละเอียดดังนี้

ตาราง ฉ.7 บิตต่างๆ ของรีจิสเตอร์ IE

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IE.7	EA	AFh	ถ้าเซต ยอมให้มีการอินเทอร์รัพท์
IE.6	-	AEh	ไม่ใช้งาน
IE.5	ET2	ADh	ยอมให้มีการ อินเทอร์รัพท์จาก Timer2 (8052)
IE.4	ES	ACh	ยอมให้มีการ อินเทอร์รัพท์จาก พอร์ทอนุกรม
IE.3.	ET1	ABh	ยอมให้มีการ อินเทอร์รัพท์จาก Timer1
IE.2	EX1	AAh	ยอมให้มีการ อินเทอร์รัพท์จาก INT1
IE.1	ET0	A9h	ยอมให้มีการ อินเทอร์รัพท์จาก Timer0
IE.0	EX0	A8h	ยอมให้มีการ อินเทอร์รัพท์จาก INT0

ในส่วนของรีจิสเตอร์ IP นั้น สามารถเข้าถึงในระดับบิตได้ ในการจัดลำดับความสำคัญของการอินเทอร์รัพท์ได้สองลำดับคือ 1 สำหรับความสำคัญสูงสุด และ 0 สำหรับความสำคัญน้อยที่สุด แต่ถ้าทำการกำหนดให้เป็น 1 ทั้งหมด ความสำคัญจะเป็นดังตาราง ฉ.8

ตาราง ฉ.8 ลำดับของการอินเทอร์รัพท์ตามค่าเริ่มต้น

ลำดับ	อินเทอร์รัพท์
1 (สูงสุด)	IE0
2	TF0
3	IE1
4	TF1
5 (ต่ำสุด)	Serial Port

ตาราง ฉ.9 บิตและหน้าที่ต่างๆ ของรีจิสเตอร์ IP

บิต	ชื่อบิต	ตำแหน่งบิต	รายละเอียด
IP.7	-	-	ไม่ใช้งาน
IP.6	-	-	ไม่ใช้งาน
IP.5	PT2	0BDh	ใช้กับ Timer2 (8052)
IP.4	PS	0BCh	ใช้กับพอร์ทอนุกรม
IP.3	PT1	0BBh	ใช้กับ Timer1
IP.2	PX1	0BAh	ใช้กับการอินเทอร์รัพท์จาก INT1
IP.1	PT0	0B9h	ใช้กับ Timer0
IP.0	PX0	0B8h	ใช้กับอินเทอร์รัพท์ จาก INT0

เมื่อไมโครคอนโทรเลอร์ถูกอินเทอร์รัพท์ จะต้องกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์โดยตำแหน่งที่จะกระโดดเรียกว่า อินเทอร์รัพท์เวกเตอร์ (Interrupt vectors) เมื่อทำโปรแกรมตอบสนองการอินเทอร์รัพท์เรียบรอยแล้ว MCS-51 จะกระโดดมายังตำแหน่งเดิม โดยก่อนที่จะกระโดดไปทำโปรแกรมตอบสนองการอินเทอร์รัพท์จะต้องเก็บค่าตำแหน่งเดิมไว้ โดยเก็บค่าอินเทอร์รัพท์เวกเตอร์ของ MCS-51 แสดงดังตาราง ฉ.10

ตาราง ฉ.10 อินเทอร์รัพท์เวกเตอร์ของอินเทอร์รัพท์ต่างๆ

อินเทอร์รัพท์	อินเทอร์รัพท์เวกเตอร์
System Reset	0000h
External0	0003h
Timer0	000Bh
External1	0013h
Timer1	001Bh
Serial Port	0023h
Timer2	002Bh

ในการเขียนโปรแกรมหลัก (Main program) จะต้องกำหนดว่าจะให้ MCS-51 ถูกอินเทอร์รัพท์ด้วยอะไร และจะให้ MCS-51 ถูกอินเทอร์รัพท์ได้หรือไม่โดยการโปรแกรมค่าต่างๆใน IE รีจิสเตอร์ถ้ามีการอินเทอร์รัพท์จากสองแหล่งขึ้นไปควรมีการจัดลำดับความสำคัญในรีจิสเตอร์ IP ดังนั้นในโปรแกรมหลักจะต้องมีการกำหนดค่า ใน IE เสมอ สำหรับโปรแกรมตอบสนองการอินเทอร์รัพท์ถือว่าเป็นโปรแกรมย่อยโปรแกรมหนึ่งจะต้องจบโปรแกรมด้วยค่า RETI (Return from interrupt)

ภาคผนวก ช

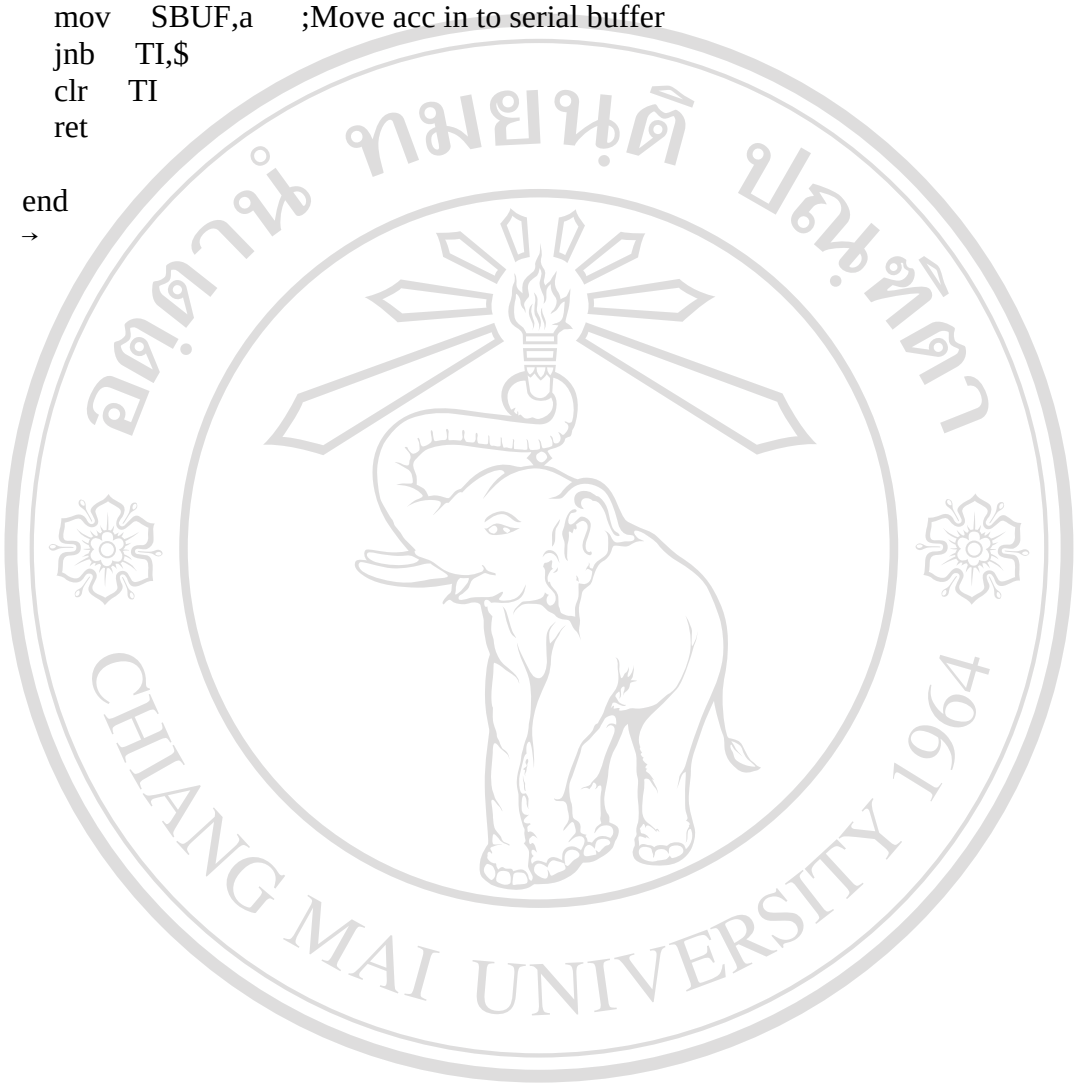
ซอร์สโค๊ดของไมโครคอนโทรเลอร์

```
; Program SERVO02.ASM
; This version uses interrupt from timer for frame time.
; Original code byThawan Sucharitakul 26 May 2006
; Re-edit for ETT-T89C51AC2 by Kongsak Dejkoonmak
; RS-232 to servo pulse converter
;
; Serial port 9,600 bps, no parity, 8 bit data, one stop bit
;
; Servo command digit 0-9 for position control 5=center
;         Each count from 5 = 10 degree displacement
;
; Servo pulse command connected to cpu port P1.0
;
; MEMORY MAP
;   Register banks           00 - 1F
;   Bit-addressable variables   20 - 2F
;   Byte variables           30 - 4F
;   Stack area               50 - 7F
;   8255 base address          8000H

    temp    equ 30H     ;Temporary variable
    width   equ temp+1  ;Pulse width in 0.1 ms
    org     0000h       ;Reset address
    jmp     MAIN
    org     000Bh       ;Timer 0 interrupt vector
    jmp     T0INT       ;Jump to timer 0 interrupt service
    org     0100h       ;Main program start address
MAIN:
    ANL     CKCON,#11111110B      ;Set CPU at Standart Speed
    mov     SP,#4Fh     ;Set stack pointer
    call    INIINT      ;Initiate timer interrupt
    call    SENDREADY   ;Prompt hoste
    mov     width,#0    ;Zero servo
  LOOP:
    jnb     RI,$
```

```
    call    GETPULSE
    jmp     LOOP


T0INT:      ;Timer 0 interrupt service
;   Set interrup period to 20 millisecond = 21701 cycles
;   Load timer register with FFFFH-21701 = AB3AH
    clr     TR0     ;Stop timer 0
    mov     TH0,#0ABH   ;Load timer 0 high byte
    mov     TL0,#3AH    ;Load timer 0 low byte
    setb    TR0         ;Start the clock
    push    acc         ;Save accumulator
    push    PSW         ;Save program status word
    setb    RS0         ;Select register bank 1
    call    SENDPULSE   ;Send servo pulse
    pop     PSW         ;Restore program status word
    pop     acc         ;Restore accumulator
    reti    Return from interrupt

GETPULSE:           ;Get servo pulse width from serial port
    clr     RI      ;Clear receive flag
    mov     a,SBUF  ;Read serial buffer
    mov     temp,a  ;Save value
    clr     cy      ;Clear carry flag
    subb    a,#'0'  ;Less than zero?
    jb      cy,GETPULSE1    ;Get out if less than 0
    clr     cy
    mov     a,#'9'
    subb    a,temp  ;Larger than 9?
    jb      cy,GETPULSE1    ;Get out if > 9
    mov     a,temp
    clr     cy
    subb    a,#'0'  ;Change ASCII to binary
    mov     width,a ;Save pulse width
  GETPULSE1:
    ret

  SENDPULSE:
    setb    P1.0    ;Servo control bit high
    call    WAIT1MS ;Wait 1 ms
    mov     a,width ;Get pulse width
    jnz     SENDPULSE2  ;Get out if zero
    clr     P1.0    ;Servo control bit low
    ret
  SENDPULSE2:
    mov     r1,width    ;Set up loop
  SENDPULSE1:
```

```
   call    WAIT01MS    ;0.1 millisecond for each loop
   djnz    r1,SENDPULSE1   ;Do it again
   clr     P1.0    ;Servo control bit low
   ret

WAIT1MS:    ;One millisecond delay loop 1085 cycles
   mov     r2,#2   ;Outer loop
 WAIT1MS1:
   mov     r3,#0FFh    ;Inner loop
 WAIT1MS2:
   djnz    r3,WAIT1MS2
   djnz    r2,WAIT1MS1
   mov     r3,#29  ;29 more loops to make 1085 cycles
   djnz    r3,$
   ret

WAIT01MS:    ;0.1 millisecond delay loop 109 cycles
   mov     r2,#53   ;53 loops compensating for call and return
   djnz    r2,$
   ret

INIINT:
   mov     TMOD,#21h       ;Timer 1 mode 2, timer 0 mode 1
   mov     SCON,#01010000b ;Serial port mode 1
   mov     TH1,#0FDH       ;Baud rate 9600
   setb    TR1             ;Run timer 1
   setb    TR0             ;Run timer 0
   setb    ET0             ;Enable timer 0 interrupt
   setb    EA              ;Enable interrupts
   ret

SENDREADY:
   mov     DPTR,#READY
   call    SENDMES
   ret

READY:
   db  'SERVO INTERFACE V 2.0 READY PRESS KEY 0-9',13,10,0

SENDMES:
   clr     a           ;Clear accumulator
   movc    a,@a+DPTR   ;Read code data
   jz      SENDMES1    ;Last charactor is zero
   call    SENDCHR     ;Send it
   inc     DPTR        ;Next letter
   jmp     SENDMES     ;Do it again
 SENDMES1:
```

```
   ret

SENDCHR:    ;Send a charector out the serial port
   mov     SBUF,a      ;Move acc in to serial buffer
   jnb     TI,$
   clr     TI
   ret

end
→
```



ภาคผนวก ซ

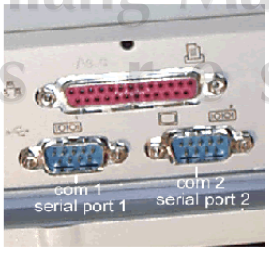
การติดต่อผ่านพอร์ท ต่างๆของคอมพิวเตอร์ส่วนบุคคล

เนื่องจากการทำงานการวิจัยนี้ ใช้คอมพิวเตอร์ส่วนบุคคลเพื่อทำการคำนวณทั้งในส่วนประมวลผลภาพและ ส่วนของเครือข่ายประสาทเทียมที่ใช้ในการตัดสินใจควบคุมรถ และเพื่อสัญญาณออกไปยังชุดควบคุมจำเป็นที่จะต้องมีความเข้าใจพื้นฐานเกี่ยวกับการติดต่อคอมพิวเตอร์ส่วนบุคคล ซึ่ง ในที่นี้จะกล่าวถึงการติดต่อผ่านพอร์ทที่ใช้ในงานวิจัยนี้ คือ พอร์ทอนุกรม (RS-232C) และ พอร์ทขนาน (Line printer port, LPT)

ซ.1 การติดต่อผ่านพอร์ทอนุกรม

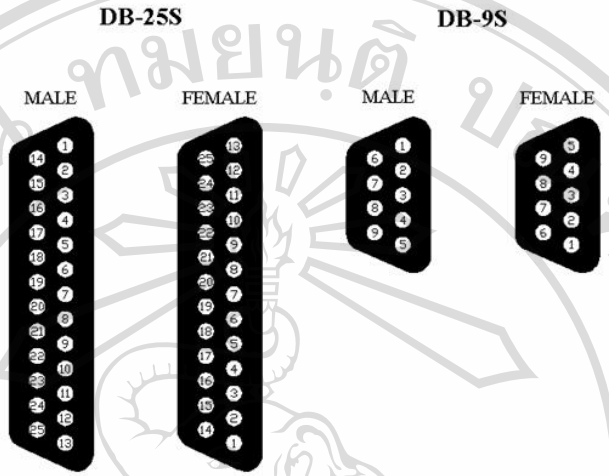
คอมพิวเตอร์ส่วนบุคคลเกือบแทบทุก ๆ เครื่องจะต้องมีพอร์ทอนุกรม และสนับสนุนการทำงานตามมาตรฐาน RS-232C และมาตรฐาน RS-232C เป็นมาตรฐานที่ออกแบบมาเพื่อที่จะให้อุปกรณ์ต่อพ่วงจากผู้ผลิตต่างๆ ใช้งานร่วมกันได้ และเป็นมาตรฐานที่ใช้งานอย่างแพร่หลายและกว้างขวาง อีกทั้งผู้ผลิตบอร์ดชุดไมโครคอนโทรลเลอร์จำนวนมากนิยมที่จะมีพอร์ทนี้เพื่อใช้ในการติดต่อกับคอมพิวเตอร์ส่วนบุคคล มาตรฐาน RS-232C เป็นมาตรฐานที่ประกาศใช้ครั้งแรกในปี ค.ศ. 1969 โดยสมาคมอุตสาหกรรมอิเลคทรอนิคส์ (Electronic industries association, EIA)

คอนเน็กเตอร์ (Connector) ที่นิยมใช้ในปัจจุบันคือ ชนิด D แบบ 9 ขา และ 25 ขา โดยส่วนมากแล้วจะอยู่บริเวณด้านหลังคอมพิวเตอร์ส่วนบุคคล มีค่าแรงดันระหว่าง -3 ถึง -15 โวลต์ สำหรับโลจิก 1 และ มีระดับแรงดันระหว่าง +3 ถึง +15 โวลต์ สามารถส่งสัญญาณโดยใช้สายสัญญาณที่มีความยาวได้สูงสุดประมาณ 150 เมตร



รูป ซ.1 คอนเน็กเตอร์ [www.thaiio.com]

คอนเน็กเตอร์แบบ D นั้นมีชื่อเรียกโดยทั่วไปว่า DB9 สำหรับ คอนเน็กเตอร์ แบบ 9 ขา และ DB25 สำหรับคอนเน็กเตอร์ แบบ 25 ขา ซึ่งคอนเน็กเตอร์ทั้ง 2 แบบนั้นไม่มีความแตกต่างในการทำงานแต่อย่างใด แต่มีการจัดเรียงขาต่างกัน



รูป ซ.2 คอนเน็กเตอร์ ทั้ง 2 แบบ [www.machinetoolhelp.com]

ตาราง ซ.1แสดงรายละเอียดของคอนเน็กเตอร์ แบบ D

D-Type 25 Pin	D-Type 9 Pin	สัญลักษณ์	ชื่อสัญญาณ
Pin 2	Pin 3	TD	Transmit data
Pin 3	Pin 2	RD	Receive data
Pin 4	Pin 7	RTS	Request to send
Pin 5	Pin 8	CTS	Clear to send
Pin 6	Pin 6	DSR	Data set ready
Pin 7	Pin 5	SG	Signal ground
Pin 8	Pin 1	CD	Carrier detect
Pin 20	Pin 4	DTR	Data Tterminal ready
Pin 22	Pin 9	RI	Ring indicator

สำหรับรายละเอียดของสายสัญญาณนั้นประกอบด้วย

Transmit Data ใช้สำหรับส่งข้อมูลออกจากคอมพิวเตอร์

Receive Data ใช้สำหรับรับข้อมูลเข้ามายังคอมพิวเตอร์

Request To Send ใช้สำหรับส่งข้อมูลไปยังอุปกรณ์ปลายทางเพื่อร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลกลับมา

Clear To Send	ใช้สำหรับตรวจสอบอุปกรณ์ที่เชื่อมต่อด้วยพร้อมที่จะรับข้อมูลหรือไม่ โดยจะคอยรับสัญญาณ RTS เมื่อทุกอย่างพร้อมก็จะทำการส่งข้อมูลออกทางขา TD
Data Set Ready	ใช้สำหรับตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง จะใช้คู่กับขา DTR
Signal Ground	เป็นกราวนด์ของระบบ
Carrier Detect	ขานี้จะมีสถานะ Active เมื่อมีการส่งสัญญาณ Carrier เข้ามา
Data Terminal Ready	ใช้สำหรับบอกให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อด้วย โดยขา DTR นี้ต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง
Ring Indicator	ขานี้จะมีสถานะ Active เมื่อได้รับสัญญาณเรียกเข้า (จากโทรศัพท์)

การติดต่อแบบอนุกรมที่นิยมใช้กับคอมพิวเตอร์นั้น เป็นการสื่อสารแบบอะซิงโครนัส (Asynchronous) นั่นคือต้องใช้สายเดียวทำหน้าที่ ส่งข้อมูลและควบคุมการส่งข้อมูล ดังนั้นข้อมูลที่อ่านได้แต่ละบิตจากการส่งแบบอนุกรมจึงต้องแยกแยะว่าใช้สำหรับวัตถุประสงค์ใดโดยสามารถแยกออกได้เป็น 4 ส่วนคือ

1. บิตเริ่มต้น (Start Bit)	ขนาด 1 บิต
2. บิตข้อมูล (Data Character)	ขนาด 7 หรือ 8 บิต
3. บิตพาริตี (Parity Bit)	ขนาด 1 บิต
4. บิตจบ (Stop Bit)	ขนาด 1 หรือ 2 บิต

ในการส่งข้อมูลแต่ละครั้งต้องส่วนประกอบ ทั้ง 4 ส่วนเสมอ ยกเว้น ส่วนที่ 3 คือบิตพาริตี้ ซึ่งจะมีหรือไม่ก็ได้ โดยพอจะสรุปหน้าที่พอสังเขปได้ดังนี้

1. บิตเริ่มต้น จะใส่ที่จุดเริ่มต้นเสมอ เพื่อเตือนอุปกรณ์ฝ่ายรับว่าข้อมูลกำลังจะมาถึง

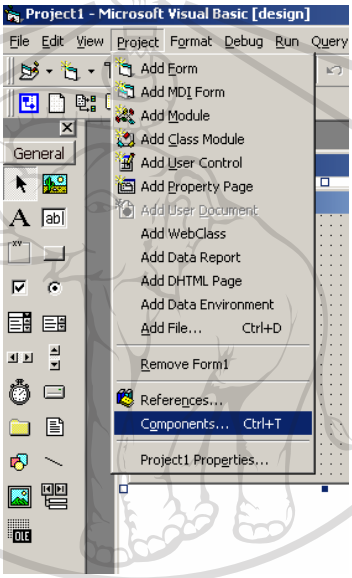
2. บิตข้อมูล เป็นส่วนของข้อมูล โดยจะส่งข้อมูล เป็นจำนวน 7 หรือ 8 บิต ซึ่งเพียงพอตามหลัก ASCII

3. บิตพาริตี เป็นการตรวจสอบความถูกต้องของข้อมูลที่ส่ง เมื่อทำการกำหนดค่าบิตพาริตีนี้เข้าไปต้องทราบทั้งตัวรับและตัวส่งว่าเป็นบิตพาริตีแบบไหน ซึ่ง อาจจะเป็นทั้งแบบ บิตพาริตีคู่ ซึ่งบิตนี้จะมีค่าเป็น 1 เมื่อข้อมูลที่ส่งออกไปทั้งหมดมีค่าเป็นจำนวนคู่ หรือ บิตพาริตีคี่ ซึ่งบิตนี้จะมีค่าเป็น 1 เมื่อข้อมูลที่ส่งออกไปทั้งหมดมีค่าเป็นจำนวนคี่ หรือ ไม่มีบิตพาริตี

4. บิตจบ เป็นบิตส่งเพื่อปิดท้ายข้อมูล

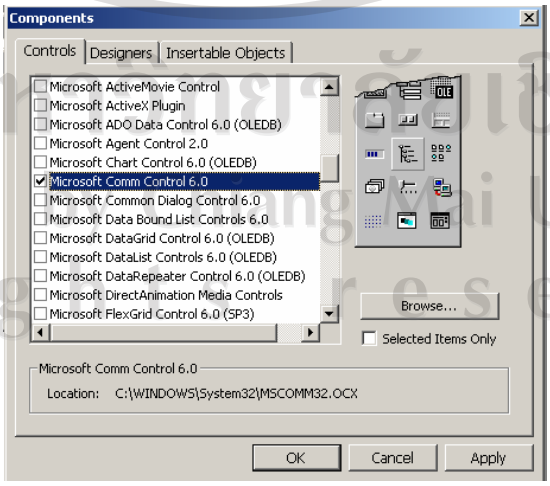
และเนื่องจากเป็นการส่งสัญญาณแบบอะซิงโครนัส ดังนั้นจึงต้องมีการกำหนดความเร็วในการสื่อสาร หรือเรียกว่า บอดเรต (Baud rate) มีหน่วยเป็นบิตต่อวินาที ซึ่งตามมาตรฐาน RS-232C แล้วความเร็วในการสื่อสารนี้จะมีใช้ดังนี้ 110, 150, 300, 600, 1200, 2400, 4800, 9600 และ 19200 บิตต่อวินาที

ในการเขียนโปรแกรมเพื่อติดต่อกับพอร์ทอนุกรม โดยใช้ภาษา VB นั้นต้องมีการเพิ่มคอนโทรล MSComm เข้าไปใน ส่วนของ Toolbox เพื่อทำการเรียกใช้งานโดยสามารถเพิ่มได้โดย เข้าไปที่ เมนู Project



รูป ซ.3 เลือกโปรเจค

ทำการเลือก Microsoft Comm Control 6.0 จากไดอะล็อก เมื่อเลือกเสร็จแล้ว จะปรากฏคอนโทรลรูป โทรศัพท์ปรากฏเพื่อให้เลือกใช้งาน



รูป ซ.4 คอนโทรล MsComm

ดังกล่าวมาข้างต้นในการติดต่อพอร์ทอนุกรมนั้นต้องทำการกำหนดค่าเริ่มต้น คือบอดเรท และ ลักษณะข้อมูลที่จะส่งออก ซึ่งกำหนดโดยคำสั่งดังต่อไปนี้ กำหนดให้ ออปเจคต์ คือชื่อของออปเจคต์ ที่กำหนดให้เป็น คอนโทรล MSComm

Object.CommPort = [value]

โดยคำสั่งนี้จะทำการกำหนดให้ว่าจะทำการติดต่อกับพอร์ทอนุกรมที่ 1 หรือ 2 ตัวอย่างเช่น MSComm1.ComPort = 1

Object.Setting = “Baud Rate, Parity, Data Bit, Stop Bit”

โดยคำสั่งนี้จำทำการกำหนดว่าจะใช้การติดต่อโดยมี บอดเรทเท่าใด พาริตี้แบบใด จำนวนบิตข้อมูล กี่บิต และมีบิตปิดท้ายกี่บิต ตัวอย่างเช่น MSComm1.Setting =”9600, N, 8, 1” เมื่อกำหนดให้มีบอดเรท 9600 บิตต่อวินาที ไม่มีพาริตี้ ข้อมูลจำนวน 8 บิต และบิตจบ 1 บิต

Object.PortOpen = [Boolean]

โดยคำสั่งนี้จะทำการกำหนดว่าจะให้พอร์ทอนุกรมนี้ทำการเปิดเพื่อการใช้งานหรือไม่ ตัวอย่างเช่น MSComm1.PortOpen = True เมื่อต้องการเปิดพอร์ทเพื่อการใช้งาน

สำหรับการติดต่อผ่านพอร์ทนั้นจะสามารถกระทำได้ก็ต่อเมื่อ ทำการกำหนดค่าต่าง ๆ ดังกล่าวมาข้างต้นแล้วเท่านั้น และเมื่อกำหนดค่าต่างๆดังฟังก์ชันข้างต้นแล้ว สามารถส่งข้อมูลออกและเข้าได้โดยคำสั่ง

Object.Output = [Value]

โดยค่าที่ส่งออกไปนี้จะเป็นไปตามที่กำหนดไว้ข้างต้น เช่น MSComm1.Output = &HFF เพื่อการส่งค่า 255 ออกไปจากพอร์ทอนุกรม ซึ่งค่าที่ส่งออกไปนี้ต้องไม่เกินขนาดบิตที่กำหนดไว้ก่อนหน้านี้

Data = Object.Input

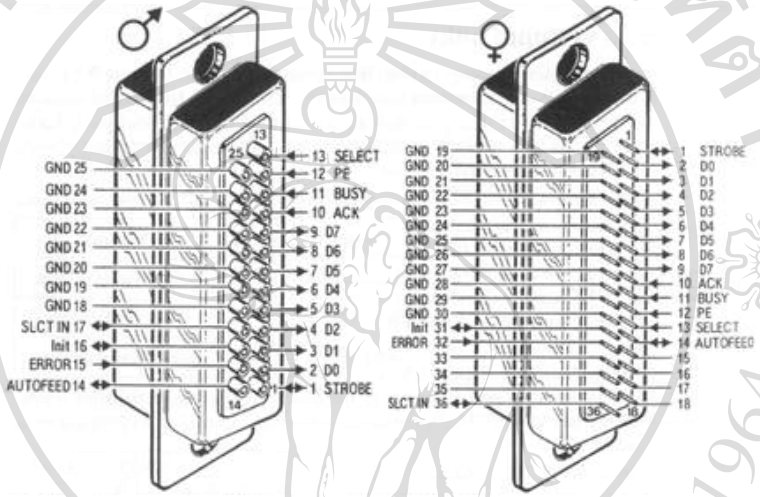
โดยคำสั่งนี้จะเป็นการรับข้อมูลเข้ามา เมื่อ Data คือตัวแปรที่ประกาศไว้ก่อนแล้วเพื่อ รับค่าจากบัฟเฟอร์พอร์ทขนาน

3.3.2 การติดต่อผ่านพอร์ทขนาน

พอร์ทขนานหรือ LPT นั้นเดิมเป็นพอร์ทที่ใช้ติดต่อกับพรินเตอร์เป็นหลัก โดยพอร์ทขนานนั้นสามารถติต่อได้ทั้งหมด 8 บิตในครั้งเดียว ลักษณะทั่วไปของของพอร์ทขนานนี้จะเป็นแบบ D-type 25 pin ซึ่งติดตั้งอยู่ด้านหลังของคอมพิวเตอร์ พอร์ทขนานนี้จะให้แรงดันในแบบ

TTL คือให้แรงดัน 0 โวลต์ สำหรับโลจิก “0” และแรงดันขนาด 5 โวลต์ สำหรับโลจิก “1” ซึ่งแรงดันในแบบ TTL นี้เป็นส่วนที่ทำให้ง่ายต่อการประยุกต์ใช้ในการติดต่องานอื่นๆ และง่ายต่อการออกแบบเพื่อใช้กับอุปกรณ์ประเภท TTL อีกด้วย อย่างไรก็ตามการติดต่อผ่านพอร์ทขนานนี้มีข้อเสียคือไม่สามารถติดต่อได้ไกลมากนักเนื่องจากแรงดันไม่สม่ำเสมอ

พอร์ทขนานมีเส้นสัญญาณทั้งหมด 17 เส้นสัญญาณ แบ่งได้เป็น 3 กลุ่ม คือพอร์ทข้อมูล (Data port) จำนวน 8 เส้นสัญญาณ, พอร์ทสถานะ (Status port) จำนวน 5 เส้นสัญญาณ, พอร์ทควบคุม (Control port) จำนวน 4 เส้นสัญญาณ



รูป ซ.5 พอร์ทขนาน [lpt.hw.cz]

แม้ว่าพอร์ทขนานจะมีเส้นสัญญาณถึง 17 เส้นสัญญาณ แต่ในงานวิจัยนี้จะใช้เพียงเส้นสัญญาณข้อมูลเพียง 1 เส้นสัญญาณเท่านั้น

ในการติดต่อผ่านพอร์ทขนานหรือ LPT นั้น โดยความสามารถของ VB เองไม่สามารถติดต่อผ่านพอร์ทนี้ได้โดยตรง ดังนั้นจึงจำเป็นต้องมีชุดโปรแกรม API ที่เขียนขึ้นเพื่อทำการติดต่อกับพอร์ทขนาน ในงานวิจัยนี้ได้ใช้ชุด API ที่ชื่อ Inpout32.dll ซึ่งเป็นไฟล์ DLL ที่พัฒนาโดยภาษา C++

สำหรับการติดตั้งไฟล์นี้เพื่อการใช้งานนั้น ให้ทำการคัดลอกไฟล์นี้ไปไว้ในโฟล์เดอร์ c:\\windows\system32 สำหรับระบบปฏิบัติการ Windows XP (ในกรณีที่ ไดร์ฟ C คือ ไดร์ฟที่ทำการติดตั้งระบบปฏิบัติการ) และเมื่อทำการติดติดตั้งเสร็จแล้ว VB จะสามรถทำการติดต่อผ่านพอร์ทขนานนี้ได้โดยการประกาศ API โดยฟังก์ชันที่จะทำการส่งข้อมูลที่จะทำการติดต่อข้อมูลมีโครงสร้างคือ

Out *portnumber,data*

โดยที่

Port number จะเป็นตัวเลขระบุตำแหน่งของพอร์ท ซึ่งในที่นี้คือ 378h สำหรับ LPT1 และ 278h สำหรับ LPT2

data คือจำนวนขนาด 8 บิต กล่าวคือ 0 ถึง 255

ในส่วนของ API นี้มีข้อจำกัดในเรื่องการติดต่อกับพอร์ทขนาน คือสามารถส่งสัญญาณออกได้เพียงเส้นสัญญาณ D0 ถึง D7 เท่านั้น แต่อย่างไรก็ตามการติดต่อนี้ถือว่าเพียงพอสำหรับงานวิจัยนี้

ตาราง ซ.2 สรุปลักษณะสัญญาณของพอร์ทขนานทั้งหมด

หมายเลขขา	ชื่อสัญญาณ	บิต	ทิศทางข้อมูล (เข้า/ออก)
1	nStrobe	-C0	ข้อมูลออก
2	Data 0 (Bit 0)	D0	ข้อมูลออก
3	Data 1 (Bit 1)	D1	ข้อมูลออก
4	Data 2 (Bit 2)	D2	ข้อมูลออก
5	Data 3 (Bit 3)	D3	ข้อมูลออก
6	Data 4 (Bit 4)	D4	ข้อมูลออก
7	Data 5 (Bit 5)	D5	ข้อมูลออก
8	Data 6 (Bit 6)	D6	ข้อมูลออก
9	Data 7 (Bit 7)	D7	ข้อมูลออก
10	nAck	S6	ข้อมูลเข้า
11	Busy	-S7	ข้อมูลเข้า
12	PaperEnd	S5	ข้อมูลเข้า
13	Select	S4	ข้อมูลเข้า
14	nAutoFeed	-C1	ข้อมูลออก
15	nError	S3	ข้อมูลเข้า
16	nInitialize	C2	ข้อมูลออก
17	nSelecPrinter	-C3	ข้อมูลออก
18 -25	Ground		

ประวัติผู้เขียน

ชื่อ	นาย คงศักดิ์ เดชคุณมาก
วัน เดือน ปี เกิด	29 ธันวาคม 2523
ประวัติการศึกษา	สำเร็จการศึกษาชั้นมัธยมศึกษาตอนปลายจาก โรงเรียนสามัคคีวิทยาคม จังหวัดเชียงราย ปีการศึกษา 2541 สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมเครื่องกล มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2545