รายการอ้างอิง

1. Ben G. Streetman, Sanjay Banerjee. Solid state electronic devices. fifth edition. ISBN 0-13-026101-7. Prentice Hall. 2000
2. ธนภัทร ภูรีพิทักษ์. อุปกรณ์อิเล็กทรอนิกส์และวงจร ภาคทฤษฎี. ISBN 974-389-090-4. สกายบุ๊กส์. 2545
3. LED information and technical data. [Online] Available from: http://www.theledlight.com/technical.html [2006, March 6]
4. History of LED technology.[Online] Available from: http://www.marktechopto.com/engineering/history.cfm [2006, March 6]
5. History of LEDs.[Online] Available from: http://www.newscenter.philips.com/About/news/article-15307.html[2006, March 6]
6. Light emitting diodes.[Online] Available from: http://wolfstone.halloweenhost.com/TechBase/litled_LightEmittingDiodes.html [2006, March 6]
7. Asian Stanley international co.,ltd. 2 colors dot matrix LED display board. 1994 (เอกสารไม่ตีพิมพ์เผยแพร่)
8. ชัยภัทร เลาหคุณากร, ปริญญา มานัสสถิตย์. แผงแสดงภาพไดโอดเปล่งแสงอเนกประสงค์. ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย. 2542 (เอกสารไม่ตีพิมพ์เผยแพร่)
9. มนปรีดี กัลยาศิริ, ฤณันต์ เอื้อวิจิตรอรุณ. ป้ายประชาสัมพันธ์อเนกประสงค์ชนิดจอภาพไดโอดเปล่งแสง. ภาควิชาวิศวกรรมไฟฟ้า จุฬาลงกรณ์มหาวิทยาลัย. 2542 (เอกสารไม่ตีพิมพ์เผยแพร่)
10. จตุชัย แพงจันทร์ , อนุโชต วุฒิพรพงษ์. เจาะระบบ Network. พิมพ์ครั้งที่ 1. ISBN 974-966-1-05-2. นนทบุรี : สำนักพิมพ์ IDC info, 2546.
11. สุวัฒน์ ปุณณชัยยะ , ต้น ตัณฑ์สุทธิวงศ์ , สุพจน์ ปุณณชัยยะ. เปิดโลก TCP/IP. พิมพ์ครั้งที่ 1. ISBN 974-7822-93-8. กรุงเทพฯ : สำนักพิมพ์ โปรวิชั่น : 2545
12. Internet protocols. [Online] Available from: http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/ip.pdf [2006, March 6]
13. รศ. มัณฑนา ปราการสมุทร. การเขียนชุดคำสั่งภาษาซี . พิมพ์ครั้งที่ 3. ISBN 974-88669-3-9. กรุงเทพฯ : สำนักพิมพ์ดวงกมล. 2534

14. Rolando Lopaz. Creating a TCP component in visual basic. [Online] Available from: http://www.15seconds.com/issue/990408.htm [2006, March 6]

15. Patrick K. Bigley. Advanced video capture. [Online] Available from: http://www.planet-source-code.com/vb/scripts/ShowCode.asp?txtCodeId=3883&lngWId=1 [2006, March 6]

16. Craig Hollabaugh,Ph.D. Embedded linux: hardware, software, and interfacing. ISBN 0-672-32226-9. Addison-Wesley .2002

17. Jamp labsheet.[Online] Available from: http://www.design-gateway.com/download/jamp/Labsheet_tutorial.pdf [2006, March 6]

18. CX82100 datasheethttp.[Online] Available from: http://www.tranzistoare.ro/datasheets/2300/500021_DS.pdf [2006, March 6]

19. Stephen Brown,Zvonko Vranesic. Fundamentals of digital logic with VHDL design. Second edition. ISBN 007-124482-4. McGRAW-HILL. 2005

20. Spartan 3 FPGA family : complete data sheet. [Online] Available from: http://direct.xilinx.com/bvdocs/publications/ds099.pdf [2006, March 9]

21. ISE 7 in depth tutorial. [Online] Available from: http://direct.xilinx.com/direct/ise7_tutorials/ise7tut.pdf [2006, March 9]

22. LED E1L5E_Sxxxx information.[Online] Available from: http://www.toyoda-gosei.com/led/products/pdf/E1L5E_Sxxxx_JEA.pdf [2006, March 9]

23. MBI5026 datasheet. [Online] Available from: http://www.mblock.com.tw/en_download_file/datasheet/MBI5026%20Datasheet%20VA.02.pdf [2006, March 9]

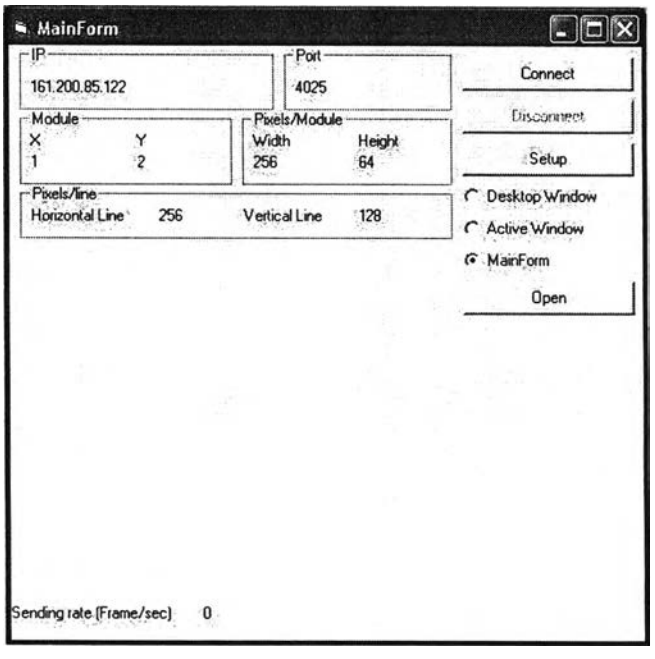
24. ปรัชญนันท์ นิลสุข. ทฤษฎีและการทำงานของเครื่องรับโทรทัศน์เบื้องต้น. ISBN 974-512-957-7. ซีเอ็ดยูเคชั่นจำกัด. 2541

ภาคผนวก

ภาคผนวก ก.

คู่มือการใช้งานระบบ

เมื่อจ่ายไฟให้กับส่วนควบคุมบนแผงแสดงภาพไดโอดเปล่งแสง ส่วนรับข้อมูลจะเริ่มทำงานเป็นเซิร์ฟเวอร์ เพื่อรอการติดต่อจากคอมพิวเตอร์ส่วนบุคคลที่ทำหน้าที่เป็นไคลเอนซ์ โดยโปรแกรมส่งข้อมูลที่ออกแบบบนคอมพิวเตอร์ส่วนบุคคลมีส่วนต่างๆ และวิธีการใช้งานมีดังนี้



ส่วนประกอบของโปรแกรมส่งข้อมูล

IP Frame บอกถึงหมายเลข IP เริ่มต้นของแผงแสดงภาพที่ต้องการส่ง โดย IP ในการส่งจะเรียงลำดับจากซ้ายไปขวา และจากบนลงล่าง ดังรูปด้านล่าง

Port Frame บอกหมายเลขพอร์ตที่ใช้ในการติดต่อไปยังเซิฟเวอร์

Module Frame บอกจำนวนโมดูลบนแผงแสดงภาพที่ใช้งาน

Pixels/Module Frame บอกขนาดจุดภาพของโมดูลที่ใช้งาน

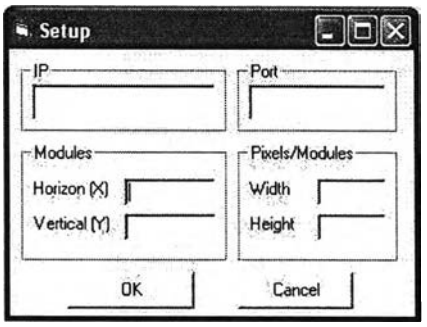
Pixels/line Frame บอกขนาดจุดภาพของแผงแสดงภาพที่ต้องการแสดง

Sending rate บอกความเร็วสูงสุดของการส่งภาพไปยังแผงแสดงภาพ

Connect Button ใช้ติดต่อไปยังส่วนรับข้อมูลบนแผงแสดงภาพที่มีหมายเลข IP และพอร์ตที่กำหนด หากไม่สามารถเชื่อมต่อกับส่วนรับข้อมูลได้จะแสดงข้อความ "Can't connect to board" เหนือ Sending rate

Disconnect Button ใช้ยกเลิกการติดต่อระหว่างส่วนส่งข้อมูลกับแผงแสดงภาพ

Set up Button ใช้เปลี่ยนหมายเลข IP , พอร์ตเชื่อมต่อ , จำนวนโมดูลในแนวนอนแนวตั้ง และขนาดจุดภาพของโมดูล เมื่อกดปุ่มจะมีหน้าจอด้านล่างขึ้นมาเพื่อให้ใส่ค่าที่ต้องการลงใหม่ หากไม่ใส่ค่าในช่องใด จะให้คงค่าเดิมไว้



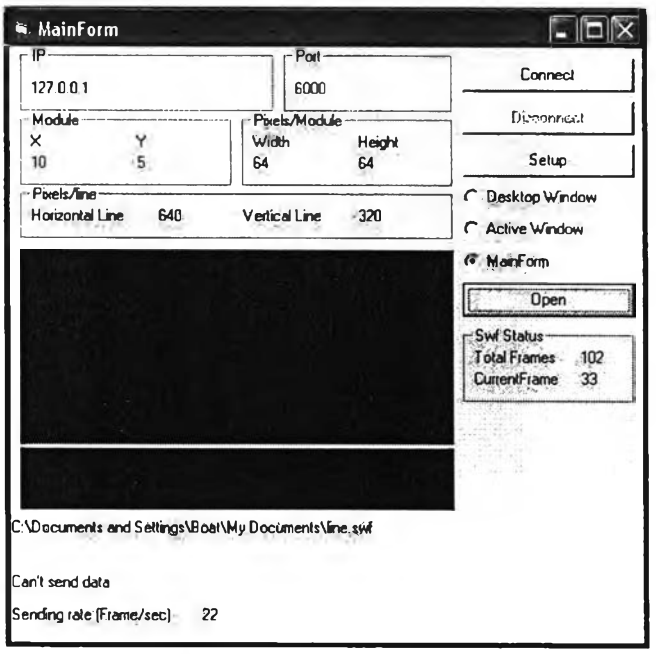
Option เป็นส่วนเลือกรูปแบบการทำงานของโปรแกรม โดยมีให้เลือกการทำงาน 3 แบบ โดยแบ่งเป็น

- Desktop Window จะเก็บภาพข้อมูลจาก desktop ของ window โดยเริ่มจากมุมซ้ายบน ไปทางขวาตามขนาดความยาวที่ต้องการให้แสดง และลงมาด้านล่างตามขนาดความสูงภาพที่ต้องการให้แสดง หากเลือกการทำงานในส่วนนี้ โปรแกรมจะเริ่มส่งข้อมูลทันทีที่เลือก

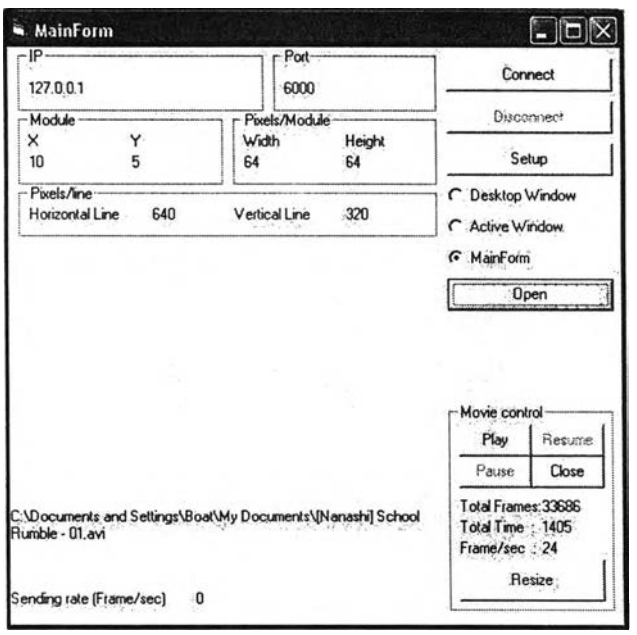
- Active Window จะเก็บภาพข้อมูลจาก window ที่ใช้งานอยู่ในปัจจุบัน โดยเก็บภาพจากมุมซ้ายบนไปทางขวาตามความยาว และลงล่างตามความสูงภาพเช่นกัน หากเลือกการทำงานในส่วนนี้ โปรแกรมจะเริ่มส่งข้อมูลทันทีที่เลือก

- Main Form จะเก็บภาพข้อมูลจากไฟล์ที่เปิดโดยสามารถเลือกเปิดไฟล์ได้ 3 รูปแบบ คือไฟล์รูป (jpg,bmp,gif) , ไฟล์ shockwaveflash (swf) และไฟล์หนัง (avi) โดยเมื่อเลือกไฟล์ที่ต้องการแสดงแล้ว จะแสดงในส่วนกลางของโปรแกรม หากไฟล์ที่เปิดเป็นไฟล์รูป หรือไฟล์ swf จะเริ่มส่งข้อมูลไปยังแผงทันทีที่เลือกไฟล์ หากเป็นไฟล์ avi จะส่งในภายหลัง การเปิดไฟล์ swf จะมี swf status frame แสดงขึ้นเพื่อบอกข้อมูลของ swf ที่แสดง และการเปิดไฟล์ avi จะมี Movie Control แสดงขึ้นเพื่อบอกข้อมูลและใช้ควบคุมการแสดงผลของไฟล์ avi นั้น

Swf Status Frame จะบอกจำนวนภาพทั้งหมด (Total Frames) ของไฟล์ swf ที่เปิด และบอกหมายเลขเฟรมที่แสดงอยู่ในปัจจุบัน (Current Frame)



Movie control Frame ทำหน้าที่ควบคุมไฟล์ avi ที่เปิด และบอกสถานะของไฟล์ โดยบอกจำนวนภาพทั้งหมด(Total Frames) ของไฟล์ เวลารวมของไฟล์ที่เปิด (Total Time) และความถี่ภาพของไฟล์ (Frames/sec) การควบคุมการแสดงไฟล์ avi จะมีส่วนต่างๆดังนี้



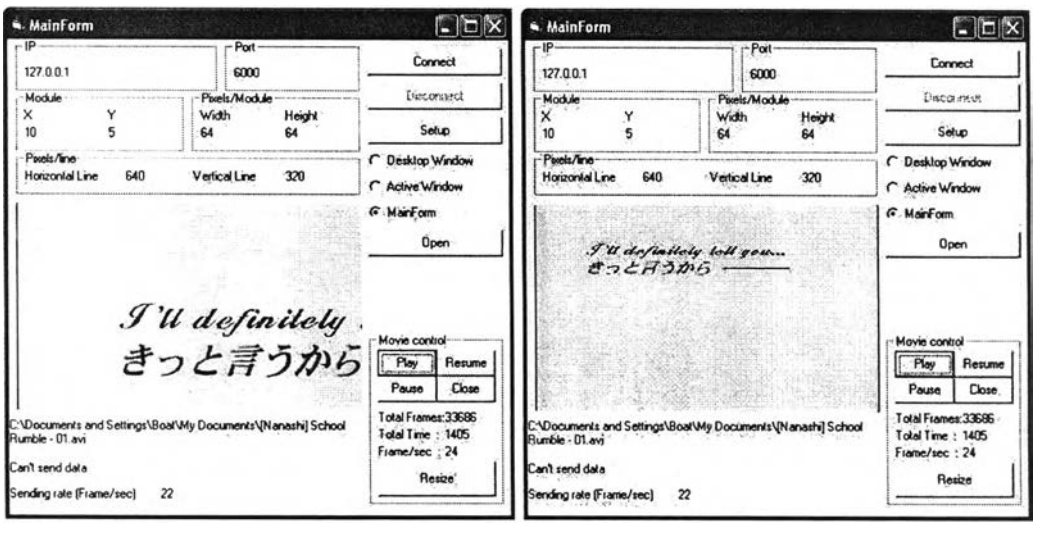
- Play เป็นคำสั่งเริ่มแสดงหนังจากส่วนต้นไฟล์ อีกทั้งเป็นส่วนให้เริ่มการส่งภาพไปยังแผงแสดงภาพในกรณีที่เลือกเปิดไฟล์ avi ด้วย

- Pause ใช้หยุดการแสดงผล ทำให้ภาพที่แสดงหยุดนิ่ง

- Resume ใช้แสดงภาพต่อจากส่วนที่หยุดไว้โดยปุ่ม Pause

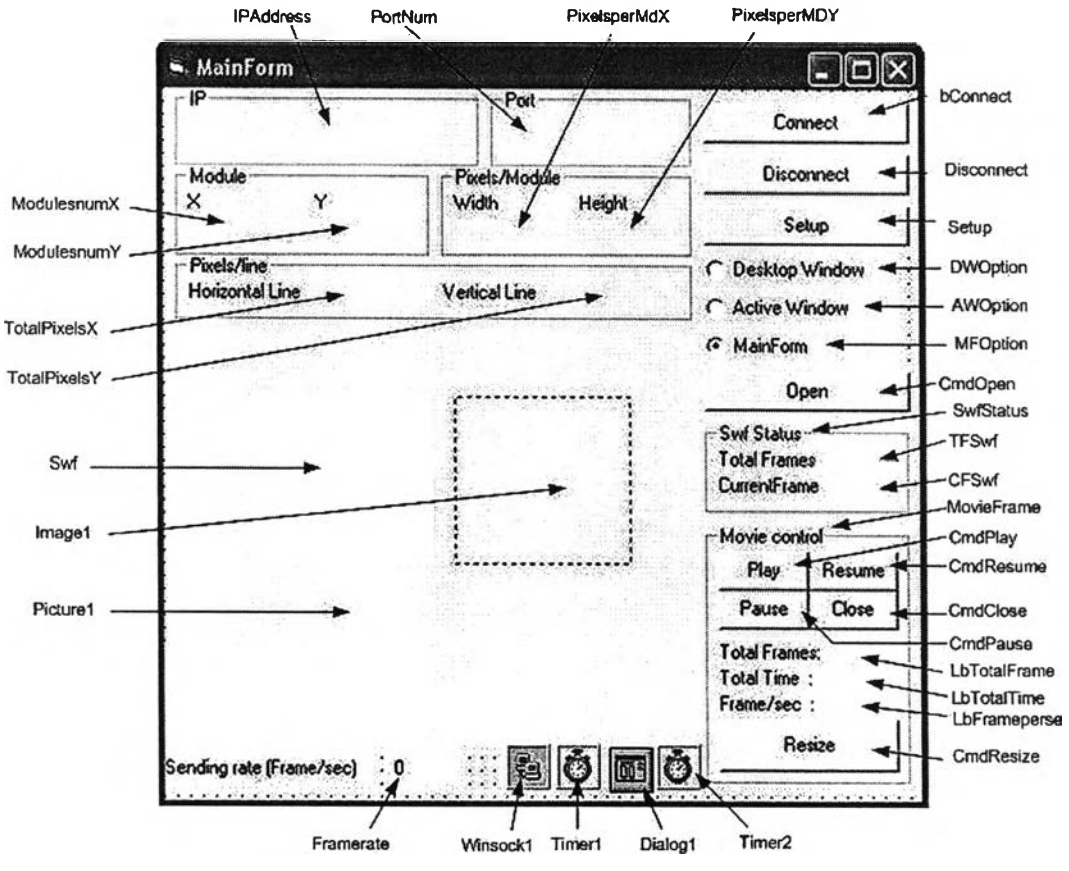
- Close ปิดไฟล์ภาพยนตร์ที่แสดงผลอยู่

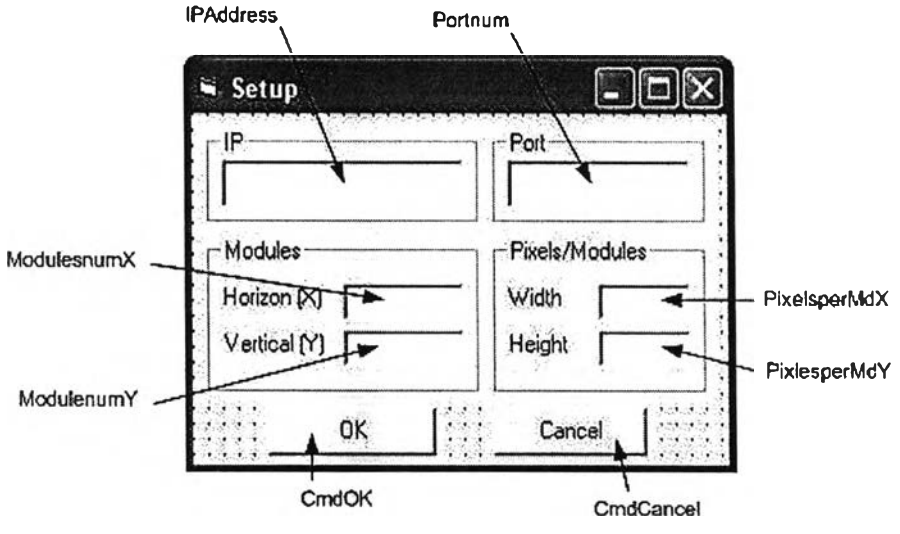
- Resize ใช้ปรับขนาดของภาพยนตร์ที่แสดงให้มีขนาดความกว้างและส่วนสูงให้เท่ากับขนาดของแผงแสดงภาพที่กำหนดบนโปรแกรมนี้ โดยผลของปุ่มนี้แสดงดังรูปด้านล่าง ภาพด้านขวาไม่มีการปรับขนาดภาพ ภาพซ้ายปรับขนาดแล้ว



ภาคผนวก ข.

โปรแกรม Visual Basic ในส่วนส่งข้อมูลบนคอมพิวเตอร์ส่วนบุคคล


IPAddress
PortNum
PixelsperMdX
PixelsperMDY
MainForm
IP
Port
Module
X
Y
Pixels/Module
Width
Height
Pixels/line
Horizontal Line
Vertical Line
Connect
Disconnect
Setup
Desktop Window
Active Window
MainForm
Open
Swf Status
Total Frames
CurrentFrame
Movie control
Play
Resume
Pause
Close
Total Frames:
Total Time :
Frame/sec :
Resize
Sending rate (Frame/sec)
0
bConnect
Disconnect
Setup
DWOption
AWOption
MFOption
CmdOpen
SwfStatus
TFSwf
CFSwf
MovieFrame
CmdPlay
CmdResume
CmdClose
CmdPause
LbTotalFrame
LbTotalTime
LbFrameperse
CmdResize
ModulesnumX
ModulesnumY
TotalPixelsX
TotalPixelsY
Swf
Image1
Picture1
Framerate
Winsock1
Timer1
Dialog1
Timer2


IPAddress
Portnum
Setup
IP
Port
Modules
Horizon (X)
Vertical (Y)
Pixels/Modules
Width
Height
OK
Cancel
ModulesnumX
ModulenumY
PixelsperMdX
PixlesperMdY
CmdOK
CmdCancel

----------MainForm----------

```
Private Const AF_INET = 2
Private Const SOCK_STREAM = 1
Private Const SOCKET_ERROR = -1
Private Const INVALID_SOCKET = -1
Private Const BI_RGB = 0&
Private Const DIB_RGB_COLORS = 0&
Private Type IN_ADDR
  S_addr As Long
End Type
Private Type SOCK_ADDR
  sin_family As Integer
  sin_port As Integer
  sin_addr As IN_ADDR
  sin_zero(0 To 7) As Byte
End Type
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 GetDesktopWindow Lib "USER32"
  () As Long
Private Declare Function GetForegroundWindow Lib
  "USER32" () As Long
Private Declare Function GetDC Lib "USER32" (ByVal hWnd
  As Long) As Long
Private Declare Function GetWindowDC Lib "USER32"
  (ByVal hWnd As Long) As Long
Private Declare Function GetDIBits Lib "gdi32" (ByVal aHDC
  As Long, ByVal hBitmap As Long, ByVal nStartScan As
  Long, ByVal nNumScans As Long, lpBits As Any, lpBI
  As BITMAPINFO, ByVal wUsage As Long) As Long
Private Declare Function CreateDIBSection Lib "gdi32"
  (ByVal hdc As Long, pBitmapInfo As BITMAPINFO,
  ByVal un As Long, ByVal 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 DeleteDC Lib "gdi32" (ByVal hdc
  As Long) As Long
Private Declare Function DeleteObject Lib "gdi32" (ByVal
  hObject As Long) As Long
Private Declare Function CreateCompatibleDC Lib "gdi32"
  (ByVal hdc As Long) As Long
Private Declare Function socket Lib "wsock32" (ByVal af As
  Long, ByVal type_specification As Long, ByVal protocol
  As Long) As Long
Private Declare Function inet_addr Lib "wsock32" (ByVal cp
  As String) As Long
Private Declare Function connect Lib "wsock32" (ByVal s As
  Long, name As SOCK_ADDR, ByVal namelen As
  Integer) As Long
Private Declare Function send Lib "ws2_32" (ByVal s As
  Long, buffer As Any, ByVal length As Long, ByVal flags
  As Long) As Long
```

```
Private Declare Function htons Lib "wsock32" (ByVal
    hostshort As Long) As Integer
Private Declare Function closesocket Lib "wsock32.dll"
    (ByVal s As Long) As Long
'Private Declare Function sendto Lib "ws2_32" (ByVal s As
    Long, buffer As Any, ByVal length As Long, ByVal flags
    As Long, sTo As SOCK_ADDR, ByVal tolen As Long) As
    Long
Private Declare Function mciSendString Lib "winmm.dll"
    Alias "mciSendStringA" (ByVal lpstrCommand As String,
    ByVal lpstrReturnString As String, ByVal uReturnLength
    As Long, ByVal hwndCallback As Long) As Long
Private Declare Function mciGetErrorString Lib "winmm.dll"
    Alias "mciGetErrorStringA" (ByVal dwError As Long,
    ByVal lpstrBuffer As String, ByVal uLength As Long) As
    Long
Private Declare Function GetShortPathName Lib "kernel32"
    Alias "GetShortPathNameA" (ByVal lpszLongPath As
    String, ByVal lpszShortPath As String, ByVal cchBuffer
    As Long) As Long
Dim sockfd(), num, rate As Integer
Dim serv_addr() As SOCK_ADDR
Dim filetype As String
Dim hWndScreen As Long
Dim Strcommd As String * 255
Dim TotalFrames As String * 128
-------------------------------
Private Sub AWOption_Click()
   CmdClose_Click
   CmdOpen.Enabled = False
   swf.Stop
   framerate.Caption = 0
   Timer1.Enabled = True
End Sub
-------------------------------
Private Sub bConnect_Click()
   Dim Pixels As String
   Dim IP1, IP2 As String
   Dim length, value, X As Integer
   ReDim serv_addr(ModulesNumX.Caption *
       ModulesNumY.Caption)
   ReDim sockfd(ModulesNumX.Caption *
       ModulesNumY.Caption)
   IP1 = IPAddress.Caption
   length = InStrRev(IP1, ".")
   IP2 = Mid(IP1, length + 1, 3)
   IP1 = Mid(IP1, 1, length)
   num = ModulesNumX.Caption * ModulesNumY.Caption
   For value = 0 To num - 1
     With serv_addr(value)
       .sin_family = AF_INET
       .sin_addr.S_addr = inet_addr(IP1 + CStr(Val(IP2) +
         value))
       .sin_port = htons(4025)
     End With
     sockfd(value) = socket(AF_INET, SOCK_DGRAM, 0)
       If (sockfd(value) = INVALID_SOCKET) Then
         Statusprogram.Caption = "Can't open socket"
       End If
     Result = connect(sockfd(value), serv_addr(value),
         Len(serv_addr(value)))
     If (Result = SOCKET_ERROR) Then
       Statusprogram.Caption = "Can't connect to board"
       bConnect.Enabled = True
       Disconnect.Enabled = False
     Else
       bConnect.Enabled = False
       Disconnect.Enabled = True
     End If
   Next
End Sub
-------------------------------
Private Sub CmdClose_Click()
Dim Result As Long
   Result = mciSendString("Close " & "movie", 0&, 0&, 0&)
   CmdPlay.Enabled = False
   CmdPause.Enabled = False
   CmdResume.Enabled = False
   MovieFrame.Visible = False
End Sub
-------------------------------
```

```
Private Sub CmdPause_Click()
Dim Result As Long
   Result = mciSendString("Pause " & "movie", 0&, 0&, 0&)
   If Not Result = 0 Then  'not success
      MsgBox "Can't Pause", vbCritical: Exit Sub
   End If
   CmdResume.Enabled = True
   CmdPause.Enabled = False
End Sub
-----------------------------
Private Sub CmdPlay_Click()
Dim Result As Long
   framerate.Caption = 0
   Strcommd = "play " & "movie" & " from " & 0 & " to " &
       Val(TotalFrames)
   Result = mciSendString(Strcommd, 0&, 0&, 0&)
   If Not Result = 0 Then
      MsgBox "Can't Play filename", vbCritical: Exit Sub
   End If
   CmdPause.Enabled = True
   SwfStatus.Visible = False
   Timer1.Enabled = True
End Sub
-------------------------
Private Sub CmdResize_Click()
Dim Result As Long
Dim Left, Top, height, width As Long
   Left = 0
   Top = 0
   height = Picture1.ScaleHeight
   width = Picture1.ScaleWidth
   Result = mciSendString("put " & "movie" & " window at " &
       Left & " " & Top & " " & width & " " & height, 0&, 0&, 0&)
   If Not Result = 0 Then  'not success
      MsgBox "Can't Resize", vbCritical: Exit Sub
   End If
End Sub
-----------------------
Private Sub CmdResume_Click()
Dim Result As Long
   Result = mciSendString("Resume " & "movie", 0&, 0&, 0&)
   If Not Result = 0 Then  'not success
      MsgBox "Can't Resume", vbCritical: Exit Sub
   End If
   CmdResume.Enabled = False
   CmdPause.Enabled = True
End Sub
-------------------------
Private Sub Disconnect_Click()
   Dim value As Integer
   Timer1.Enabled = False
   For value = 0 To num - 1
      closesocket (sockfd(value))
   Next
   Disconnect.Enabled = False
End Sub
-------------------------
Private Sub DWOption_Click()
   CmdClose_Click
   CmdOpen.Enabled = False
   swf.Stop
   framerate.Caption = 0
   Timer1.Enabled = True
End Sub
-------------------------
Private Sub Form_Load()
   Timer1.Enabled = False
   Timer1.Tag = 0
   IPAddress.Caption = "161.200.85.122"
   PortNum.Caption = "4025"
   ModulesNumX.Caption = 1
   ModulesNumY.Caption = 2
   PixelsperMdX.Caption = 256
   PixelsperMdY.Caption = 64
   TotalPixelsX.Caption = ModulesNumX.Caption *
       PixelsperMdX.Caption
   TotalPixelsY.Caption = ModulesNumY.Caption *
       PixelsperMdY.Caption
   Image1.Top = 0
   Image1.Left = 0
   Image1.Visible = False
   hWndScreen = Picture1.hdc
```

```
    Disconnect.Enabled = False
    ReDim sockfd(ModulesNumX.Caption *
        ModulesNumY.Caption)
End Sub
---------------------------
Private Sub Form_QueryUnload(Cancel As Integer,
    UnloadMode As Integer)
Dim Result As Long
    Result = mciSendString("Close All", 0&, 0&, 0&)
    If Not Result = 0 Then
        MsgBox "Can't Close function", vbCritical: Exit Sub
    End If
End Sub
---------------------------
Private Sub CmdOpen_Click()
Dim Result As Long
Dim lenShort As Long
Dim ShortPathAndFile As String
Dim TotalTime As String * 128
Dim tmp As String * 255
Dim tmp2 As Integer
Const WS_CHILD = &H40000000
    CmdClose_Click
    swf.Stop
    framerate.Caption = 0
    With Dialog1
        .DialogTitle = "Open file"
        .ShowOpen
        FilenameLb.Caption = .FileName
        filetype = LCase(Right(.FileName, 3))
        Select Case filetype
            Case "jpg", "bmp", "tif", "gif"
                MovieFrame.Visible = False
                SwfStatus.Visible = False
                swf.Visible = False
                Image1.Visible = True
                Image1.Picture = LoadPicture(.FileName)
                Image1.Left = 0
                Image1.Top = 0
                Image1.width = Picture1.width
                Image1.height = Picture1.height
                Image1.Stretch = True
                Timer1.Enabled = True
            Case "swf"
                MovieFrame.Visible = False
                SwfStatus.Visible = True
                Image1.Visible = False
                swf.Visible = True
                swf.Movie = .FileName
                swf.Loop = True
                swf.Left = 0
                swf.Top = 0
                swf.width = Picture1.width
                swf.height = Picture1.height
                Timer1.Enabled = True
            Case "avi"
                SwfStatus.Visible = False
                swf.Visible = False
                Image1.Visible = False
                MovieFrame.Visible = True
                lenShort = GetShortPathName(Dialog1.FileName,
                    tmp, 255)
                ShortPathAndFile = Left$(tmp, lenShort)
                Strcommd = "open " & ShortPathAndFile & " type "
                    & "AviVideo" & " Alias " & "movie" & " parent " &
                    Picture1.hWnd & " Style " & WS_CHILD
                Result = mciSendString(Strcommd, 0&, 0&, 0&)
                If Not Result = 0 Then
                    MsgBox "Can't open filename", vbCritical: Exit
                        Sub
                End If
                CmdPlay.Enabled = True
                Result = mciSendString("set " & "movie" & " time
                    format ms", TotalTime, 128, 0&)
                Result = mciSendString("status " & "movie" & "
                    length", TotalTime, 128, 0&)
                LbTotalTime.Caption = CInt(TotalTime / 1000)
                Result = mciSendString("set " & "movie" & " time
                    format frames", TotalTime, 128, 0&)
                Result = mciSendString("status " & "movie" & "
                    length", TotalFrames, 128, 0&)
                LbTotalFrames.Caption = Val(TotalFrames)
```

```
            LbFramepersec.Caption =
                CInt(LbTotalFrames.Caption /
                LbTotalTime.Caption)
        End Select
    End With
End Sub
-----------------------------
Private Sub MFOption_Click()
    CmdOpen.Enabled = True
    hWndScreen = Picture1.hdc
End Sub
-----------------------------
Private Sub Setup_Click()
    SetupForm.Show
End Sub
-----------------------------
Private Sub Timer1_Timer()
Dim mBMPhdc As Long
Dim mBMPhBM As Long
Dim OldDC  As Long
Dim Result As Long
Dim mBMPbmpInfo As BITMAPINFO
Dim BMPInfo As BITMAPINFO
Dim Pixels() As Byte
Dim a, b, i, X, Y As Integer
Dim length As Integer
Dim FN As Integer
Dim z As String
    rate = rate + 1
    If framerate.Caption < rate Then
        framerate.Caption = rate
    End If
    Select Case filetype
        Case "swf"
        CFSwf.Caption = swf.CurrentFrame
        TFSwf.Caption = swf.TotalFrames
    End Select
    If DWOption Then
        hWndScreen = GetWindowDC(GetDesktopWindow())
    ElseIf AWOption Then
        hWndScreen = GetDC(GetForegroundWindow())
Else
    hWndScreen = Picture1.hdc
End If
With mBMPbmpInfo.bmiHeader
    .biBitCount = 24
    .biCompression = BI_RGB
    .biPlanes = 1
    .biSize = Len(mBMPbmpInfo.bmiHeader)
    .biWidth = TotalPixelsX.Caption
    .biHeight = TotalPixelsY.Caption
    .biSizeImage = .biWidth * .biHeight * 3
End With
mBMPhdc = CreateCompatibleDC(hWndScreen)
mBMPhBM = CreateDIBSection(mBMPhdc,
    mBMPbmpInfo, DIB_RGB_COLORS, ByVal &O0,
    ByVal &O0, ByVal &O0)
OldDC = SelectObject(mBMPhdc, mBMPhBM)
Call BitBlt(mBMPhdc, 0, 0, TotalPixelsX.Caption,
    TotalPixelsY.Caption, hWndScreen, 0, 0, vbSrcCopy)
ReDim Pixels(TotalPixelsX.Caption * TotalPixelsY.Caption
    * 3)
GetDIBits mBMPhdc, mBMPhBM, 0,
    TotalPixelsY.Caption, Pixels(0), mBMPbmpInfo,
    DIB_RGB_COLORS
SelectObject mBMPhdc, OldDC
DeleteObject mBMPhBM
DeleteDC mBMPhdc
length = PixelsperMdX.Caption * 3
For b = 1 To PixelsperMdY.Caption
    a = 0
    For Y = 1 To ModulesNumY
        i = 3 * TotalPixelsX.Caption * ((ModulesNumY - Y) *
            PixelsperMdY.Caption - b)
        For X = 1 To ModulesNumX
            Result = send(sockfd(a), Pixels(i), length, 0)
                '64*3bit
            If (Result = SOCKET_ERROR) Then
                Statusprogram.Caption = "Can't send data"
            End If
            i = i + length
```

```
            a = a + 1
          Next
        Next
      Next
End Sub
```

```
Private Sub Timer2_Timer()
   rate = 0
End Sub
```

---------- SetupForm ----------

```
Private Sub CmdCancel_Click()
   Unload Me
End Sub
```

```
Private Sub CmdOK_Click()
   If ModulesNumX.Text <> "" Then
      MainForm.ModulesNumX.Caption =
      ModulesNumX.Text
   If ModulesNumY.Text <> "" Then
      MainForm.ModulesNumY.Caption =
      ModulesNumY.Text
   If PixelsperMdX.Text <> "" Then
      MainForm.PixelsperMdX.Caption =
      PixelsperMdX.Text
   If PixelsperMdY.Text <> "" Then
      MainForm.PixelsperMdY.Caption =
      PixelsperMdY.Text
   If PortNum.Text <> "" Then MainForm.PortNum.Caption =
      PortNum.Text
   If IPAddress.Text <> "" Then
      MainForm.IPAddress.Caption = IPAddress.Text
   MainForm.TotalPixelsX.Caption =
      MainForm.ModulesNumX.Caption *
      MainForm.PixelsperMdX.Caption
   MainForm.TotalPixelsY.Caption =
      MainForm.ModulesNumY.Caption *
      MainForm.PixelsperMdY.Caption
   MainForm.Picture1.height =
      CInt(MainForm.TotalPixelsY.Caption)
   MainForm.Picture1.width =
      CInt(MainForm.TotalPixelsX.Caption)
   Unload Me
End Sub
```

ภาคผนวก ค.

รายละเอียดภาษา C บนส่วนรับข้อมูล

```
#include <stdio.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define SERV_TCP_PORT      4025
#define MAXLINE 96
#define GPIO_OUT2 0x3500d0
#define GPIO_OE2 0x3500b8
main(argc,argv)
int argc;
char *argv[];
{int sockfd,newsockfd,clilen,n;
 char line[MAXLINE];
  int x;
 struct sockaddr_in cli_addr,serv_addr;
 volatile unsigned long *pGPIOOut,*pGPIODir,*addr;
 long c,d,e;
 pGPIOOut = GPIO_OUT2;
 pGPIODir = GPIO_OE2;
 addr       = 0x00200000;
 *pGPIODir = 0x0000883E;
 *pGPIOOut = 0x883E0000;
//socket
 sockfd = socket(AF_INET,SOCK_STREAM,0);
 bzero((char*)&serv_addr,sizeof(serv_addr));
 serv_addr.sin_family = AF_INET;
 serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
 serv_addr.sin_port = htons(SERV_TCP_PORT);
//bind
 bind(sockfd,(struct sockaddr*)&serv_addr,
        sizeof(serv_addr));
//listen
 listen(sockfd,1);
 clilen = sizeof(cli_addr);
 //accept
newsockfd = accept(sockfd,(struct sockaddr*)&cli_addr, &clilen);
printf("accept succesc \n");
*pGPIOOut = 0x883E0004;
for(;;)
{for(y=0;y<16;y++)
   {memset(line,0,sizeof(line));
    n = read(newsockfd, line, MAXLINE);// [0/1/2
Blue/Green/Red/Blue]
    x = 0;
    *pGPIOOut = 0x883E0004;
    *pGPIOOut = 0x883E000C; // STA
    *addr = 0x00000000;
    *pGPIOOut = 0x883E001C;  //clk
    *pGPIOOut = 0x883E0004;
    for(i=0;i<32;i++)
      {addr = 0x00200000 +((line[x] &
0x000000FF)<<4);
       *addr = (line[x+1]<<24)+((line[x+2] &
0x000000FF) <<16);
       *pGPIOOut = 0x883E0014;
       *pGPIOOut = 0x883E0004;
        x = x + 3;
       }
 }
*pGPIOOut = 0x883E0000;
*pGPIOOut = 0x883E0002;
 *pGPIOOut = 0x883E0012;
  *pGPIOOut = 0x883E0000;
 }
}
/*
address  GPIO     Jamp port
15       31       J3pin1
11       27       J3pin2
5        21       J3pin6
4        20       J3pin7          CLK
3        19       J3pin8          STA
2        18       J3pin9          STB
1        17       J3pin10         Dast
0        16 (I2C_CLK)     J3pin11
*/
```

ภาคผนวก ง.

รายละเอียดภาษา VHDL ของ FPGA ภายในวงจรขับ

```
----------------------------------------------------------------
--                      Main Program
----------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity LED is
port        (CLK       : in  STD_logic;
             STA       : in  STD_logic;
             STB       : in  STD_logic;
             Dast      : in  STD_logic;
             CLK_inner : in STD_logic;
             Red_in    : in STD_logic_vector (7 downto 0);
             Green_in  : in STD_logic_vector (7 downto 0);
             Blue_in   : in STD_logic_vector (7 downto 0);
             STA_out   : out STD_logic;
             Shift     : out STD_logic_vector (15 downto 0);
             Latch : out STD_logic_vector (7 downto 0);
             PWM  : out STD_logic_vector (15 downto 0));
end LED;
architecture Behavioral of LED is
      component Red_part
      port (CLK_inner: in STD_logic;
            CLK        : in STD_logic;
            WA         : in STD_logic;
            Red_in  : in STD_logic_vector (7 downto 0);
            addrA1 : in STD_logic_vector (9 downto 0);
            addrA2 : in STD_logic_vector (9 downto 0);
            Red_out: out STD_logic_vector (9 downto 0));
      end component;
      component Green_part
      port   (CLK_inner : in STD_logic;
              CLK         : in STD_logic;
              WA          : in STD_logic;
              Green_in    : in STD_logic_vector (7 downto 0);
              addrA1      : in STD_logic_vector (9 downto 0);
              addrA2      : in STD_logic_vector (9 downto 0);
             Green_out : out STD_logic_vector (9 downto 0));
      end component;
      component Blue_part
      port  (CLK_inner : in STD_logic;
             CLK         : in STD_logic;
             WA          : in STD_logic;
             Blue_in     : in STD_logic_vector (7 downto 0);
             addrA1      : in STD_logic_vector (9 downto 0);
             addrA2      : in STD_logic_vector (9 downto 0);
            Blue_out    : out STD_logic_vector (9 downto 0));
      end component;
      component Controller
      port  (CLK        : in  STD_logic;
             CLK_inner: in STD_logic;
             STA        : in STD_logic;
             STB        : in STD_logic;
             DAST       : in STD_logic;
             WA         : out STD_logic;
             STA_out  : out  STD_logic;
             latch      : out STD_logic;
             addr1      : out STD_logic_vector (9 downto 0);
addr2      : out STD_logic_vector (9 downto 0);
             se_CLK   : out STD_logic_vector (15 downto 0);
             counter  : out STD_logic_vector (9 downto 0));
      end component;
      component Comparator
      port  (Red        : in STD_logic_vector(9 downto 0);
             Green      : in  STD_logic_vector(9 downto 0);
             Blue       : in STD_logic_vector(9 downto 0);
             counter    : in STD_logic_vector(9 downto 0);
             CLK_inner : in STD_logic;
             se_CLK     : in STD_logic;
             PWM        : out STD_logic;
             shift      : out STD_logic);
      end component;
      signal WA       : STD_logic;
      signal addr1 : STD_logic_vector (9 downto 0);
      signal addr2 : STD_logic_vector (9 downto 0);
      signal se_CLK : STD_logic_vector (15 downto 0);
      signal Red_out : STD_logic_vector (9 downto 0);
      signal Green_out : STD_logic_vector (9 downto 0);
      signal Blue_out: STD_logic_vector (9 downto 0);
      signal counter  : STD_logic_vector (9 downto 0);
```

```
    signal latch_1   : STD_logic;
    signal addr2_2 : STD_logic_vector(9 downto 0);
Begin
Red_part1     : Red_part        port map
    (CLK_inner,CLK,WA,Red_in,addr1,addr2,Red_out);
Green_part1  : Green_part      port map
    (CLK_inner,CLK,WA,Green_in,addr1,addr2,Green_out);
Blue_part1    : Blue_part       port map
    (CLK_inner,CLK,WA,Blue_in,addr1,addr2,Blue_out);
controller1     : controller      port map
    (CLK,CLK_inner,STA,STB,DAST,WA,STA_out,latch_1,addr1,
    addr2,se_CLK, counter);
Comparator1 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(0),PWM(0),shift(0));
Comparator2 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(1),PWM(1),shift(1));
Comparator3 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(2),PWM(2),shift(2));
Comparator4 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(3),PWM(3),shift(3));
Comparator5 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(4),PWM(4),shift(4));
Comparator6 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(5),PWM(5),shift(5));
Comparator7 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(6),PWM(6),shift(6));
Comparator8 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(7),PWM(7),shift(7));
Comparator9 : Comparator      port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(8),PWM(8),shift(8));
Comparator10 : Comparator     port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(9),PWM(9),shift(9));
Comparator11 : Comparator     port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(10),PWM(10),shift(10));
Comparator12 : Comparator     port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(11),PWM(11),shift(11));
Comparator13 : Comparator     port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(12),PWM(12),shift(12));
Comparator14 : Comparator     port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(13),PWM(13),shift(13));
Comparator15 : Comparator port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(14),PWM(14),shift(14));
Comparator16 : Comparator port map (Red_out,Green_out,
    Blue_out,counter,CLK_inner,se_CLK(15),PWM(15),shift(15));
Latch(0)  <= latch_1;
Latch(1)  <= latch_1;
Latch(2)  <= latch_1;
Latch(3)  <= latch_1;
Latch(4)  <= latch_1;
Latch(5)  <= latch_1;
Latch(6)  <= latch_1;
Latch(7)  <= latch_1;
end Behavioral;
------------------------------------------------------------
--                    Red part              --
------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Red_part is
port  (CLK_inner : in          STD_logic;
       CLK       : in  STD_logic;
       WA        : in  STD_logic;
       Red_in    : in  STD_logic_vector (7 downto 0);
       addrA1    : in  STD_logic_vector (9 downto 0);
       addrA2    : in  STD_logic_vector (9 downto 0);
       Red_out   : out STD_logic_vector (9 downto 0));
end Red_part;
architecture Behavioral of Red_part is
    component compensate_red
     port (addr      : in       STD_logic_vector(7 downto 0);
           data      : out      STD_logic_vector(9 downto 0));
    end component;
    component DualportRAM
    port  (CLK_inner : in       STD_logic;
           clk       : in       STD_logic;
           we        : in       STD_logic;
           addra     : in       STD_logic_vector(9 downto 0);
           addrb     : in       STD_logic_vector(9 downto 0);
           di        : in       STD_logic_vector(9 downto 0);
```

```
        do : out  STD_logic_vector(9 downto 0));
    end component;
    signal red1     : STD_logic_vector (9 downto 0);
Begin
compensate_red1  : compensate_red  port map
    (Red_in,red1);
DualportRAM1     : DualportRAM     port map
    (CLK_inner,CLK,WA,addrA1,addrA2,red1,Red_out);
End Behavioral;
------------------------------------------------
--                Green part                  --
------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Green_part is
port     (CLK_inner : in     STD_logic;
          CLK       : in     STD_logic;
          WA        : in     STD_logic;
          Green_in  : in     STD_logic_vector (7 downto 0);
          addrA1    : in     STD_logic_vector (9 downto 0);
          addrA2    : in     STD_logic_vector (9 downto 0);
          Green_out : out    STD_logic_vector (9 downto 0));
end Green_part;
architecture Behavioral of Green_part is
    component compensate_green
    port  (addr     : in     STD_logic_vector(7 downto 0);
           data     : out    STD_logic_vector(9 downto 0));
    end component;
    component DualportRAM
    port  (CLK_inner : in    STD_logic;
           clk       : in    STD_logic;
           we        : in    STD_logic;
           addra     : in    STD_logic_vector(9 downto 0);
           addrb     : in    STD_logic_vector(9 downto 0);
           di        : in    STD_logic_vector(9 downto 0);
           do        : out   STD_logic_vector(9 downto 0));
    end component;
    signal Green1  : STD_logic_vector (9 downto 0);
 Begin
compensate_green1 : compensate_green  port map
(Green_in,Green1);
DualportRAM1     : DualportRAM     port map
(CLK_inner,CLK,WA,addrA1,addrA2,Green1,Green_out);
End Behavioral;
------------------------------------------------
--                Blue part                   --
------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Blue_part is
port  (CLK_inner : in  STD_logic;
       CLK       : in  STD_logic;
       WA        : in  STD_logic;
       Blue_in   : in  STD_logic_vector (7 downto 0);
       addrA1    : in  STD_logic_vector (9 downto 0);
       addrA2    : in  STD_logic_vector (9 downto 0);
       Blue_out  : out STD_logic_vector (9 downto 0));
end Blue_part;
architecture Behavioral of Blue_part is
    component compensate_Blue
    port  (addr     : in     STD_logic_vector(7 downto 0);
           data     : out    STD_logic_vector(9 downto 0));
    end component;
    component DualportRAM
    port  (CLK_Inner : in    STD_logic;
           clk       : in    STD_logic;
           we        : in    STD_logic;
           addra     : in    STD_logic_vector(9 downto 0);
           addrb     : in    STD_logic_vector(9 downto 0);
           di        : in    STD_logic_vector(9 downto 0);
           do        : out   STD_logic_vector(9 downto 0));
    end component;
    signal Blue1   : STD_logic_vector (9 downto 0);
Begin
compensate_blue1:compensate_blue port map(Blue_in,Blue1);
DualportRAM1     : DualportRAM  port map
    (CLK_inner,CLK,WA,addrA1,addrA2,Blue1,Blue_out);
End Behavioral;
```

```
--------------------------------------------------------------------
--                    ROM (Compensate Red)                        --
--------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity compensate_red is
port        (addr     : in      std_logic_vector(7 downto 0);
             data     : out     std_logic_vector(9 downto 0));
end compensate_red;
architecture syn of compensate_red is
    type rom_type is array (255 downto 0) of std_logic_vector
(9 downto 0);
constant ROM : rom_type :=
("0011111111","0011111110","0011111101","0011111100","0011
111011","0011111010","0011111001","0011111000",
"0011110111","0011110110","0011110101","0011110100","00111
10011","0011110010","0011110001","0011110000",
"0011101111","0011101110","0011011101","0011101100","00111
01011","0011101010","0011101001","0011101000",
"0011100111","0011100110","0011010101","0011100100","00111
00011","0011100010","0011100001","0011100000",
"0011011111","0011011110","0011011101","0011011100","00110
11011","0011011010","0011011001","0011011000",
"0011010111","0011010110","0011010101","0011010100","00110
10011","0011010010","0011010001","0011010000",
"0011001111","0011001110","0011001101","0011001100","00110
01011","0011001010","0011001001","0011001000",
"0011000111","0011000110","0011000101","0011000100","00110
00011","0011000010","0011000001","0011000000",
"0010111111","0010111110","0010111101","0010111100","00101
11011","0010111010","0010111001","0010111000",
"0010110111","0010110110","0010110101","0010110100","00101
10011","0010110010","0010110001","0010110000",
"0010101111","0010101110","0010101101","0010101100","00101
01011","0010101010","0010101001","0010101000",
"0010100111","0010100110","0010100101","0010100100","00101
00011","0010100010","0010100001","0010100000",
"0010011111","0010011110","0010011101","0010011100","00100
11011","0010011010","0010011001","0010011000",
"0010010111","0010010110","0010010101","0010010100","00100
10011","0010010010","0010010001","0010010000",
"0010001111","0010001110","0010001101","0010001100","00100
01011","0010001010","0010001001","0010001000",
"0010000111","0010000110","0010000101","0010000100","00100
00011","0010000010","0010000001","0010000000",
"0001111111","0001111110","0001111101","0001111100","00011
11011","0001111010","0001111001","0001111000",
"0001110111","0001110110","0001110101","0001110100","00011
10011","0001110010","0001110001","0001110000",
"0001101111","0001101110","0001011101","0001101100","00011
01011","0001101010","0001101001","0001101000",
"0001100111","0001100110","0001010101","0001100100","00011
00011","0001100010","0001100001","0001100000",
"0001011111","0001011110","0001011101","0001011100","00010
11011","0001011010","0001011001","0001011000",
"0001010111","0001010110","0001010101","0001010100","00010
10011","0001010010","0001010001","0001010000",
"0001001111","0001001110","0001001101","0001001100","00010
01011","0001001010","0001001001","0001001000",
"0001000111","0001000110","0001000101","0001000100","00010
00011","0001000010","0001000001","0001000000",
"0000111111","0000111110","0000111101","0000111100","00001
11011","0000111010","0000111001","0000111000",
"0000110111","0000110110","0000110101","0000110100","00001
10011","0000110010","0000110001","0000110000",
"0000101111","0000101110","0000101101","0000101100","00001
01011","0000101010","0000101001","0000101000",
"0000100111","0000100110","0000100101","0000100100","00001
00011","0000100010","0000100001","0000100000",
"0000011111","0000011110","0000011101","0000011100","00000
11011","0000011010","0000011001","0000011000",
"0000010111","0000010110","0000010101","0000010100","00000
10011","0000010010","0000010001","0000010000",
"0000001111","0000001110","0000001101","0000001100","00000
01011","0000001010","0000001001","0000001000",
"0000000111","0000000110","0000000101","0000000100","00000
00011","0000000010","0000000001","0000000000");

 Begin
 data <= ROM(conv_integer(addr));
 End syn;
```

```
----------------------------------------------------------------
--                    ROM (Compensate Green)                  --
----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity compensate_Green is
port        (addr     : in      std_logic_vector(7 downto 0);
             data     : out     std_logic_vector(9 downto 0));
end compensate_Green;
architecture syn of compensate_Green is
    type rom_type is array (255 downto 0) of std_logic_vector
(9 downto 0);
 constant ROM : rom_type :=
("0011111111","0011111110","0011111101","0011111100","0011
111011","0011111010","0011111001","0011111000",
"0011110111","0011110110","0011110101","0011110100","00111
10011","0011110010","0011110001","0011110000",
"0011101111","0011101110","0011011101","0011101100","00111
01011","0011101010","0011101001","0011101000",
"0011100111","0011100110","0011010101","0011100100","00111
00011","0011100010","0011100001","0011100000",
"0011011111","0011011110","0011011101","0011011100","00110
11011","0011011010","0011011001","0011011000",
"0011010111","0011010110","0011010101","0011010100","00110
10011","0011010010","0011010001","0011010000",
"0011001111","0011001110","0011001101","0011001100","00110
01011","0011001010","0011001001","0011001000",
"0011000111","0011000110","0011000101","0011000100","00110
00011","0011000010","0011000001","0011000000",
"0010111111","0010111110","0010111101","0010111100","00101
11011","0010111010","0010111001","0010111000",
"0010110111","0010110110","0010110101","0010110100","00101
10011","0010110010","0010110001","0010110000",
"0010101111","0010101110","0010101101","0010101100","00101
01011","0010101010","0010101001","0010101000",
"0010100111","0010100110","0010100101","0010100100","00101
00011","0010100010","0010100001","0010100000",
"0010011111","0010011110","0010011101","0010011100","00100
11011","0010011010","0010011001","0010011000",
"0010010111","0010010110","0010010101","0010010100","00100
10011","0010010010","0010010001","0010010000",
"0010001111","0010001110","0010001101","0010001100","00100
01011","0010001010","0010001001","0010001000",
"0010000111","0010000110","0010000101","0010000100","00100
00011","0010000010","0010000001","0010000000",
"0001111111","0001111110","0001111101","0001111100","00011
11011","0001111010","0001111001","0001111000",
"0001110111","0001110110","0001110101","0001110100","00011
10011","0001110010","0001110001","0001110000",
"0001101111","0001101110","0001011101","0001101100","00011
01011","0001101010","0001101001","0001101000",
"0001100111","0001100110","0001010101","0001100100","00011
00011","0001100010","0001100001","0001100000",
"0001011111","0001011110","0001011101","0001011100","00010
11011","0001011010","0001011001","0001011000",
"0001010111","0001010110","0001010101","0001010100","00010
10011","0001010010","0001010001","0001010000",
"0001001111","0001001110","0001001101","0001001100","00010
01011","0001001010","0001001001","0001001000",
"0001000111","0001000110","0001000101","0001000100","00010
00011","0001000010","0001000001","0001000000",
"0000111111","0000111110","0000111101","0000111100","00001
11011","0000111010","0000111001","0000111000",
"0000110111","0000110110","0000110101","0000110100","00001
10011","0000110010","0000110001","0000110000",
"0000101111","0000101110","0000101101","0000101100","00001
01011","0000101010","0000101001","0000101000",
"0000100111","0000100110","0000100101","0000100100","00001
00011","0000100010","0000100001","0000100000",
"0000011111","0000011110","0000011101","0000011100","00000
11011","0000011010","0000011001","0000011000",
"0000010111","0000010110","0000010101","0000010100","00000
10011","0000010010","0000010001","0000010000",
"0000001111","0000001110","0000001101","0000001100","00000
01011","0000001010","0000001001","0000001000",
"0000000111","0000000110","0000000101","0000000100","00000
00011","0000000010","0000000001","0000000000");

 Begin
 data <= ROM(conv_integer(addr));
 End syn;
```

```
----------------------------------------------------------
--                    ROM (Compensate Blue)              --
----------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity compensate_Blue is
port      (addr     : in      std_logic_vector(7 downto 0);
           data     : out     std_logic_vector(9 downto 0));
end compensate_Blue;
architecture syn of compensate_Blue is
    type rom_type is array (255 downto 0) of std_logic_vector
(9 downto 0);
 constant ROM : rom_type :=
("0011111111","0011111110","0011111101","0011111100","0011
111011","0011111010","0011111001","0011111000",
"0011110111","0011110110","0011110101","0011110100","00111
10011","0011110010","0011110001","0011110000",
"0011101111","0011101110","0011011101","0011101100","00111
01011","0011101010","0011101001","0011101000",
"0011100111","0011100110","0011010101","0011100100","00111
00011","0011100010","0011100001","0011100000",
"0011011111","0011011110","0011011101","0011011100","00110
11011","0011011010","0011011001","0011011000",
"0011010111","0011010110","0011010101","0011010100","00110
10011","0011010010","0011010001","0011010000",
"0011001111","0011001110","0011001101","0011001100","00110
01011","0011001010","0011001001","0011001000",
"0011000111","0011000110","0011000101","0011000100","00110
00011","0011000010","0011000001","0011000000",
"0010111111","0010111110","0010111101","0010111100","00101
11011","0010111010","0010111001","0010111000",
"0010110111","0010110110","0010110101","0010110100","00101
10011","0010110010","0010110001","0010110000",
"0010101111","0010101110","0010101101","0010101100","00101
01011","0010101010","0010101001","0010101000",
"0010100111","0010100110","0010100101","0010100100","00101
00011","0010100010","0010100001","0010100000",
"0010011111","0010011110","0010011101","0010011100","00100
11011","0010011010","0010011001","0010011000",
"0010010111","0010010110","0010010101","0010010100","00100
10011","0010010010","0010010001","0010010000",
"0010001111","0010001110","0010001101","0010001100","00100
01011","0010001010","0010001001","0010001000",
"0010000111","0010000110","0010000101","0010000100","00100
00011","0010000010","0010000001","0010000000",
"0001111111","0001111110","0001111101","0001111100","00011
11011","0001111010","0001111001","0001111000",
"0001110111","0001110110","0001110101","0001110100","00011
10011","0001110010","0001110001","0001110000",
"0001101111","0001101110","0001011101","0001101100","00011
01011","0001101010","0001101001","0001101000",
"0001100111","0001100110","0001010101","0001100100","00011
00011","0001100010","0001100001","0001100000",
"0001011111","0001011110","0001011101","0001011100","00010
11011","0001011010","0001011001","0001011000",
"0001010111","0001010110","0001010101","0001010100","00010
10011","0001010010","0001010001","0001010000",
"0001001111","0001001110","0001001101","0001001100","00010
01011","0001001010","0001001001","0001001000",
"0001000111","0001000110","0001000101","0001000100","00010
00011","0001000010","0001000001","0001000000",
"0000111111","0000111110","0000111101","0000111100","00001
11011","0000111010","0000111001","0000111000",
"0000110111","0000110110","0000110101","0000110100","00001
10011","0000110010","0000110001","0000110000",
"0000101111","0000101110","0000101101","0000101100","00001
01011","0000101010","0000101001","0000101000",
"0000100111","0000100110","0000100101","0000100100","00001
00011","0000100010","0000100001","0000100000",
"0000011111","0000011110","0000011101","0000011100","00000
11011","0000011010","0000011001","0000011000",
"0000010111","0000010110","0000010101","0000010100","00000
10011","0000010010","0000010001","0000010000",
"0000001111","0000001110","0000001101","0000001100","00000
01011","0000001010","0000001001","0000001000",
"0000000111","0000000110","0000000101","0000000100","00000
00011","0000000010","0000000001","0000000000");

 Begin
 data <= ROM(conv_integer(addr));
 End syn;
```

ต้นฉบับ หน้าขาดหาย

```
            y <= "100";
        elsif y = "100" then
            shift <= '1';
            y <= "101";
        elsif y = "101" then
            PWM <= pwm_b;
            shift <= '0';
            y <= "110";
        elsif y = "110" then
            shift <= '1';
            y <= "111";
        else
            shift <= '0';
        end if;
    end if;
    end process;
End Behavioral;
------------------------------------------------------------
--                    Controller                          --
------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Controller is
port    (CLK        : in    STD_logic;
         CLK_Inner  : in    STD_logic;
         STA        : in    STD_logic;
         STB        : in    STD_logic;
         DAST       : in    STD_logic;
         WA         : out   STD_logic;
         STA_out    : out   STD_logic;
         latch      : out   STD_logic;
         addr1      : out   STD_logic_vector (9 downto 0);
         addr2      : out   STD_logic_vector (9 downto 0);
         se_CLK     : out   STD_logic_vector (15 downto 0);
         counter    : out   STD_logic_vector (9 downto 0));
end Controller;
architecture Behavioral of Controller is
    component clear
    port  (DAST         : in    STD_logic;
           CLK_inner    : in    STD_logic;
           CLR          : out   STD_logic);
    end component;
    component  addr_RAM
    port  (CLK          : in    STD_logic;
           CLK_Inner    : in    STD_logic;
           STA          : in    STD_logic;
           STB          : in    STD_logic;
           DAST         : in    STD_logic;
           CLR          : in    STD_logic;
           STA_out      : out   STD_logic;
           WA           : out   STD_logic;
           addr1        : out   STD_logic_vector (9 downto 0);
           addr2        : out   STD_logic_vector (9 downto 0));
    end component;
    component selector
    port  (CLK_inner    : in    STD_logic;
           CLR          : in    STD_logic;
           latch        : out   STD_logic;
           se_CLK       : out   STD_logic_vector (15 downto 0);
           counter      : out   STD_logic_vector (9 downto 0));
    end component;
    signal CLR : STD_logic;
Begin
clear1     : clear  port map     (DAST,CLK_inner,CLR);
addr_RAM1 : addr_RAM  port map        (CLK,CLK_inner,
    STA,STB,DAST,CLR,STA_out,WA,addr1,addr2);
selector1  : selector   port map
    (CLK_inner,CLR,latch,se_CLK,counter);
End Behavioral;
------------------------------------------------------------
--                    addr_RAM                            --
------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity addr_RAM is
port    (CLK        : in    STD_logic;
         CLK_inner  : in    STD_logic;
         STA        : in    STD_logic;
```

```
        STB         : in    STD_logic;
        DAST        : in    STD_logic;
        CLR         : in    STD_logic;
        STA_out     : out   STD_logic;
        WA          : out   STD_logic;
        addr1       : out   STD_logic_vector (9 downto 0);
        addr2       : out   STD_logic_vector (9 downto 0));
end addr_RAM;
architecture Behavioral of addr_RAM is
    component timing
    port  (CLK      : in    STD_logic;
        STA         : in    STD_logic;
        STB         : in    STD_logic;
        STA_out     : in    STD_logic;
        Dast        : in    STD_logic;
        WA          : out   STD_logic);
    end component;
    component addr_1
    port  (CLK      : in    STD_logic;
        DAST        : in    STD_logic;
        STA         : in    STD_logic;
        STB         : in    STD_logic;
        selector    : in    STD_logic;
        addr1       : out   STD_logic_vector (9 downto 0);
        STA_out     : out   STD_logic);
    end component;
    component addr_2
    port  (CLK_inner : in   STD_logic;
        CLR         : in    STD_logic;
        addr2       : out   STD_logic_vector (9 downto 0);
        selector    : out   STD_logic);
    end component;
    signal stop     : STD_logic;
    signal z        : STD_logic;
    signal WA1      : STD_logic;
Begin
timing1   : timing   port map (CLK,STA,STB,stop,Dast,WA1);
addr_11   : addr_1   port map (CLK,DAST,STA
   ,STB,z,addr1,stop);
addr_21   : addr_2   port map (CLK_inner,CLR,addr2,z);
STA_out <= stop;
WA      <= WA1;
End Behavioral;
-------------------------------------------------------------
-- CLR signal (use for clear data on part that using CLK_inner) -
-------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity Clear is
port    (DAST           : in    STD_logic;
         CLK_inner      : in    STD_logic;
         CLR            : out   STD_logic);
end Clear;
architecture Behavioral of Clear is
    signal I            : STD_logic;
Begin
    process (Dast)
    begin
        if CLK_Inner'event and CLK_inner = '1' then
            if I = '0' then
                if DAST   = '1' then
                    CLR <= '1';
                    I <= '1';
                end if;
            elsif DAST = '1' then
                CLR <= '0';
                I <= '1';
            else
                CLR <= '0';
                I <= '0';
            end if;
        end if;
    end process;
End Behavioral;
-------------------------------------------------------------
--                      timing                             --
-------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity timing is
port      (CLK      : in      STD_logic;
           STA      : in      STD_logic;
           STB      : in      STD_logic;
           STA_out  : in      STD_logic;
           DAST     : in      STD_logic;
           WA       : out     STD_logic);
end timing;
architecture Behavioral of timing is
Begin
    process (STA,STA_out,DAST)
    begin
        if CLK'event and CLK = '1' then
            if DAST = '1' then
                wa <= '0';
            elsif STB = '1' then
                if STA = '1' then
                    wa <= '1';
                elsif STA_out = '1' then
                    wa <= '0';
                end if;
            else
                wa <= '0';
            end if;
        end if;
    end process;
End Behavioral;
```

```
--------------------------------------------------
-                    addr_1                      -
--------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity addr_1 is
port      (CLK      : in      STD_logic;
           DAST     : in      STD_logic;
           STA      : in      STD_logic;
           STB      : in      STD_logic;
           selector : in      STD_logic;
           addr1    : out     STD_logic_vector (9 downto 0);
           STA_out  : out     STD_logic);
end addr_1;
architecture Behavioral of addr_1 is
    signal buff : STD_logic_vector (4 downto 0);
        -- used to identify pixel's position
    signal buff2: STD_logic_vector (8 downto 0);
        -- address for Address1
Begin
    process (CLK,DAST,STB)
    begin
        if CLK'event and CLK = '1' then
            if DAST = '1' then
                buff <= "00000";
                buff2 <= "111111111";
                STA_out <= '0';
            elsif STB = '1' then
                if STA = '1' then
                    buff <= "00000";
                    buff2 <= buff2 + 1;
                elsif buff = "11110" then
                    STA_out <= '1';
                    buff <= buff + 1;
                    buff2 <= buff2 + 1;
                elsif buff = "11111" then
                    STA_out <= '0';
                else
                    buff <= buff + 1;
                    buff2 <= buff2 + 1;
                end if;
            else
                STA+out <= '0';
            end if;
        end if;
    end process;
    addr1(0) <= buff2(0);
    addr1(1) <= buff2(1);
    addr1(2) <= buff2(2);
    addr1(3) <= buff2(3);
    addr1(4) <= buff2(4);
    addr1(5) <= buff2(5);
```

```
        addr1(6) <= buff2(6);
        addr1(7) <= buff2(7);
        addr1(8) <= buff2(8);
        addr1(9) <= selector;
End Behavioral;
------------------------------------------------------------
--                      addr_2                            --
------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity addr_2 is
port    (CLK_inner  : in    STD_logic;
         CLR        : in    STD_logic;
         addr2      : out   STD_logic_vector (9 downto 0);
         selector   : out   STD_logic);
end addr_2;
architecture Behavioral of addr_2 is
    signal z                : STD_logic;
    signal buff3    : STD_logic_vector (8 downto 0);
    -- address for Reading data use in PWM part
    signal buff9    : STD_logic_vector(2 downto 0);
    -- waiting time for next loop (same as buff8 in selector)
Begin
    process (CLK_inner,CLR,z)
    begin
        if CLK_inner'event and CLK_inner = '1' then
            if CLR = '1' then
                if z = '0' then
                    z <= '1';
                else
                    z <= '0';
                end if;
                buff3 <= "000000000";
                buff9 <= "000";
            elsif buff3 < 480 then
                buff3 <= buff3 + 32 ;
            elsif buff3 < 511 then
                buff3 <= buff3 + 33;
            elsif buff9 = "111" then
                buff9 <= "000";

                buff3 <= "000000000";
            else
                buff9 <= buff9 + 1;
            end if;
        end if;
    end process;
    addr2(0) <= buff3(0);
    addr2(1) <= buff3(1);
    addr2(2) <= buff3(2);
    addr2(3) <= buff3(3);
    addr2(4) <= buff3(4);
    addr2(5) <= buff3(5);
    addr2(6) <= buff3(6);
    addr2(7) <= buff3(7);
    addr2(8) <= buff3(8);
    addr2(9) <= not z;
    selector <= z;
end Behavioral;
------------------------------------------------------------
-- PWM Clock  (use for shift data to each comparator and gen
-- shift&Latch for --  MBI5026 )
------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity selector is
port    (CLK_inner : in     STD_logic;
         CLR        : in    STD_logic;
         latch      : out   STD_logic;
         se_CLK     : out   STD_logic_vector (15 downto 0);
         counter    : out   STD_logic_vector (9 downto 0));
end selector;
architecture Behavioral of selector is
    signal buff5 : STD_logic_vector(15 downto 0);
        -- select comparator
    signal buff6 : STD_logic_vector(4 downto 0);
        -- waiting time for counter
    signal buff7 : STD_logic_vector(9 downto 0);
```

```
        -- counter
    signal buff8 : STD_logic_vector(2 downto 0);
        -- wait untill last pixel storage data
    signal d : STD_logic;
Begin
    process (CLK_inner)
    begin
        if CLK_inner'event and CLK_inner = '1' then
            if CLR = '1' then
                buff5 <= "0000000000000000";
                buff6 <= "00000";
                buff7 <= "0000000000";
                buff8 <= "000";
                latch <= '0';
                d <= '1';
            elsif d = '1' then
                d <= '0';
                buff5 = "0000000000000001";
            elsif buff5 = "0000000000000001" then
                latch <= '0';
                buff5 <= "0000000000000010";
            elsif buff5 = "0000000000000010" then
                buff5 <= "0000000000000100";
            elsif buff5 = "0000000000000100" then
                buff5 <= "0000000000001000";
            elsif buff5 = "0000000000001000" then
                buff5 <= "0000000000010000";
            elsif buff5 = "0000000000010000" then
                buff5 <= "0000000000100000";
            elsif buff5 = "0000000000100000" then
                buff5 <= "0000000001000000";
            elsif buff5 = "0000000001000000" then
                buff5 <= "0000000010000000";
            elsif buff5 = "0000000010000000" then
                buff5 <= "0000000100000000";
            elsif buff5 = "0000000100000000" then
                buff5 <= "0000001000000000";
            elsif buff5 = "0000001000000000" then
                buff5 <= "0000010000000000";
            elsif buff5 = "0000010000000000" then
                buff5 <= "0000100000000000";
            elsif buff5 = "0000100000000000" then
                buff5 <= "0001000000000000";
            elsif buff5 = "0001000000000000" then
                buff5 <= "0010000000000000";
            elsif buff5 = "0010000000000000" then
                buff5 <= "0100000000000000";
            elsif buff5 = "0100000000000000" then
                buff5 <= "1000000000000000";
            elsif buff6 = "11111" then
                if buff8 = "111" then
                    latch <= '1';
                    buff5 <= "0000000000000001";
                    buff7 <= buff7 + "0100010011";
                    buff6 <= "00000";
                    buff8 <= "000";
                else
                    buff8 <= buff8 + 1;
                    buff5 <= "0000000000000000";
                end if;
            else
                buff6 <= buff6 + 1;
                buff5 <= "0000000000000001";
            end if;
        end if;
    end process;
    se_CLK <= buff5;
    counter <= buff7;
End Behavioral;
```

ประวัติผู้เขียนวิทยานิพนธ์

นายวุฒิไกร รัตนวุฒิสุวรรณ เกิดวันที่ 1 กุมภาพันธ์ พ.ศ. 2524 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า จากจุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2544 และได้เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า (ห้องปฏิบัติการวัดคุมอุตสาหกรรม) ที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2545