Interactive Gantt Chart Scheduler in Access

Aleksander Wojtasz

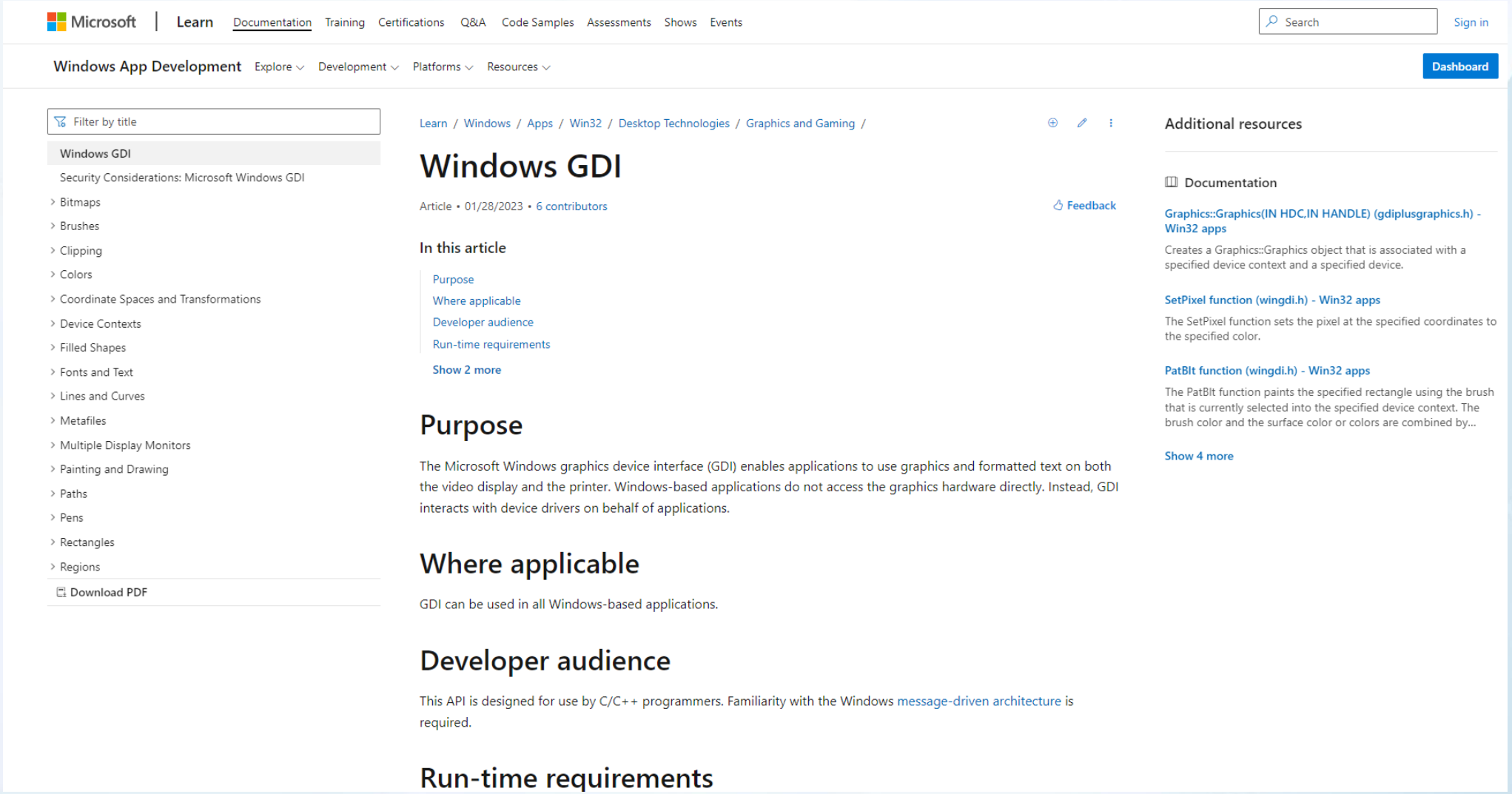
Access Europe User Group Meeting
Wed 5 July 2023

wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

What is GDI?

https://learn.microsoft.com/en-us/windows/win32/gdi/windows-gdi



wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

GDI library file in Microsoft Windows

https://en.wikipedia.org/wiki/Microsoft_Windows_library_files#GDI32.DLL

GDI32.DLL [edit]

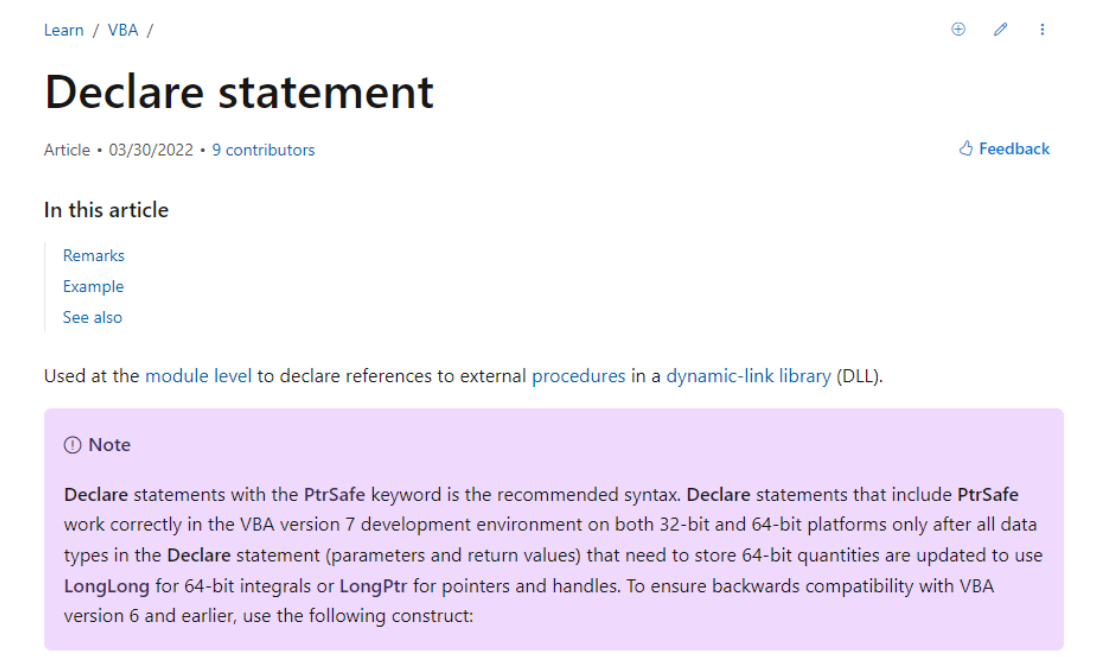
GDI32.DLL exports Graphics Device Interface (GDI) functions that perform primitive drawing functions for output to video displays and printers. It is used, for example, in the XP version of Paint. Applications call GDI functions directly to perform low-level drawing (line, rectangle, ellipse), text output, font management, and similar functions.[8][9]

Initially, GDI supported 16 and 256 color EGA/VGA display cards and monochrome printers. The functionality has expanded over the years, and now includes support for things like TrueType fonts, alpha channels, and multiple monitors.[10]

CHAPTER 1 - Theory

How can we call gdi functions from VBA code

https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/declare-statement



Learn / VBA /

Declare statement

Article • 03/30/2022 • 9 contributors

Feedback

In this article

- Remarks
- Example
- See also

Used at the module level to declare references to external procedures in a dynamic-link library (DLL).

Note

Declare statements with the **PtrSafe** keyword is the recommended syntax. **Declare** statements that include **PtrSafe** work correctly in the VBA version 7 development environment on both 32-bit and 64-bit platforms only after all data types in the **Declare** statement (parameters and return values) that need to store 64-bit quantities are updated to use **LongLong** for 64-bit integrals or **LongPtr** for pointers and handles. To ensure backwards compatibility with VBA version 6 and earlier, use the following construct:

Declare Statement – in the vba code

```
Private Declare PtrSafe Function apiMoveToEx Lib "gdi32" Alias "MoveToEx" _
(ByVal hdc As LongPtr, ByVal x As Long, ByVal y As Long, lpPoint As Any) As Long
'above was lpPoint as POINTAPI, changed to Any to allow NULL

Private Declare PtrSafe Function LineTo Lib "gdi32" _
(ByVal hdc As LongPtr, ByVal x As Long, ByVal y As Long) As Long

Private Declare PtrSafe Function apiCreateFontIndirect Lib "gdi32" Alias _
        "CreateFontIndirectA" (lpLogFont As LOGFONT) As LongPtr

Private Declare PtrSafe Function apiSetTextColor Lib "gdi32" Alias "SetTextColor" _
(ByVal hdc As LongPtr, ByVal crColor As Long) As Long

Private Declare PtrSafe Function apiSetBkColor Lib "gdi32" Alias "SetBkColor" _
(ByVal hdc As LongPtr, ByVal crColor As Long) As Long

Private Declare PtrSafe Function SetBkMode Lib "gdi32" _
(ByVal hdc As LongPtr, ByVal nBkMode As Long) As Long

Private Declare PtrSafe Function apiDrawText Lib "user32" Alias "DrawTextA" _
    (ByVal hdc As LongPtr, ByVal lpStr As String, _
    ByVal nCount As Long, lpRect As RECT, ByVal wFormat As Long) As Long

Private Declare PtrSafe Function apiCreateSolidBrush Lib "gdi32" _
Alias "CreateSolidBrush" _
(ByVal crColor As Long) As LongPtr

Private Declare PtrSafe Function Rectangle Lib "gdi32" _
(ByVal hdc As LongPtr, ByVal X1 As Long, ByVal Y1 As Long, _
ByVal X2 As Long, ByVal Y2 As Long) As Long

Private Declare PtrSafe Function GetEnhMetaFileHeader Lib "gdi32" _
(ByVal hemf As LongPtr, ByVal cbBuffer As Long, lpemh As ENHMETAHEADER) As Long

Private Declare PtrSafe Function PlayEnhMetaFile Lib "gdi32" _
(ByVal hdc As LongPtr, ByVal hemf As LongPtr, lpRect As RECT) As Long
```

wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

In GDI32 library often input parameters are not simple datatypes (they are data structures)
So we need declare data structures that are in accordance with original gdi functions parameters
And use them as input parameters:

```
Private Declare PtrSafe Function CreateDIBSection Lib "gdi32" _
(ByVal hdc As LongPtr, pBitmapInfo As BITMAPINFO, _
ByVal un As Long, ByRef lplpVoid As LongPtr, ByVal handle As LongPtr, _
ByVal dw As Long) As LongPtr
```

```
Private Type BITMAPINFO
  bmiHeader As BITMAPINFOHEADER
  bmiColors As RGBQUAD
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 'ERGBCompression
  biSizeImage As Long
  biXPelsPerMeter As Long
  biYPelsPerMeter As Long
  biClrUsed As Long
  biClrImportant As Long
End Type
```

```
Private Type RGBQUAD
  rgbBlue As Byte
  rgbGreen As Byte
  rgbRed As Byte
  rgblReterved As Byte
End Type
```

Original function call

Syntax

C++ Copy

```
HBITMAP CreateDIBSection(
  [in]  HDC              hdc,
  [in]  const BITMAPINFO *pbmi,
  [in]  UINT             usage,
  [out] VOID             **ppvBits,
  [in]  HANDLE           hSection,
  [in]  DWORD            offset
);
```

C++

```
typedef struct tagBITMAPINFO {
  BITMAPINFOHEADER bmiHeader;
  RGBQUAD          bmiColors[1];
} BITMAPINFO, *LPBITMAPINFO, *PBITMAPINFO;
```

C++

```
typedef struct tagBITMAPINFOHEADER {
  DWORD biSize;
  LONG  biWidth;
  LONG  biHeight;
  WORD  biPlanes;
  WORD  biBitCount;
  DWORD biCompression;
  DWORD biSizeImage;
  LONG  biXPelsPerMeter;
  LONG  biYPelsPerMeter;
  DWORD biClrUsed;
  DWORD biClrImportant;
} BITMAPINFOHEADER, *LPBITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

Two central objects of GDI32 library: Memory Device Context & DIB (Device-Independent Bitmaps)

1. Memory Device Contexts

https://learn.microsoft.com/en-us/windows/win32/gdi/memory-device-contexts

Memory Device Contexts

Article • 01/07/2021 • 3 contributors

Feedback

To enable applications to place output in memory rather than sending it to an actual device, use a special device context for bitmap operations called a *memory device context*. A memory DC enables the system to treat a portion of memory as a virtual device. It is an array of bits in memory that an application can use temporarily to store the color data for bitmaps created on a normal drawing surface. Because the bitmap is compatible with the device, a memory DC is also sometimes referred to as a *compatible device context*.

The memory DC stores bitmap images for a particular device. An application can create a memory DC by calling the CreateCompatibleDC function.

The **CreateCompatibleDC** function creates a memory device context (DC) compatible with the specified device.

Creating a DC in VBA code :

```
' Create a DC compatible with the current display
m_hDC = CreateCompatibleDC(0&)
```

wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

2. DIB (Device-Independent Bitmaps)

https://learn.microsoft.com/en-us/windows/win32/gdi/device-independent-bitmaps

Device-Independent Bitmaps

Article • 11/19/2022 • 6 contributors

Feedback

A device-independent bitmap (DIB) contains a *color table*. A color table describes how pixel values correspond to *RGB* color values, which describe colors that are produced by emitting light. Thus, a DIB can achieve the proper color scheme on any device. A DIB contains the following color and dimension information:

- The color format of the device on which the rectangular image was created.
- The resolution of the device on which the rectangular image was created.
- The palette for the device on which the image was created.
- An array of bits that maps red, green, blue (RGB) triplets to pixels in the rectangular image.
- A data-compression identifier that indicates the data compression scheme (if any) used to reduce the size of the array of bits.

The color and dimension information is stored in a BITMAPINFO structure, which consists of a BITMAPINFOHEADER structure followed by two or more RGBQUAD structures. The **BITMAPINFOHEADER** structure specifies the dimensions of the pixel rectangle, describes the device's color technology, and identifies the compression schemes used to reduce the size of the bitmap. The **RGBQUAD** structures identify the colors that appear in the pixel rectangle.

Creating a DIB in VBA code :

```
' Create our DIBSection
hDib = CreateDIBSection(m_hDC, m_bmi, DIB_RGB_COLORS, m_lPtr, 0, 0)
```

https://blackice.com/Help/Tools/Document%20Imaging%20SDK%20webhelp/WebHelp/What_is_DIB.htm

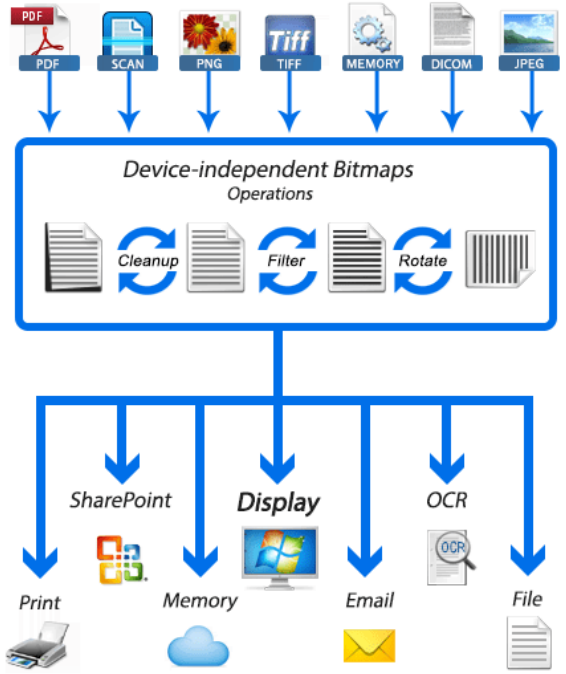
What is DIB?

DIBs, or device independent bitmaps, were created by Microsoft with the goal of creating an image format which can be displayed regardless of the display device used, thus the device independent portion of its name. DIBs were created for use in the first Microsoft Windows operating systems, but are still widely used today. Additional details on the DIB file format structure and history are available from the link below:

http://msdn.microsoft.com/en-us/library/windows/desktop/dd183562(v=vs.85).aspx
(The link cannot be opened in the help system. Please copy and paste into a browser)

If the above link does not work, check the Microsoft Developers Network (MSDN) for DIBs.

Black Ice Software's imaging toolkits utilize the DIB as the focal point for all of the toolkits. Any supported image file format can be loaded from a file into a DIB, which can then be displayed, manipulated, printed or saved into a different file format.



wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

Attaching Memory Device Context to DIB

https://learn.microsoft.com/en-us/windows/win32/gdi/memory-device-contexts

The original bitmap in a memory DC is simply a placeholder. Its dimensions are one pixel by one pixel. Before an application can begin drawing, it must select a bitmap with the appropriate width and height into the DC by calling the SelectObject function.

Attaching in VBA code :

```
m_hBmpOld = SelectObject(m_hDC, m_hDIb)
```

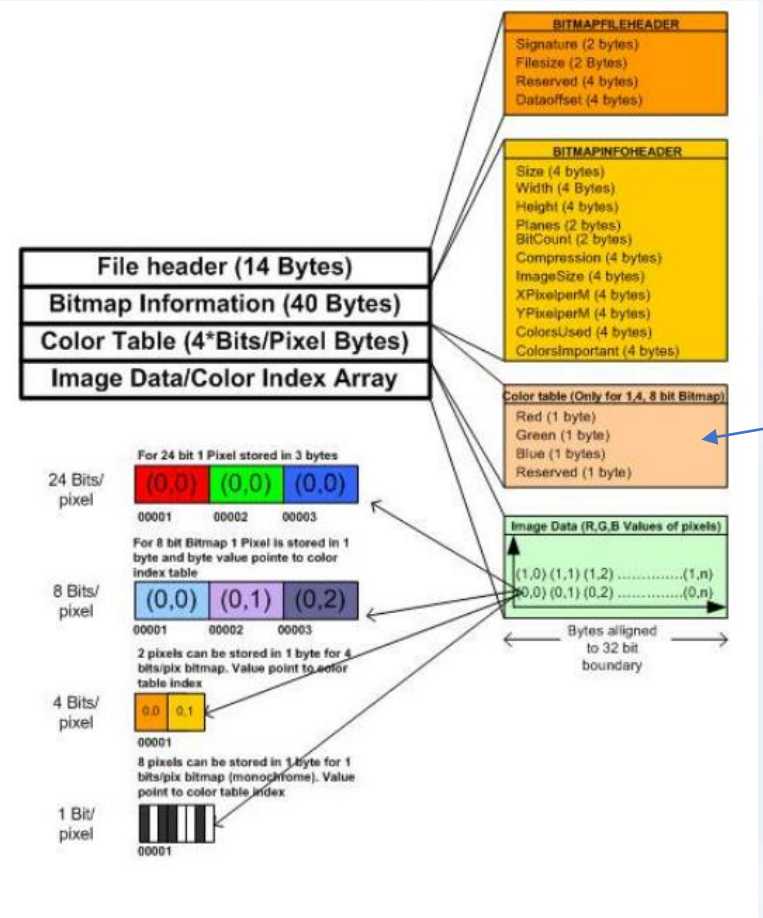
We can now perform all the available drawing operations with the use of Memory Device Context attached to our DIB

CHAPTER 1 - Theory

Transfering DIB data into Microft Access Image Control Data

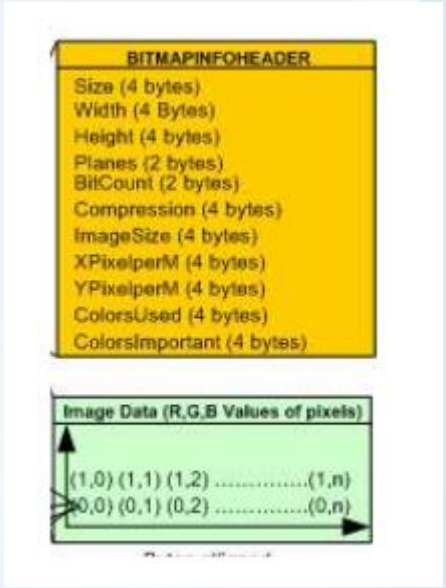
Understanding Bitmap Structure:

https://binaryworld.net/main/CodeDetail.aspx?CodeId=3685



We don't have this
Because we use 24-bits bitmap

We only use these blocks:



wojtasz.aleksander@gmail.com

CHAPTER 1 - Theory

Transfering DIB data into Microsoft Access Image Control Data

```
Private MemoryPictureTable() As Byte
```

Array of Bytes that will hold our whole Bitmap Data

```
Public Function DIBtoPictureData(Optional DIBnum As Long = 0) As Boolean

' DIBSECTION structure
Dim ds As DIBSECTION

If DIBnum = 0 Then
    ' Fill in our DIBSECTION structure
    lngret = apiGetObject(hDib, LenB(ds), ds)
Else
    ' Fill in our DIBSECTION structure for our Backup DIB
    lngret = apiGetObject(m_hDib2, LenB(ds), ds)
End If

If MemoryPictureTable_counter <> ds.dsBmih.biSizeImage + 40 Then
' Allow 40 Bytes for the DIBHeader

  ReDim MemoryPictureTable(ds.dsBmih.biSizeImage + 40)
  MemoryPictureTable_counter = ds.dsBmih.biSizeImage + 40
End If

If DIBnum = 0 Then
    apiCopyMemory MemoryPictureTable(40), ByVal m_lPtr, ds.dsBmih.biSizeImage
Else
    apiCopyMemory MemoryPictureTable(40), ByVal m_lPtr2, ds.dsBmih.biSizeImage
End If

apiCopyMemory MemoryPictureTable(0), ds.dsBmih, 40

m_ImageControl.PictureData = MemoryPictureTable

End Function
```

Declaration of DIBSECTION Structure

Fill in structure with current data taken from DIB

Resize the Array accordingly to hold current bitmap bits + 40 bytes for bitmap info header

Copying bitmap bits to the range from 40-th bit

Copying „BITMAP INFO HEADER” structure data taken from DIBSECTION to first 40 bits of memory array

Assign filled in Array of bytes to PictureData property of Ms Access Image Control

wojtasz.aleksander@gmail.com

CHAPTER 2 - Basic drawing operations

GDI 32 libray saves the following parameters for all these drawing operations for as long as they remain unchanged:

- Background Color (Brush)

```
Dim hNewBrush As LongPtr
Dim hOldBrush As LongPtr

hNewBrush = apiCreateSolidBrush(m_FillColor)

' Select new brush onto our DC
hOldBrush = SelectObject(m_hDC, hNewBrush)
```

```
Private Declare PtrSafe Function apiCreateSolidBrush Lib "gdi32" _
Alias "CreateSolidBrush" _
(ByVal crColor As Long) As LongPtr
```

- Fore Color (Pen Color + Line Style)

```
  Dim hNewPen As LongPtr
  Dim hOldPen As LongPtr

  hNewPen = apiCreatePen(style, thickness, m_ForeColor)

  hOldPen = SelectObject(m_hDC, hNewPen)
```

```
Private Declare PtrSafe Function apiCreatePen Lib "gdi32" Alias "CreatePen" _
    (ByVal nPenStyle As Long, ByVal nWidth As Long, ByVal crColor As Long) As LongPtr
```

- Font Settings (Font Family , Font Size, Font Attributes: Bold, Underline)

```
Dim lNewFont, lOldFont
Dim FontSize As Long
FontSize = m_FontSize
FontSize = CLng(CDbl(FontSize) * CDbl(1.6))

lNewFont = CreateFont(FontSize, 0, 0, 0, m_FontWeight, _
                      m_FontItalic, m_FontUnderline, False, _
                      0, 7, 16, IIf(False, 4, 0), 0, m_FontName)

lOldFont = SelectObject(m_hDC, lNewFont)

DeleteObject (lOldFont)
```

```
Private Declare PtrSafe Function CreateFont Lib "gdi32" Alias _
        "CreateFontA" (ByVal H As Long, ByVal W As Long, ByVal E As Long, ByVal o As Long, ByVal W As Long, ByVal i As Long, _
        ByVal u As Long, ByVal S As Long, ByVal C As Long, ByVal OP As Long, ByVal CP As Long, ByVal Q As Long, _
        ByVal PAF As Long, ByVal F As String) As LongPtr
```

wojtasz.aleksander@gmail.com

CHAPTER 2 - Basic drawing operations

Draw Lines

```
Dim hNewPen As LongPtr
Dim hOldPen As LongPtr

hNewPen = apiCreatePen(style, thickness, m_ForeColor)

hOldPen = SelectObject(m_hDC, hNewPen)

Call apiMoveToEx(m_hDC, x1, y1, ByVal 0&)
LineTo m_hDC, x2, y2

Call SelectObject(m_hDC, hOldPen)
Call DeleteObject(hNewPen)
```

```
Private Declare PtrSafe Function apiMoveToEx Lib "gdi32" Alias "MoveToEx" _
(ByVal hdc As LongPtr, ByVal X As Long, ByVal Y As Long, lpPoint As Any) As Long
```

```
Private Declare PtrSafe Function LineTo Lib "gdi32" _
(ByVal hdc As LongPtr, ByVal X As Long, ByVal Y As Long) As Long
```

LINES STYLES

SOLID = 0
DASH = 1 ' -------
DOT = 2 ' .....
DASHDOT = 3 ' _._._._
DASHDOTDOT = 4 ' _.._.._

CreatePen returns a pen with the specified width but with the PS_SOLID style if you specify a width greater than one for the following styles: PS_DASH, PS_DOT, PS_DASHDOT, PS_DASHDOTDOT.

wojtasz.aleksander@gmail.com

CHAPTER 2 - Basic drawing operations

Draw Rectangles

```
Dim hNewBrush As LongPtr
Dim hOldBrush As LongPtr
Dim lngTmpColor As Long

Dim hNewPen As LongPtr
Dim hOldPen As LongPtr

hNewBrush = apiCreateSolidBrush(m_FillColor)

' Select new brush onto our DC
hOldBrush = SelectObject(m_hDC, hNewBrush)

' Use a NULL Pen so ther is no Border around
' the rectangle
hNewPen = apiCreatePen(PS_SOLID, BorderThickness, m_ForeColor)
hOldPen = SelectObject(m_hDC, hNewPen)

Call Rectangle(m_hDC, x1, y1, x2, y2)


' Cleanup
Call SelectObject(m_hDC, hOldBrush)
Call DeleteObject(hNewBrush)

Call SelectObject(m_hDC, hOldPen)
Call DeleteObject(hNewPen)
```

```
Private Declare PtrSafe Function Rectangle Lib "gdi32" _
(ByVal hdc As LongPtr, ByVal x1 As Long, ByVal y1 As Long, _
ByVal x2 As Long, ByVal y2 As Long) As Long
```

wojtasz.aleksander@gmail.com

CHAPTER 2 - Basic drawing operations

Draw Text

- Step1 : Activate Current Font

```
Dim lNewFont, lOldFont
Dim FontSize As Long
FontSize = m_FontSize
FontSize = 16

lNewFont = CreateFont(FontSize, 0, 0, 0, m_FontWeight, _
                      m_FontItalic, m_FontUnderline, False, _
                      0, 7, 16, IIf(False, 4, 0), 0, m_FontName)

lOldFont = SelectObject(m_hDC, lNewFont)

DeleteObject (lOldFont)
```

```
Private Declare PtrSafe Function CreateFont Lib "gdi32" Alias _
        "CreateFontA" (ByVal H As Long, ByVal W As Long, ByVal E As Long, ByVal o As Long, ByVal W As Long, ByVal i As Long, _
        ByVal u As Long, ByVal S As Long, ByVal C As Long, ByVal OP As Long, ByVal CP As Long, ByVal Q As Long, _
        ByVal PAF As Long, ByVal F As String) As LongPtr
```

- Step2 : Draw Text

```
Dim lngret As Long

lngret = apiSetTextColor(m_hDC, m_ForeColor
lngret = apiSetBkColor(m_hDC, m_BackColor)
lngret = SetBkMode(m_hDC, m_BackMode)

Dim MyRect As RECT

MyRect.left = left
MyRect.right = right
MyRect.top = top
MyRect.bottom = bottom

If align = 0 Then al = DT_CENTER Or DT_WORDBREAK
If align = 1 Then al = DT_LEFT Or DT_WORDBREAK
If align = 2 Then al = DT_RIGHT Or DT_WORDBREAK

lngret = DrawTextEx(m_hDC, StrPtr(strText), Len(strText), MyRect, al, ByVal 0&)
```

```
Private Declare PtrSafe Function DrawTextEx Lib "user32" Alias "DrawTextExW" _
(ByVal hdc As LongPtr, ByVal lpsz As LongPtr, ByVal N As Long, lpRect As RECT, _
ByVal un As Long, lpDrawTextParams As Any) As Long
```

wojtasz.aleksander@gmail.com

CHAPTER 2 - Basic drawing operations

Draw Pictures

- Step1 : Prepare Picture – Load Picture into memory (create another DIB and paste picture data)

```
Private Function DIBFromFile(pFile As String, ByRef pDIB As tDIB) As Boolean
    Dim lImg As Object
    Dim lOldImg
    Dim lLoadhBmp As bitmap
    Dim lBmp As BITMAPINFO
    Dim lloaddc
    Dim lhdc

    On Error GoTo Error_handler  ' getting the   DC
    lhdc = GetDC(0)

    lloaddc = CreateCompatibleDC(lhdc)  ' Create DC
    Set lImg = LoadPicture(pFile)  ' Loading the  image from provided path
    lOldImg = SelectObject(lloaddc, lImg)  ' Selects the image bitmap in the display context


    pDIB.hdc = CreateCompatibleDC(lhdc)  ' Creation of a new DC in input pDIB
    SetGraphicsMode pDIB.hdc, GM_ADVANCED  'Change graphic Mode to extend win32

    Call GetObjectBmp(lImg.handle, LenB(lLoadhBmp), lLoadhBmp)  ' fill in LLoadBmp structure with loded picture data

    '--filli in BITMAPINFO structure
    lBmp.bmiHeader.biSize = LenB(lBmp.bmiHeader)  ' Image Bitmap Header
    lBmp.bmiHeader.biHeight = lLoadhBmp.bmHeight
    lBmp.bmiHeader.biWidth = lLoadhBmp.bmWidth
    lBmp.bmiHeader.biCompression = BI_RGB
    lBmp.bmiHeader.biBitCount = 24
    lBmp.bmiHeader.biPlanes = 1
    lBmp.bmiHeader.biSizeImage = (lLoadhBmp.bmWidth * lLoadhBmp.bmHeight * 3) + (4 - ((lLoadhBmp.bmWidth * 3) Mod 4)) * lLoadhBmp.bmHeight

    pDIB.hDIB = CreateDIBSection(pDIB.hdc, lBmp, DIB_RGB_COLORS, pDIB.DIBPTR, 0, 0) ' Creation new DIB
    pDIB.hOldDIB = SelectObject(pDIB.hdc, pDIB.hDIB)

    BitBlt pDIB.hdc, 0, 0, lLoadhBmp.bmWidth, lLoadhBmp.bmHeight, lloaddc, 0, 0, SRCCOPY  ' If no precise size or Anti Aliasing filter active, then copies the image in its original
    ' Updating global bitmap data
    pDIB.BI = lBmp
    pDIB.width = lBmp.bmiHeader.biWidth
    pDIB.height = lBmp.bmiHeader.biHeight
    ' Release DC
    ReleaseDC 0, lhdc
    ' Deleting temporary objects
    DeleteObject (SelectObject(lloaddc, lOldImg))
    DeleteDC lloaddc
    DIBFromFile = True
    On Error GoTo 0
    Exit Function
Error_handler:
    ' ' Release DC
    ReleaseDC 0, lhdc
    ' Delete temporary objects
    If lOldImg <> 0 Then DeleteObject (SelectObject(lloaddc, lOldImg))
    DeleteDC lloaddc
    DIBFromFile = False
End Function
```

pDIB input parameter is the Structure tDIB - this input parameter is used to get back data of created Picture (using ByRef)

```
Private Type tDIB
    hdc As LongPtr
    width As Long
    height As Long
    hDIB As LongPtr
    DIBPTR As LongPtr
    hOldDIB As LongPtr
    BI As BITMAPINFO
    bitmap As LongPtr
End Type
```

wojtasz.aleksander@gmail.com

CHAPTER 2 - Basic drawing operations

Draw Pictures

- Step2 : Copy picture data into main DIB

```
lngret = StretchBlt(m_hDC, X, Y, width, height, mem_pict.hdc, 0, 0, mem_pict.width, mem_pict.height, SRCCOPY)
```

```
Private Declare PtrSafe Function StretchBlt Lib "gdi32" (ByVal hdc As LongPtr, _
ByVal X As Long, ByVal Y As Long, ByVal nWidth As Long, _
ByVal nHeight As Long, ByVal hSrcDC As LongPtr, ByVal xSrc As Long, _
ByVal ySrc As Long, ByVal nSrcWidth As Long, ByVal nSrcHeight As Long, _
ByVal dwRop As Long) As Long
```

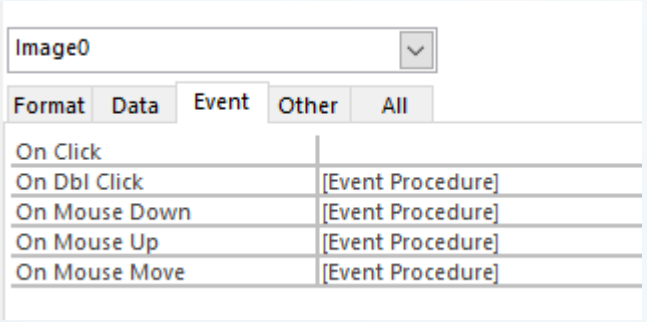
The **StretchBlt** function copies a bitmap from a source rectangle into a destination rectangle, stretching or compressing the bitmap to fit the dimensions of the destination rectangle, if necessary. The system stretches or compresses the bitmap according to the stretching mode currently set in the destination device context.

C++

```
BOOL StretchBlt(
  [in] HDC   hdcDest,
  [in] int   xDest,
  [in] int   yDest,
  [in] int   wDest,
  [in] int   hDest,
  [in] HDC   hdcSrc,
  [in] int   xSrc,
  [in] int   ySrc,
  [in] int   wSrc,
  [in] int   hSrc,
  [in] DWORD rop
);
```

CHAPTER 3 - Deeper Description of Mouse events on MS Access form controls

In MS Access we use the following events:



We use **TwipsPerPixe**l to convert native MSAccess units – twips to memory picture units - pixels

```
Public Function TwipsPerPixel() As Double

    'Handle to device
    Dim lngDC As LongPtr
    Dim lngPixelsPerInch As Long
    Const nTwipsPerInch = 1440

    lngDC = GetDC(0)
    lngPixelsPerInch = apiGetDeviceCaps(lngDC, LOGPIXELSX)

    TwipsPerPixel = CDbl(nTwipsPerInch) / CDbl(lngPixelsPerInch)

End Function
```

Sample event handler code for image control mouse events:

```
Private Sub Image0_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)


x_down = CLng(CDbl(x) / twips_per_pixel)
y_down = CLng(CDbl(y) / twips_per_pixel)

If Button = 2 Then  'if the right mouse  button then...

      If mouse_state <> 1 Then
          state_r_menu = 3  'grid inside job rectangle
      Else
          Call set_state_r_menu(x_down, y_down)  'resolve state on basis of x and y coordinates
      End If

   Exit Sub
End If

If Button = 1 Then l_button = True
```

```
Private Sub Image0_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)

Dim cor_x As Long
Dim cor_y As Long

cor_x = CLng(CDbl(x) / twips_per_pixel)
cor_y = CLng(CDbl(y) / twips_per_pixel)

If l_button = False Then
   Call verify_mouse_state(cor_x, cor_y)
End If
```

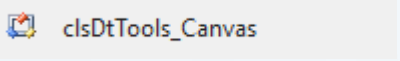
https://en.wikipedia.org/wiki/Twip

A **twip** (abbreviating "twentieth of a point", "twentieth of an inch point",[1] or "twentieth of an Imperial point"[*citation needed*]) is a typographical measurement, defined as $1/20$ of a typographical point. One twip is $1/1440$ inch, or 17.64 µm.[2]

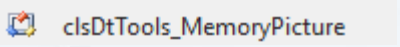
wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

For convenience, all the related drawing memory pictures operations have been embedded into one class module : clsDTTools_Canvas



clsDtTools_MemoryPicture is a small helper class module that helps us keep “memory pictures” handlers data



Declare Canvas object as public variable in the form module

```
Option Compare Database
Option Explicit

Public pb As clsDtTools_Canvas
```

Initialise canvas object in “OnLoad” event handler

```
Private Sub Form_Load()

'set up form resources

Set pb = New clsDtTools_Canvas
' You MUST set the ImageControl prop
pb.ImageControl = Me.Image0
' You MUST set the ImageForm prop
pb.ImageForm = Me
' Clear the Image control to FillColor
```

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Creating tables to store data:

tblJobs

JobID	Customer	PhoneNumber	VIN	Make_model	Job_Descr	JobDate	StartTime	Duration	EndTime	Tech	JobStatus	x	y	JobNumber	UnitNumber	Notes	Click to Add
1	HOGGAN	(333) 333-3333	19UYA3158L00000	Mercedes Benz 190E	Service	2015-04-01	3	2	5	19	2	0	0		77-BGH		
2	2	(612) 111-1112	20QAZ1244L00001	Toyota Corolla	Service of front sus	2015-04-01	1	2	3	7	4	0	0	JJ-XX7	UNO-BH445		
3	Factory Technics I	(554) 334-4555	21WSX0987L00003	Hyundai ZX	Service	2015-04-01	4	2	6	4	5	0	0	24X-55	AK-44-Sx	Here are notes	
4	Jim Terking	(612) 111-1114	22PLMO2345A0005	Porshe Carrera GT	Service	2015-04-01	11	2	13	12	4	0	0		Some new unit num		
8	John Doe	(612) 111-1118	25RTY9876L000008	Oldsmobile Super 88	Service of car body	2015-04-01	1	2	3	0	3	5	5		UNIT NO2223		
9	Ashley Kimbley	(444) 444-4444	26WPR8765L00000	Cadillac Fleetwood	Service	2015-04-01	18	13	31	8	4	0	0		NUM BB-222		
16	Sample customer	(444) 443-3334	01ABC1234D00000	BMW 730i '91	Service	2015-04-01	3	2	5	12	4	0	0		SAMPLE UN 3304		
17	John Cussac	(612) 111-1234	12WDV1234L09875	Citroen 2CV '78	Servicing	2015-04-01	29	13	42	12	1	0	0				
18	3	( ) -4405	WCS33444	Trabant 601	Smarowanie zwroti	2015-04-02	1	2	3	0	1	5	5				
22	Elein Geppers	( )444-4334	d444DVVD	Delorean DMC	Gas pedal linkage h	2015-04-01	7	2	9	15	1	0	0				
23	Mary Jo Fernande	(554) 545-4554	34343343434	Volkswagen Pheton	Paint front bumper	2015-04-01	10	3	13	9	4	0	0		GG-HHTT		
24	anonym	444skd	445-5553	Mercedes Benz w126 2.6 V	This is standard ser	2015-04-01	6	2	8	9	3	0	0		UNIT NO 4445		
27	2	(444) 444-4444	terte	Jaguar Type E	repair engine head	2015-04-01	1	2	3	0	2	20	20		TU-TT456		
28	Customer BBCC			Ndffdfdsd		2015-04-03	6	3	9	12	1	0	0				
29	3	( ) 44-4443	gffdfdggfd	New added job	sddsffsd	2015-04-03	8	2	10	18	1	0	0				
30	4	e3333		New added job	ffdasfdsfdfd	2015-04-03	1	2	3	4	3	0	0				
31	4	( ) -3333		unassigned		2015-04-03	4	2	6	18	5	0	0				
33	3	( )222-2222		New added job		2015-04-02	2	2	4	9	3	0	0				
34	3		4455	New added job	sdfsfsdsfd	2015-04-02	3	2	5	3	3	0	0				
35	Customer	(44) 343-3434	fie33333	Mercedes 190	Repair of suspensic	2015-04-01	1	2	3	15	1	0	0		ttt	Notes	
37	Changed name of	(333) 333-3333	eeeeweef	Syrena 105	sfsdfdsfdsfdsfsd	2015-04-01	8	2	10	12	6	0	0	XX 11	New nubmer		
38	Client A	211 334 33	21ABCD121234	Tomson 33-44	Repair the suspens	2015-04-01	35	8	43	7	2	0	0		UN-33-56		
39	John Haroolds	555 6764 8	0000458654	BUICK ELECTRA '77	repair engine	2015-04-01	4	2	6	15	5	0	0				
40	Perkins Johnatan			New added job	Repair body paint a	2015-04-01	10	7	17	15		0	0	X-BB-A	X-JU-7 unit number		
41				New added job		2015-04-02	1	2	3			20	20				
42	New Customer na			New added job	new job descriptio	2015-04-02	1	2	3			35	35	x-332	New unit number		
43				New added job	Make somer repair	2015-04-02	3	2	5	16	5	0	0		NEW UNIT UN		
44	ff444	(222) 332-3232	ffff	New added job	ffdffdfd	2015-04-02	5	6	11	6	2	0	0	55fg	444334	fffff	
45				New added job		2015-04-09	1	2	3			5	5				
46				New added job		2015-04-09	1	2	3	0		20	20				
47	fd3333ff	(222) 333-3333	VIN CADILLAC	Cadillac	to jest opis	2015-04-09	2	4	6	2	1	0	0		fddfssfdfd		
49	hoggan 2			New added job		2015-04-02	6	4	10	3	2	0	0		22 unit number		
50	Test Cusotmer			New added job	Description 234	2015-04-01	14	2	16	9	2	0	0		333-test112		
51				New added job		2015-05-31	2	5	7	18	2	0	0		test		
52	Anthony Roolck	(333) 456-6666		New added job	Modiry air conditio	2015-04-01	14	2	16	7	3	0	0	555	55-TU-X9	fggfgf	
54	Addam Johnson	(333) 445-6432	44eeee	VW	Repair front bumpe	2015-04-04	5	8	13	2	2	0	0		VWT-TT33		
55				New added job		2015-04-04	2	2	4	4	3	0	0		tttt		
56				New added job	dsffdsdsf	2015-04-01	6	2	8	19	3	0	0		tets		
57	rrrww			New added job	fdsfds	2015-04-01	18	4	22	4	6	0	0		yy6	werrewrew	
58			333dfsfd33	New added job	tty	2015-04-01	1	2	3	0		35	35		test		
59				New added job		2014-05-13	4	5	9	2		0	0				
60				New added job		2014-05-13	4	2	6	6		0	0				
61				New added job		2015-04-02	1	2	3			170	5	4444			
62				New added job		2015-04-01	21	2	23	9	2	0	0	5567	test number 24		
64				New added job		2015-04-01	19	12	31	19	6	0	0		Some information h		
65				New added job		2015-04-19	23	2	25	2		0	0		test		
66	yytt			New added job		2015-04-01	1	2	3	0		170	5				
67				New added job		2015-04-01	16	7	23	3		0	0				
68	Client bbb			New added job		2015-04-01	26	5	31	14		0	0		333gh		

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Creating tables to store data:

tblTechs

TechID	TechName	selected	Click to Add
4	Tech A	☐	
2	Tech B	☐	
3	Tech C	☐	
14	Tech D	☐	
5	Tech E	☐	
6	Tech F	☐	
7	Tech G	☑	
8	Tech H	☑	
9	Tech I	☐	
12	Tech J	☐	
18	Tech K	☐	
19	Tech L	☐	
15	Tech M	☐	
17	Tech N	☐	
16	Tech O	☐	
✳	(New)	☐	

tblTechInDay

RowId	TechID	DateAssign	Order	Click to Add
298	4	2023-07-06	1	
299	2	2023-07-06	2	
300	3	2023-07-06	3	
301	14	2023-07-06	4	
302	5	2023-07-06	5	
303	6	2023-07-06	6	
304	7	2023-07-06	7	
305	8	2023-07-06	8	
306	9	2023-07-06	9	
307	12	2023-07-06	10	
308	18	2023-07-06	11	
309	19	2023-07-06	12	
310	15	2023-07-06	13	
311	17	2023-07-06	14	
312	16	2023-07-06	15	

Hours_dict

HourID	HourDescr	Click to Add
1	0:00 AM	
2	0:30 AM	
3	1:00 AM	
4	1:30 AM	
5	2:00 AM	
6	2:30 AM	
7	3:00 AM	
8	3:30 AM	
9	4:00 AM	
10	4:30 AM	
11	5:00 AM	
12	5:30 AM	
13	6:00 AM	
14	6:30 AM	
15	7:00 AM	
16	7:30 AM	
17	8:00 AM	
18	8:30 AM	
19	9:00 AM	
20	9:30 AM	
21	10:00 AM	
22	10:30 AM	
23	11:00 AM	
24	11:30 AM	
25	12:00 PM	
26	12:30 PM	
27	1:00 PM	
28	1:30 PM	
29	2:00 PM	
30	2:30 PM	
31	3:00 PM	
32	3:30 PM	
33	4:00 PM	
34	4:30 PM	
35	5:00 PM	
36	5:30 PM	
37	6:00 PM	
38	6:30 PM	
39	7:00 PM	
40	7:30 PM	

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Declare structures (custom data types) and memory arrays to store data in memory:

```
'-------------------------------------------------
Private Type Employee

     emp_id As Long
     Name As String

End Type

Private emp_counter As Long
Private Employees(1 To 50) As Employee

'-------------------------------------------------
```

```
'-------------------------------------------------

Private Type Job

     JobID As Long
     row As Integer
     start_time As Integer
     Duration As Integer
     x As Integer
     y As Integer
     color_no As Integer
     Make_mod As String
     VIN As String
     Cust_Name As String
     Teleph As String
     Descr As String
     Unit_Nuber As String

End Type

Private job_counter As Long
Private Jobs(1 To 200) As Job

'-------------------------------------------------
```

```
Private Type layout_job

  x1 As Long
  x2 As Long
  y1 As Long
  y2 As Long
  job_id As Long
  jobs_order As Integer

End Type

Private layout_jobs(1 To 500) As layout_job
Private lay_counter As Long
```

```
Private hour dict(1 To 24) As String   'hours dictionary to fill in labels at the top of the form
```

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Current Date is kept in public variable “current_date” - this is a form module level variable:

```
Public current_date As Date    'current date from callendar
```

Populate memory arrays with data from MS Access tables:

```
'------------------------------------------------------------
Public Sub load_hour_dict()

Dim i As Integer

hour_dict(1) = "0 AM"
hour_dict(2) = "1 AM"
hour_dict(3) = "2 AM"
hour_dict(4) = "3 AM"
hour_dict(5) = "4 AM"
hour_dict(6) = "5 AM"
hour_dict(7) = "6 AM"
hour_dict(8) = "7 AM"
hour_dict(9) = "8 AM"
hour_dict(10) = "9 AM"
hour_dict(11) = "10 AM"
hour_dict(12) = "11 AM"
hour_dict(13) = "12 AM"
hour_dict(14) = "1 PM"
hour_dict(15) = "2 PM"
hour_dict(16) = "3 PM"
hour_dict(17) = "4 PM"
hour_dict(18) = "5 PM"
hour_dict(19) = "6 PM"
hour_dict(20) = "7 PM"
hour_dict(21) = "8 PM"
hour_dict(22) = "9 PM"
hour_dict(23) = "10 PM"
hour_dict(24) = "11 PM"

End Sub
```

```
Public Sub load_tech()

Dim strSql As String

strSql = " SELECT tblTechInDay.TechID, tblTechs.TechName " _
       & " FROM tblTechInDay LEFT JOIN tblTechs ON tblTechInDay.TechID = tblTechs.TechID " _
       & " WHERE (((tblTechInDay.DateAssign) = #" & conv_date(current_date) & "#)) " _
       & " ORDER BY tblTechInDay.Order "


Dim db As DAO.Database
Dim rs As DAO.Recordset

Set db = CurrentDb()
Set rs = db.OpenRecordset(strSql)

If rs.EOF = True Then
  emp_counter = 0
Else
  emp_counter = 1
End If

     Do Until rs.EOF = True

       Employees(emp_counter).emp_id = rs(0)
       Employees(emp_counter).Name = Nz(rs(1), "Empty")

       emp_counter = emp_counter + 1
       rs.MoveNext
     Loop

rs.Close
Set rs = Nothing
Set db = Nothing

If emp_counter > 0 Then
  emp_counter = emp_counter - 1  'moves back counter by 1 position
End If


End Sub
```

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Populate memory arrays with data from MS Access tables:

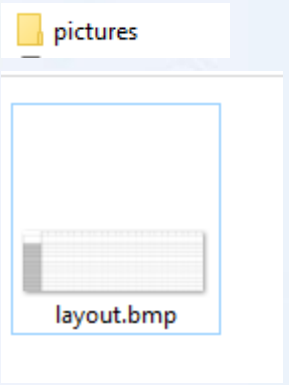
```
Public Sub load_jobs()

Dim strSql As String

strSql = " SELECT tblTechInDay.TechID, tblTechInDay.Order, tblJobs.JobID, tblJobs.StartTime, " _
       & " tblJobs.Duration, tblJobs.x, tblJobs.y, tblJobs.JobStatus, tblJobs.Make_model, tblJobs.VIN, " _
       & " tblJobs.Customer, tblJobs.PhoneNumber, tblJobs.Job_Descr, tblJobs.JobDate, tblJobs.UnitNumber " _
       & " FROM tblTechInDay RIGHT JOIN tblJobs ON (tblTechInDay.TechID = tblJobs.Tech) AND " _
       & " (tblTechInDay.DateAssign = tblJobs.JobDate) " _
       & " WHERE (((tblJobs.JobDate) = #" & conv_date(current_date) & "#)) ORDER BY tblTechInDay.Order, tblJobs.JobID "

Dim db As DAO.Database
Dim rs As DAO.Recordset

Set db = CurrentDb()
Set rs = db.OpenRecordset(strSql)

If rs.EOF <> True Then
  job_counter = 1
Else
  job_counter = 0
End If

     Do Until rs.EOF = True

       Jobs(job_counter).JobID = rs(2)
       Jobs(job_counter).row = Nz(rs(1), 0)
       Jobs(job_counter).start_time = rs(3)
       Jobs(job_counter).Duration = rs(4)
       Jobs(job_counter).x = Nz(rs(5), 0)
       Jobs(job_counter).y = Nz(rs(6), 0)
       Jobs(job_counter).color_no = Nz(rs(7), 1)
       Jobs(job_counter).Make_mod = Nz(rs(8), "")
       Jobs(job_counter).VIN = Nz(rs(9), "")
       Jobs(job_counter).Cust_Name = Nz(rs(10), "")
       Jobs(job_counter).Teleph = Nz(rs(11), "")
       Jobs(job_counter).Descr = Nz(rs(12), "")
       Jobs(job_counter).Unit_Nuber = Nz(rs(14), "")

       job_counter = job_counter + 1
       rs.MoveNext
     Loop

rs.Close
Set rs = Nothing
Set db = Nothing

If job_counter > 0 Then
  job_counter = job_counter - 1 'moves back jobs counter
End If

End Sub
```

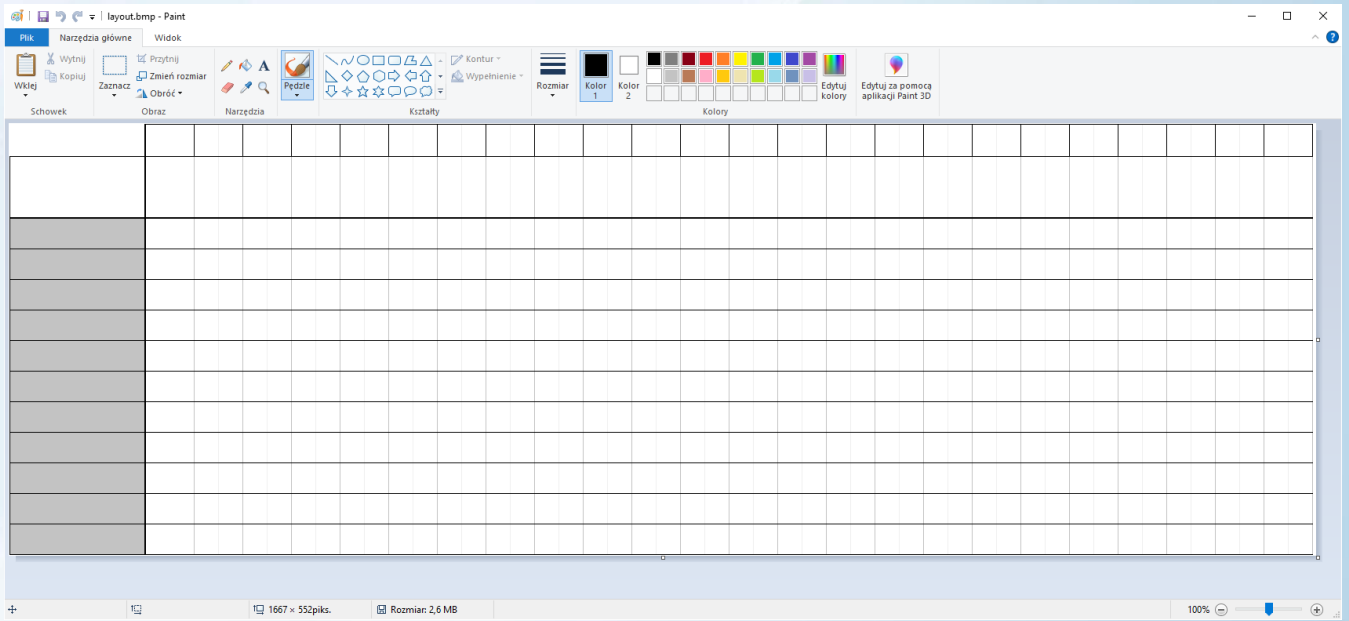
wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Initialise memory picture (background picture) which is loaded from file in folder “pictures”:

```
Public Sub initialize_pictures()

On Error GoTo err_handl

Debug.Print CurrentProject.Path & "pictures\layout.bmp"

Dim rez As Boolean
Call pb.inicialize_dib(CurrentProject.Path & "\pictures\layout.bmp", "layout")


Me.Image0.Picture = ""  'removes lodaed picture

Exit Sub

err_handl:

MsgBox "Err type: " & Err.Description

Exit Sub
```





wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Initialise memory picture (background picture) which is loaded from file in folder “pictures”:

```
Public Sub Draw()

Call pb.Draw_Memory_Picture("layout", 1, 1, 1667, 552)

Call draw_employees

Call draw_jobs

Call draw_hours_at_top

pb.DIBtoPictureData


End Sub
```

```
Public Sub draw_hours_at_top()

'activate the font

 pb.FontSize = CInt(14)
 pb.FontName = "Calibri"
 pb.FontBold = True
 pb.activate_font

'draw hour labels at the top of the form

Dim i As Integer
Dim ret As Boolean

Dim width As Integer, height As Integer, left As Integer, top As Integer, bottom As Integer

 top = 3
 width = 62
 height = 41
 left = 176
 bottom = top + height

 Dim cur_hour As String  'current hour


For i = 1 To 24

   cur_hour = hour_dict(i)  'take from hour_dict (ionary) array

    Call pb.DrawRectangle(left - 1, top, left + width, top + height, RGB(200, 200, 200), 1)


   Call pb.put_text_in_my_place(cur_hour, left, left + width, top + 10, top + 30, 0, 0)

   left = left + width 'increase left to the  position position of next hours rectangle

Next i


End Sub
```

```
Public Sub draw_employees()

Dim kol As Long
Dim i As Integer

kol = RGB(185, 234, 255)

Call draw_name_rectangle("UNASSIGNED", 0, RGB(255, 255, 255))

Dim pos As Integer  'real position

For i = 1 To emp_counter

   pos = i - vscrol
   If pos > 0 And pos < 12 Then
     Call draw_name_rectangle(Employees(i).Name, pos, kol)
   End If


Next i

End Sub
```

```
Public Sub draw_name_rectangle(Name As String, row As Integer, color As Long)

 pb.FontSize = CInt(11)
 pb.FontName = "Calibri"
 pb.FontBold = True
 pb.activate_font

 Dim width As Integer, height As Integer, left As Integer, top As Integer

 width = 172
 height = 39
 left = 3

 top = 123 + ((row - 1) * 39)


 If row = 0 Then top = 65


 If row <> 0 Then
   Call pb.DrawRectangle(left - 1, top - 1, left + width, top + height, color, 1)
 End If

 ret = pb.put_text_in_my_place(Name, left, left + width, top + 10, top + height - 7, 0, 0)

End Sub
```

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Draw Jobs rectangles:

```
Public Sub Draw()

Call pb.Draw_Memory_Picture("layout", 1, 1, 1667, 552)

Call draw_employees

Call draw_jobs

Call draw_hours_at_top

pb.DIBtoPictureData


End Sub
```

```
Public Function get_full_color(color_no As Integer) As Long

    Select Case color_no

    Case 1
       get_full_color = RGB(254, 252, 175) 'light yellow
    Case 2
       get_full_color = RGB(128, 255, 0) 'green
    Case 3
       get_full_color = RGB(255, 201, 14) 'orange
    Case 4
       get_full_color = RGB(255, 255, 0) 'yellow
    Case 5
       get_full_color = RGB(255, 121, 121) 'light red
    Case 6
       get_full_color = RGB(195, 195, 195) 'gray
    Case Else
       get_full_color = RGB(210, 210, 210) 'light gray

    End Select

End Function
```

```
Public Sub draw_jobs()

   lay_counter = 0

Dim i As Long
Dim pos As Integer

For i = 1 To job_counter

   If Jobs(i).row = 0 Then
     pos = Jobs(i).row      'for unassigned draw the with the row = 0
   Else
     pos = Jobs(i).row - vscrol
     If pos = 0 Then pos = -1  'if at assigned job pos in result is 0 then change it to -1 to not show this job at unassinged ar
   End If

 If pos > -1 And pos < 12 Then
   Call draw_job_rectangle(pos, _
                           Jobs(i).start_time, _
                           Jobs(i).Duration, _
                           Jobs(i).x, _
                           Jobs(i).y, _
                           Jobs(i).color_no, _
                           Jobs(i).JobID, _
                           Jobs(i).Make_mod, _
                           Jobs(i).VIN, _
                           Jobs(i).Cust_Name, _
                           Jobs(i).Teleph, _
                           Jobs(i).Descr, _
                           Jobs(i).Unit_Nuber, _
                           i)

  End If
Next i

'----------------------------
```

```
Public Sub draw_job_rectangle(row As Integer, start_time As Integer, Duration As Integer, _
x As Integer, y As Integer, color_no As Integer, JobID As Long, Make_mod As String, _
VIN As String, Cust_Name As String, Teleph As String, Descr As String, UnitNumber As String, order As Long)

Dim left As Integer, right As Integer, top As Integer, bottom As Integer

'--top-------------
If row = 0 Then
  top = 44 + y
  left = 176 + x
  right = 176 + (Duration * 62) + x

Else
  top = 123 + ((row - 1) * 39) + y

  left = 176 + ((start_time - 1) * 31) + x
  right = 176 + ((start_time + Duration - 1) * 31) + x

End If

bottom = top + 39

'--------------------

Dim color As Long

color = get_full_color(color_no)

'pb.ForeColor = RGB(255, 0, 0)
Call pb.DrawRectangle(left - 1, top - 1, right, bottom, color, 1)

pb.ForeColor = RGB(0, 0, 0)  'back to black color - in case actual color was red

'-remember current dispatch position into array of layout_jobs---------
lay_counter = lay_counter + 1

layout_jobs(lay_counter).x1 = left - 1
layout_jobs(lay_counter).x2 = right
layout_jobs(lay_counter).y1 = top - 1
layout_jobs(lay_counter).y2 = bottom
layout_jobs(lay_counter).job_id = JobID
layout_jobs(lay_counter).jobs_order = order

'-------------------------------------------------------------------------------

'--place UNIT NUMBER
 pb.FontSize = CInt(8)
 pb.FontName = "Calibri"
 pb.FontBold = False
 pb.activate_font

  ret = pb.put_text_in_my_place(CStr(UnitNumber), left + 5, right, top, top + 13, 1, test_col)


'--Customer name ----
 pb.FontSize = CInt(8)
 pb.FontName = "Calibri"
 pb.FontBold = True
 pb.activate_font

  ret = pb.put_text_in_my_place(Cust_Name, left + 5, right, top + 12, top + 25, 1, test_col)

'--Description----
 pb.FontSize = CInt(8)
 pb.FontName = "Calibri"
 pb.FontBold = False
 pb.activate_font

 ret = pb.put_text_in_my_place(Descr, left + 5, right, top + 25, top + 38, 1, test_col)
```

Caprutre current positions
Of rectangles into
layout_jobs

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Reacting to mouse move events over the layout:

```
Private Sub Image0_MouseMove(Button As Integer, _
Shift As Integer, x As Single, y As Single)

Dim cor_x As Long
Dim cor_y As Long

cor_x = CLng(CDbl(x) / twips_per_pixel)
cor_y = CLng(CDbl(y) / twips_per_pixel)

If l_button = False Then
   Call verify_mouse_state(cor_x, cor_y)
End If


If l_button = True Then

   If mouse_state = 12 Then Call verify_mouse_move12(cor_x, cor_y)
   If mouse_state = 11 Then Call verify_mouse_move11(cor_x, cor_y)
   If mouse_state = 10 Then Call verify_mouse_move10(cor_x, cor_y)
End If


End Sub
```

```
Public Sub verify_mouse_state(cor_x As Long, cor_y As Long)

Dim id_last_active As Long
Dim order_last_active As Long
Dim state_last_active As Integer
Dim recovoered_state As Integer


id_last_active = 0
state_last_active = 1


Dim i As Integer

For i = 1 To lay_counter

   recovoered_state = recover_mouse_state_for_job(layout_jobs(i), cor_x, cor_y)

   If recovoered_state <> 1 Then
      id_last_active = layout_jobs(i).job_id
      order_last_active = layout_jobs(i).jobs_order
      state_last_active = recovoered_state

   End If

Next i

If current_job_id <> id_last_active Then


    current_job_id = id_last_active
    current_job_order = order_last_active


    pb.Clear
    Call Draw

End If

If state_last_active <> mouse_state Then
   mouse_state = state_last_active
'-----------------------------------------------------
'-------change screen pointer to SW arrows
    If mouse_state = 3 Or mouse_state = 4 Then
       Application.Screen.MousePointer = 9
    Else
       Application.Screen.MousePointer = 0
    End If
'-----------------------------------------------------
End If
```

```
Private Function recover_mouse_state_for_job(disp As layout_job, _
x As Long, y As Long) As Integer

Dim state As Integer

state = 2  'default value - inside job

'1 test outside

If x < disp.x1 - 2 Or _
   x > disp.x2 + 2 Or _
   y < disp.y1 - 2 Or _
   y > disp.y2 + 2 Then

   state = 1   'outside

End If

'-----------------------------------------------------------
'4 test if in the near of  x1

   If (Abs(x - disp.x1) < 3) And (y > disp.y1) And (y < disp.y2) Then
       state = 3
       GoTo finish
   End If
'-----------------------------------------------------------
'5 test if in the near of  x2

    If (Abs(x - disp.x2) < 3) And (y > disp.y1) And (y < disp.y2) Then
       state = 4
       GoTo finish
    End If
'-----------------------------------------------------------

finish:
   recover_mouse_state_for_job = state

End Function
```

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Reacting to mouse down / up events over the layout:

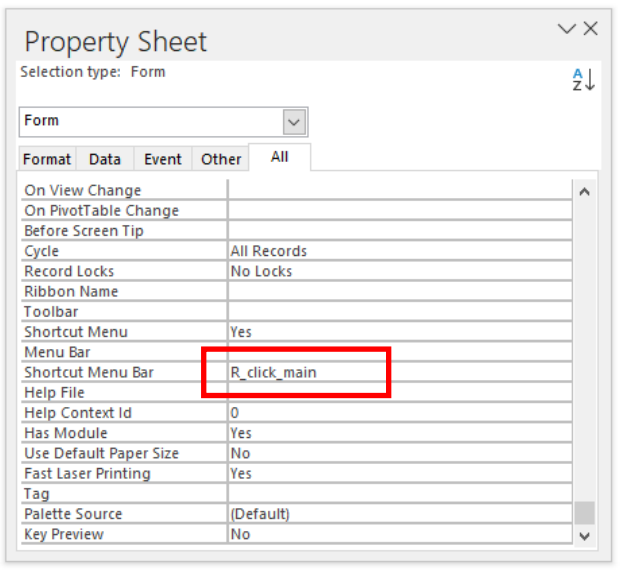
```
Private Sub Image0_MouseDown(Button As Integer, Shift As Integer, _
x As Single, y As Single)


x_down = CLng(CDbl(x) / twips_per_pixel)
y_down = CLng(CDbl(y) / twips_per_pixel)

If Button = 2 Then  'if the right mouse  button then...

      If mouse_state <> 1 Then
          state_r_menu = 3  'grid inside job rectangle
      Else
          Call set_state_r_menu(x_down, y_down)  'resolve state on basis of x and y coordinates
      End If

   Exit Sub
End If


If current_job_id = 0 Then   'if no one dispatch is active leave procedure here
  Exit Sub
End If

x_curr_job_down = Jobs(current_job_order).x
y_curr_job_down = Jobs(current_job_order).y
duration_curr_job_down = Jobs(current_job_order).Duration
start_time_curr_job_down = Jobs(current_job_order).start_time

If Button = 1 Then l_button = True

If mouse_state = 2 Then mouse_state = 12
If mouse_state = 3 Then mouse_state = 10
If mouse_state = 4 Then mouse_state = 11


pb.Clear
Call Draw

End Sub
```

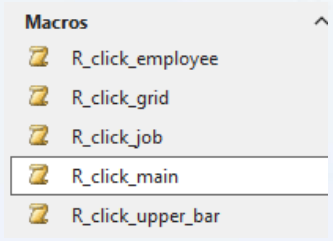
```
Private Sub Image0_MouseUp(Button As Integer, Shift As Integer, _
x As Single, y As Single)

If current_job_order = 0 Then
 Exit Sub
End If

'verify if mouse up coordinates are not the same as mouse down - if y

Dim x_up As Long
Dim y_up As Long

x_up = CLng(CDbl(x) / twips_per_pixel)
y_up = CLng(CDbl(y) / twips_per_pixel)


If x_up = x_down And y_up = y_down Then
   If Button = 1 Then l_button = False
   If mouse_state = 12 Then mouse_state = 2
   'MsgBox "Will load job details"
   Exit Sub
End If


Screen.MousePointer = 11

Dim y_absolute As Long

Dim current_row_base_y
current_row_base_y = find_row_base_y(Jobs(current_job_order).row)

y_absolute = current_row_base_y + Jobs(current_job_order).y

Dim row As Integer
row = set_row_from_y_absolute(y_absolute)

If row > emp_counter Then
'-------------------------------------------------------------

  Jobs(current_job_order).x = x_curr_job_down
  Jobs(current_job_order).y = y_curr_job_down

  'current_job_id = 0
  'current_job_order = 0
  'mouse_state = 1
  If Button = 1 Then l_button = False
```

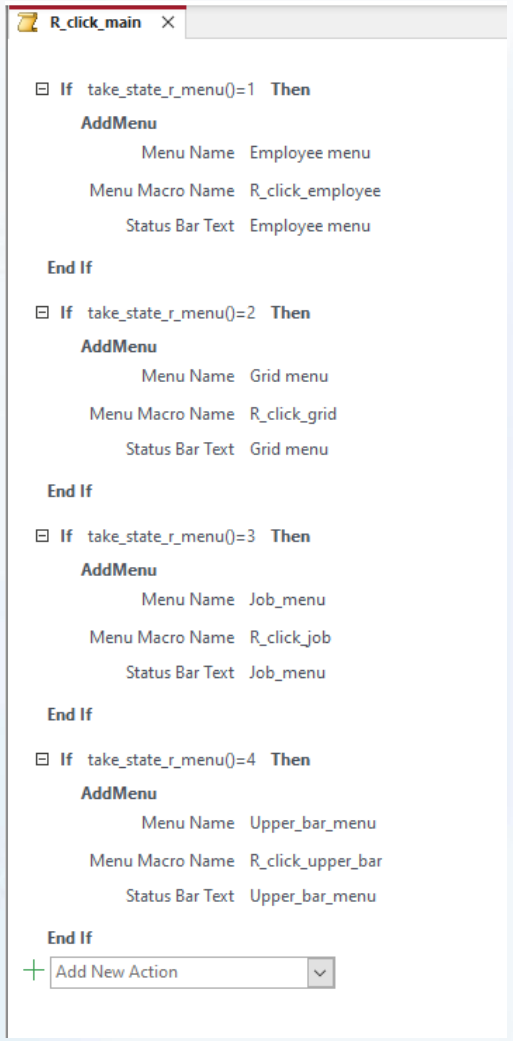
wojtasz.aleksander@gmail.com

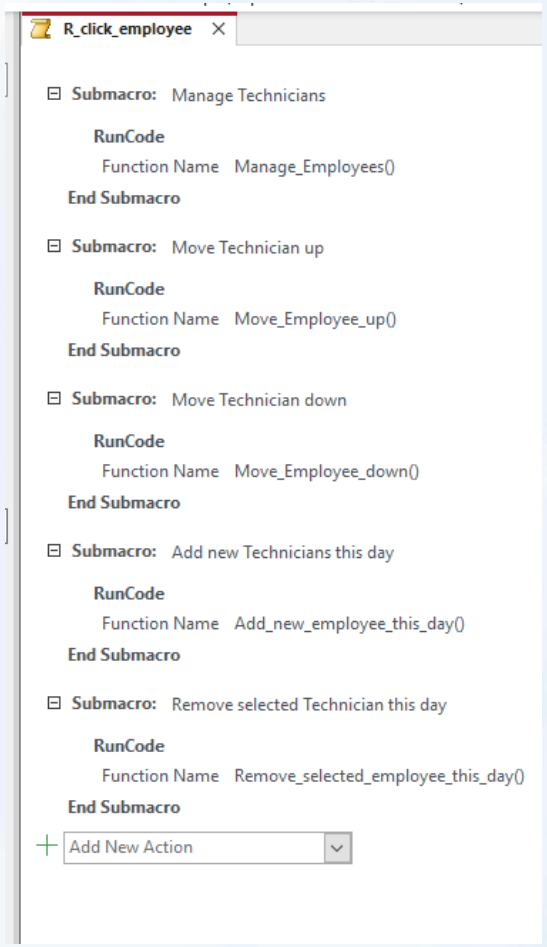
CHAPTER 4 - Putting everything above together:

Right mouse click – context menu









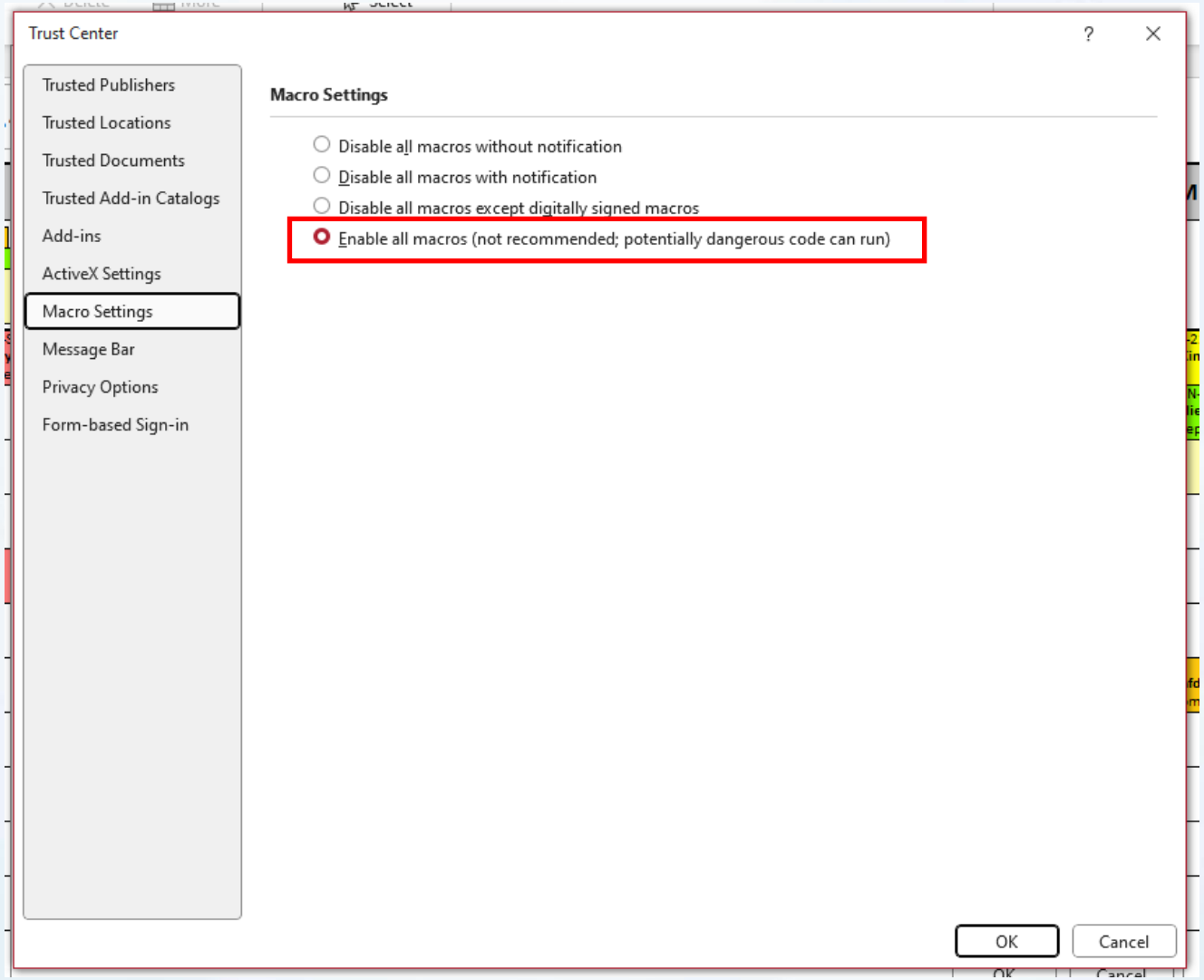

```
Public Function Manage_Employees() As Integer

     DoCmd.OpenForm "tblTechs", acNormal

End Function
```

```
Public Function Move_Employee_up() As Integer

  Dim row As Integer
  row = get_current_row

  If row = 0 Or row > 11 + vscrol Then
     Exit Function
  End If
```

wojtasz.aleksander@gmail.com

CHAPTER 4 - Putting everything above together:

Options - Trust Center - Macro Settings



wojtasz.aleksander@gmail.com