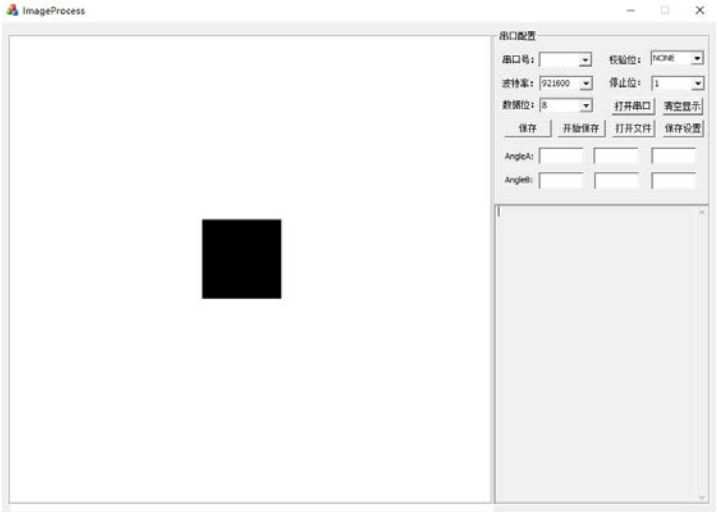
DAY1

今天，首先对整个分散实习任务进行了了解，随后，被安排了任务：运用 MFC 写一个简单的显示界面。

我首先对 MFC 进行了了解：微软基础类库（英语：Microsoft Foundation Classes，简称 MFC）是微软公司提供的一个类库（class libraries），以 C++类的形式封装了 Windows API，并且包含一个应用程序框架，以减少应用程序开发人员的工作量。其中包含大量 Windows 句柄封装类和很多 Windows 的内建控件和组件的封装类。

然后，查阅入门教程，着手实现，完成结果如下：



通过一天的学习，对 MFC 编程有了一些较为浅显的认识，编写 MFC 程序需要包含#include<afxwin.h>头文件，自己创建 MFC 应用程序框架：自定义类继承自 CWinApp 应用程序类，创建 MyApp app 应用程序对象（有且只有一个）。程序入口为 InitInstance()函数。在程序入口函数中创建自定义框架类（MyFrame 继承自 CFrameWnd）。在 MyFrame 类构造函数中创建窗口 Create。显示更新窗口并保存框架类对象指针。

消息映射是一个将消息和成员函数相互关联的表。比如，框架窗口接收到一个鼠标左击消息，MFC 将搜索该窗口的消息映射，如果存在一个处理 WM_LBUTTONDOWN 消息的处理程序，然后就调用 OnLButtonDown。

将消息映射添加到一个类中所做的全部工作：

1）在所操作类中，声明消息映射宏

2）通过放置标识消息的宏来执行消息映射，相应的类将在 BEGIN_MESSAGE_MAP 和 END_MESSAGE_MAP 的调用之间处理消息。

3）对应消息处理函数分别在类中声明，类外定义

DAY2

学习有关位图知识：BMP(Bitmap-File)图形文件是 Windows 采用的图形文件格式，在 Windows 环境下运行的所有图象处理软件都支持 BMP 图象文件格式。位图文件可看成由 4 个部分组成：位图文件头（bitmap-file header）、位图信息头（bitmap-information header）、彩色表（color table）和定义位图的字节(位图数据，即图像数据，Data Bits 或

Data Body)阵列，BMP 文件的数据按照从文件头开始的先后顺序分为四个部分：

- 位图文件头(bmp file header)：　提供文件的格式、大小等信息
- 位图信息头(bitmap information)：提供图像数据的尺寸、位平面数、压缩方式、颜色索引等信息
- 调色板(color palette)：可选，如使用索引来表示图像，调色板就是索引与其对应的颜色的映射表
- 位图数据(bitmap data)：图像数据区

数据段名称	大小（byte）	开始地址	结束地址
位图文件头(bitmap-file header)	14	0000h	000Dh
位图信息头(bitmap-information header)	40	000Eh	0035h
调色板(color table)	由 biBitCount 决定	0036h	未知
图片点阵数据(bitmap data)	由图片大小和颜色定	未知	未知

DAY3

上网查阅关于利用 MFC 显示 BMP 图像的资料进行学习。经过不懈的努力，最后完成今日任务：显示 BMP 图片。真的是不容易啊!

记录一下具体步骤，如下：

1.创建位图并调用函数 LoadImage 装载图标、光标或位图。

```
HBITMAP m_hBitmap;
m_hBitmap=(HBITMAP)LoadImage(HINSTANCE hinst,LPCTSTR lpszName,UINT uType,int cxDesired,int cyDesired,UINT fuLoad)
```

2.定义并创建一个内存设备环境 DC,调用函数 CreateCompatibleDC 创建兼容的 DC.

```
CDC dcBmp; dcBmp.CreateCompatibleDC(pDC) ;
```

3.定义 BITMAP 变量,调用函数 GetBitmap 将图片载入位图中,该定义是为后去图像的长宽等信息.

```
BITMAP m_bmp; m_bitmap.GetBitmap(&m_bmp);
```

4.调用函数 SelectObject 将位图选入兼容内存设备环境 DC 中.

```
dcBmp.SelectObject(&m_bitmap);
```

5.将兼容的 DC 中的位图填到当前 DC 中,调用函数 BitBlt 或 strechBlt 显示图像.

(1).BitBlt()该函数对指定的源设备环境区域中的像素进行位块(bit_block)转换,以传送到目标

设备环境.

pDC->BitBlt(0,0,m_bmp.bmWidth,m_bmp.bmHeight,&dcBmp,0,0,SRCCOPY);

(2)stretchBlt()该函数从源矩形中复制位图到目标矩形,必要是按目标设备设置的模式进行图像拉伸或压缩.

```
pDC->StretchBlt(0,0,m_nDrawWidth,m_nDrawHeight,&dcBmp,0,0,m_bmp.bmWidth,m_bmp.bmHeight,SRCCOPY);
```

6.恢复临时 DC 的位图,删除 CreateCompatibleDC 得到的图片 DC,删除内存中的位图及释放系统资源.

```
dcBmp.SelectObject(pbmpOld); DeleteObject(&m_bitmap);  dcBmp.DeleteDC();
```

DAY4

今天指导老师和我讲了一下 USB 转串口的一些东西，听的有点玄乎，不是很懂…于是就上网查阅了一些博客进行学习：涉及的知识点主要就是 USB 转串口的获得 USB 设备属性，枚举接口，挂上串口的过程。数据流的过程主要就是 COM_OPEN -> COM_READ -> COM_WRITE -> COM_CLOSE。

COM_OPEN 是一个相当重要的函数，这个函数自然石不需要做任何修改这是属于微软 MDD 的函数，其对应的两个文件。COM_MDD2,SERPDDCM,是基本上都不需要修改的。但是其流程需要好好的分析。线程函数是在初始化的时候如何就进入等待状态的，而 CMiniThread 这个类里就有 ThreadStart()这个成员函数，作用是清除挂起状态（用到了这个函数 ResumeThread）。构造函数创建了线程 CMiniThread::ThreadProc，并且处于挂起状态，ThreadProc 调用了 ThreadRun()这个成员函数（是个纯虚函数，在 CPddXXXUart 里有实现，而这个实现就是 IST）。当这个函数执行后那么整个初始化基本上就完成了其最后的结果就是让这个线程等着 USB 那边的数据过来。其实感觉还是不大明白，明天去问问指导老师。

DAY5

今天上手串口调试工具，真的是太反人类了，但也许是我不大会用。

界面如下：

首先是遇到的第一个问题：找 COM 号，上网查了一下具体步骤：系统与安全->管理

工具->计算机管理->设备管理->com 口。接下来的事情就特别麻烦！查阅资料查了好久，就是因为两个协议的问题。在正常的数据传输系统中，传输数据的端口都有固定的格式要求，首先 USB 有数据输入，经过电平转换进入 mcu,注意数据仍是串口协议，非串口协议的数据 SV613 模块不能识别处理。所以，在我打开点击后，若进行插拔，则 USB 进行了重新的协议，但是 COM 协议还是未更新，还是原来的参数，故继续发送信号会显示串口被占用！发送失败！解决完这个问题后还有问题，那就是时不时的会显示串口尚未连接！发送失败！这个我查阅了好久，最后竟然是我 usb 接口接触有问题导致的。还有问题：Cannot set COM port parameters，最后查到的答案是 It is FS.There was only one instance of KiTTY running.It works fine when I use it to communicate with the board using the boards UART interface. The UART interface IC is connected to the processor's USART2 pins.

DAY6

今天的任务是通过 Win 系统来对外围设备进行通信处理，做串口方面的程序，使用 CreateFile 打开串口通信端口。在对串口操作之前，需要首先打开串口。

在 Window 32bit 的操作系统上， 将串口（通信设备）作为文件来处理，所以串口的打开、关闭、读写所使用的 API 函数与文件操作一样。所以打开串口使用 CreateFile 函数，读写串口使用 ReadFile、WriteFile，函数。关闭串口使用 CloseHandle 函数。

下面是对关键函数的记录：

CreateFile()创建或者打开一个文件或者 I/O 设备，通常使用的 I/O 形式有文件、文件流、目录、物理磁盘、卷、终端流等。如执行成功，则返回文件句柄。它其中有个参数 lpFileName 是用来要打开的串口逻辑名，用字符串来表示，如 COM1， COM2，分别表示串口 1，和串口 2。如果要确定工控机上有那些串口，要使用设备管理器查看。

CloseHandle()函数很简单，它的参数是使用 CreateFile 打开的端口句柄。调用这个函数可以实现串口关闭。

ReadFile()从文件指针指向的位置（设备文件，通信）开始将数据读出到一个文件中，且支持同步和异步操作，如果文件打开方式没有指明 FILE_FLAG_OVERLAPPED 的话，当程序调用成功时，它将实际读出文件的字节数保存到 lpNumberOfBytesRead 指明的地址空间中。FILE_FLAG_OVERLAPPED 允许对文件进行重叠操作。

WriteFile()将数据写入一个文件（设备文件，通信）。该函数比 fwrite 函数要灵活的多。也可将这个函数应用于对通信设备、管道、套接字以及邮槽的处理。返回时，TRUE（非零）表示成功，否则返回零。会设置 GetLastError。

DAY7

昨天在实现的时候遇到了 bug，通过上网查阅资料都没有找到解决方法，故拖到了今天，在向指导老师询问后，得到了解决。下面记录一下解决的过程。

Createfile()函数打开串口失败，这个问题我卡了好久，就短短几行代码，定义的 handle 的值始终为 0xffffffff，等于 INVALID_HANDLE_VALUE，打开串口失败

```
if (hcom == INVALID_HANDLE_VALUE)
                {
                    DWORD err = GetLastError();
```

```
                printf("err:%d\n", err);
                //LeaveCriticalSection(&m_csCommunicationSync);
                //return false;
            }
```

通过打印 CreateFile 返回错误值，发现是 5，返回 5 拒绝访问，原因的读写的文件只读属性被设置了，将该属性去掉文件读取就正确了或者或者将参数中关于写的属于全部去掉 GENERIC_READ,FILE_SHARE_READ。

但是，还是不行，怀疑还是没有找到“COM4”的文件，上网查找了还是找不到方法：尝试了 CreatefileA()函数等等均没有解决问题，于是打算用 Open()函数来进行对串口的访问。Open()是 SerialPort 串口操作类的函数，串口进行操作的类，其中包括写和读操作，类可设置串口参数、设置接收函数、打开串口资源、关闭串口资源,操作完成后,一定要关闭串口、接收串口数据事件、接收数据出错事件、获取当前全部串口、把字节型转换成十六进制字符串等功能。

DAY8

今天，指导老师和我说代码规范，很委婉的表达了我代码缺乏规范的问题，他向我展示了他所写的代码，并向我指明了所需要注意的代码规范，如注释，因为我写的代码注释比较少，每个模块都缺少代码说明，代码的可读性较差，他建议我使用良好的编码样式和清晰的命名来减少注释，对模块、函数、变量、数据结构、算法和关键代码做注释，应重视注释的质量而不是简单的最求数量。

今天在运行过程中，遇到了一个问题 Detected memory leaks!

Detected memory leaks!

Dumping objects ->

{9877} normal block at 0x05785AD0, 152 bytes long.

Data: << N N x 7 > 3C AC 4E 10 00 00 00 00 BC A4 4E 10 78 B6 37 00

Object dump complete.

找到了泄漏的那块内存分配的底层操作的地方，利用 "调试" –> "退出" ,快捷键为:"Shift + F11" .跳出当前函数..然后一直往"上"跳,边跳边查看调用栈.直到看到自己写的代码，可以发现自己分配了内存,但是没有合理释放.处理后再次运行就没有出现内存泄漏了。

DAY9

今日主要完成了串口接收数据

起初一筹莫展，在和实习指导老师进行沟通后，找到了实现方法——状态机。

图像帧格式被定义为：从左到右依次：‘GRAY’（4 字节）+图像数据长度（2 字节）+预留（4 字节）+图像数据+图像数据累加和（short 累加）

一共设置了 10 个状态，分别如下：初始状态 0，若接收字符不为‘G’，则一直为 0 状态，反之跳转到 1 状态。若下一个字符接收到的是‘R’，则跳转到 2 状态，反之重新回到 0 状态，如此进行判断，最后得到所需要的数据。同时，也加入了累加和来进行数据正确性的校验。

原先打算使用循环冗余校验（Cyclic Redundancy Check， CRC）来实现。CRC 是一种根据网络数据包或计算机文件等数据产生简短固定位数校验码的一种信道编码技术，主要

用来检测或校验数据传输或者保存后可能出现的错误。它是利用除法及余数的原理来作错误侦测的。但是，太难了，就采用了累加和来进行简单的判断。

DAY10

今日完成了对接收到数据的显示，主要算法流程为：

按下“打开串口”按键，调用 AfxBeginThread()函数（除了前面两个参数外，后面的都是默认参数，可以省略。而必须有的这两个参数，一个是线程函数的指针，一个是传递给这个函数的参数），结束线程的两种方式 1：这是最简单的方式,也就是让线程函数执行完成,此时线程正常结束.它会返回一个值,一般 0 是成功结束, 2：如果想让别一个线程 B 来结束线程 A,那么,就需要在这两个线程中传递信息.不管是工作者线程还是界面线程,如果想在线程结束后得到它的确结果,那么你可以调用::GetExitCodeThread 函数，这里是选用方法一。

线程开始 DataReceivingThread()函数，使用昨天所完成的串口接收数据函数 ReceiveSerialData()，接收成功返回 1，将接收的数据存入定义好的 ReceiveQueue 结构体中，然后使用前天完成的 SetRawData()函数完成 BMP 格式的转换。转换完成后调用 OnPaint()函数完成显示工作，其中会调用 SetWnd()、ViewBmp()、BufferToHBITMAP()

BufferToBITMAP 函数主要完成对 BMP 格式的封装：如下代码

```
hDIB = lpBuffer + bmfHeaderLen;
    BITMAPINFOHEADER &bmiHeader = *(LPBITMAPINFOHEADER)hDIB ;
    BITMAPINFO &bmInfo = *(LPBITMAPINFO)hDIB ;
    lpDIBBits=(lpBuffer)+((BITMAPFILEHEADER *)lpBuffer)->bfOffBits;
    CClientDC dc(m_pWnd);
    hShowBMP = CreateDIBitmap(dc.m_hDC,&bmiHeader,CBM_INIT,lpDIBBits,
                              &bmInfo,DIB_RGB_COLORS);
```

总算实现了显示工作，感觉太不容易了

DAY11

今天完成其他功能：首先是清空显示，这个比较简单，调用 SetWindowText()函数将 windowtext 设置为空;

接下来是缩放功能，在软件界面可利用鼠标进行图形的缩放功能，关键是它的缩放还根据鼠标位置的不同而实现不同的缩放功能。具体关键细节有如下情况：

1、滚轮滚动有两个方向，一般一个方向用于放大、另一个用于缩小，但具体哪个方向是放大哪个缩小，其实都无所谓，不同软件都不太一样。

2、一般鼠标停留的在哪个位置进行滚动滚轮时，给人的感觉是整个软件界面以鼠标点为中心进行缩放的，也就是图形或图像上鼠标的点不动，其余位置成比例放大缩小。不管鼠标是否落在图像内或在图像外均体现出该功能。因为对 BMP 还是不大了解，重新学习了一下，同时经过上网查阅，发现实现形式很简单：在类向导中添加鼠标滚轮消息 WM_MOUSEWHELL 的消息响应函数：OnMouseWheel(UINT nFlags, short zDelta, CPoint pt)。说明：这里的参数 zDelta 代表滚轮滚动的方向，＞0 为正方形、＜0 为负方向；而 pt 则表示鼠标的方向，不过这里不能用 pt ,因为它不是鼠标在窗口中的坐标而是鼠标在整个显示器内的坐标。

```
m_BMP.m_nX -= m_BMP.m_nWeight*0.2/2;
```

```
m_BMP.m_nY -= m_BMP.m_nHeight*0.2/2;
m_BMP.m_nWeight += (int)m_BMP.m_nWeight*0.2;
m_BMP.m_nHeight += (int)m_BMP.m_nHeight*0.2;
Invalidate(false);
```

缩小同理，此处设定的为每次滚轮滚动，缩放大小为 BMP 位图大小的 0.1 倍。

DAY12

今天完成剩下的功能：首先是文件保存功能，这个也比较简单，上网搜索一下，可以找到很多的教程，CFileDialog 类封装了 Windows 常用的文件对话框。常用的文件对话框提供了一种简单的与 Windows 标准相一致的文件打开和文件存盘对话框功能，所以先用 CFileDialog 构造函数构造一个对象

```
if(dlg.DoModal() == IDOK)
    {
        strFileName = dlg.GetPathName();
        if(!file.Open(strFileName,CFile::modeCreate|CFile::modeWrite))
        {
            AfxMessageBox("Create Dat File Fail.");
            return;
        }
        file.Write(m_BMP.m_pBmpBuf, m_BMP.BmpBufLen);
        file.Close();
        sprintf(g_sendData, "\r\nSave " + strFileName + " success;\r\n");
        ::SendMessage(hWnd, WM_EDITSHOW, 0, 1);
    }
```

基本过程：首先调用 OnFileSave()，OnFileSave 调用 DoFileSave()，DoFileSave()判断文件是否只读，是则 DoSave(NULL)，否则调用 DoSave(m_strPathName)调用 CDocument::OnSaveDocument()保存文档。若 bReplace==TRUE 则 SetPathName(newName)覆盖当前路径名。OnSaveDocument()打开文件，创建 CArchive 对象 saveArchive，Serialize(saveArchive)读写文件，SetModifiedFlag(FALSE)。

打开文件功能的实现同理，就不在此记录了。

保存设置功能的实现也挺简单的，仅仅就是获取框中 MinA、MinB、MaxA、MaxB、StepA StepB 的值进行保存。本来是打算把得到的值化为整型数进行保存，但是发现再要显示的话，还得重新转回 string 格式，故删掉了转为整数型的操作。

综上所述，今天任务比较简单。

DAY13

今天给指导老师展示了一下效果，发现了笔在纸上书写时，采集到的图片在屏幕上进行显示时出现一卡一卡的现象，于是今天为之进行改进。

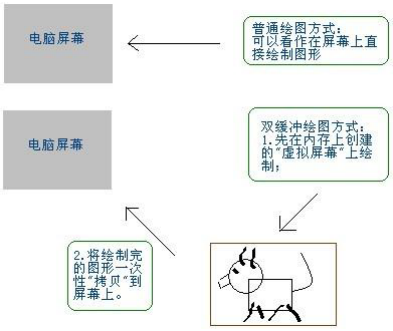
查阅网上博客：得到了几个解决方法：首先当然是去掉 MFC 提供的背景绘制过程了。

- 可以在窗口形成时给窗口的注册类的背景刷付 NULL
- 也可以在形成以后修改背景

```
static CBrush brush(RGB(255,0,0));
SetClassLong(this->m_hWnd,GCL_HBRBACKGROUND,(LONG)(HBRUSH)brush);
```

- 简单也可以重载 OnEraseBkgnd(CDC* pDC)直接返回 TRUE，这样背景没有了，

结果图形显示的确不闪了，但是显示也象前面所说的一样，变得一团乱。接下来继续查找方法，得到了解决方法——双缓冲原理：



原理基本上是这样的：可以把电脑屏幕看作一块黑板。然后我们在内存环境中建立一个“虚拟“的黑板，然后在这块黑板上绘制复杂的图形，等图形全部绘制完毕的时候，再一次性的把内存中绘制好的图形“拷贝”到另一块黑板（屏幕）上。采取这种方法可以提高绘图速度，极大的改善绘图效果。

DAY14

今天是实习的倒数第二天，书写实验报告。

由于之前 COM 口都是写死的，故今天来进行一下改进，从系统打开注册表获取到 COM 口在控件上进行显示。查询注册表的方法是比较常见的方法，通过查看“ HKEY_LOCAL_MACHINE\HARDWARE\DEVICEMAP\SERIALCOMM ”项来获取串口信息。方法就是使用编程方法读取注册表内信息，相当于用户通过在运行框内输入 ”regedit” （或 regedit32 ）直接打开注册表。临时定义 256 个字符串组，因为系统最多也就 256 个，使用 RegOpenKeyEx()打开一个制定的注册表键,成功返回 ERROR_SUCCESS 即“0”值，进入 if 判断语句，读取键值，如果键值等于 ERROR_SUCCESS 或者 ERROR_MORE_DATA，每读取一次 dwName 和 dwSizeofPortName 都会被修改，一定要重置,否则会出现很离奇的错误,就试过因没有重置,出现先插入串口号大的（如 COM4）,再插入串口号小的（如 COM3），此时虽能发现两个串口，但都是同一串口号（COM4）的问题。

DAY15

今天是实习最后一天。今天发现用 CREATEFILE 打开 1-9 的串口没问题,但是打开 COM10 之类的串口号，返回的 m_hComm 就为 INVALID_HANDLE_VALUE，上网查阅了一下资料，CreateFile() 可获得到串行端口的句柄。 "Win 32 编程人员参考"项"CreateFile() 提及的共享模式必须是 0 中，创建参数必须是 OPEN_EXISTING，并且该模板必须为 NULL，不能直接简单粗暴的用“COM10”、“COM11”指定串行端口，那么如果需要打开 COM10 以上的串口，就需要把第一个参数从原来的“COM10”改为“\\\\.\\COM10”，就能打开成功了。原文如下：

CreateFile("\\\\.\\COM10", // address of name of the

```
communications     device communications     device
                   fdwAccess, //     access     (read-write)     mode
                   0,     //     share     mode
                   NULL, //     address     of     security     descriptor
                   OPEN_EXISTING, //     how     to     create
                   0, //     file     attributes
                   NULL   //     handle     of     file     with     attributes     to     copy
              );
```

我试验了一下，如果打开 COM1 的话，也可以使用“\\\\.\\COM1”。所以可以把传入的参数变为"\\\\.\\+Port"。

所以在代码中进行增设：

```
if (atoi(strCommName.Mid(3, strCommName.GetLength()-3)) >= 10)
    {
        strCommName = _T("\\\\.\\") + strCommName;
    }
```

今天任务还是挺简单的，于是开始着手实验报告的书写。