```
procedure MappingDemo_MemoryStream64;
var
  data: TMemoryStream64;
  m64: TMemoryStream64;
begin
  // TMemoryStream64提供了内存映射方法
  // 使用内存映射可以避免两个Stream的反复copy
  // 另外，内存映射还可以直接对一个内存块使用Stream方法操作
  // 换句话说，TStringList的LoadFromStream+SaveToStream方法，通过TMemoryStream64中转，可以高速操作内存块
  data := TMemoryStream64.Create;
  data.Size := 1024 * 1024 * 1024;

  m64 := TMemoryStream64.Create;

  // 将data的申明的内存块直接映射到m64中，这种方法没有copy，非常适合大内存块交换
  // 使用SetPointerWithProtectedMode方法映射后，Position会被置0
  m64.SetPointerWithProtectedMode(data.Memory, data.Size);

  // 现在，我们可以使用任意TStream的方法来操作内存块，这是高速内存映射

  // 释放时养成一个好习惯，先释放使用了内存映射的类，再释放宿主
  DisposeObject([m64, data]);
end;

procedure MappingDemo_MemoryRaster;
var
  data: TMemoryRaster;
  mr: TMemoryRaster;
begin
  // TMemoryRaster也提供了类似的内存映射方法原理和TMemoryStream64相同
  data := newRaster();
  data.SetSize(10000, 10000, RasterColorF(0, 0, 0));

  mr := newRaster();
  // 将data的光栅直接映射到mr中，这种方法没有copy，非常适合大光栅化的处理
  // 使用SetWorkMemory方法映射后，mr的width,height,bits都来自data
  mr.SetWorkMemory(data);

  // 现在，我们可以使用任意TMemoryRaster的方法来操作，这是高速内存映射

  // 释放时养成一个好习惯，先释放使用了内存映射的类，再释放宿主
  DisposeObject([mr, data]);
end;

```