第12章 视窗管理器（WM）

使用μC/GUI 管理器（WM）时，在显示屏上显示的的所有内容包括在一个窗口里面——屏幕上的一块区域，该区域作为一个绘制或显示对象的用户接口部件。窗口可以是任意大小，你可能在屏幕上同时显示多个窗口，甚至在其它窗口的上面部分或完全地显示。

视窗管理器提供了一套函数，使你能很容易地对许多窗口进行创建，移动，调整大小及其它操作。它也可以提供更低层的支持，这通过管理显示屏上的窗口的层，及通过给你的应用程序发送信号以显示影响它的窗口的修改来完成。

μC/GUI 的视窗管理器是一个独立的（可选的）的软件项目，它没有包括进μC/GUI 基本软件包里。视窗管理器的软件位于子目录“GUI\WM”下。

12.1 术语解释

窗口外形是矩形，由它们的原点（左上角的 X 和 Y 坐标）及它们的 X 和 Y 尺寸（分别是宽和高）所定义。μC/GUI 中一个窗口：

- 是一个矩形
- 有一个 Z 坐标
- 可能是隐藏的或可见的
- 可能拥有有效/或无效区域
- 可以或者不可以有透明区域
- 可以或者不可以有一个回调函数

活动窗口

当前正在使用进行绘图操作的窗口被当作活动窗口。与最顶层窗口一样，它不是必需的。

回调函数

回调函数在用户程序中定义，当一个指定的事件发生时，通知图形系统调用指定的函数。通常应用于一个窗口内容改变时自动重绘的场合。

子/父窗口，同胞

一个子窗口的定义是相对于另一个窗口，该窗口称为父窗口。无论什么时候，一个父窗口移动了，它的子窗口会相应随之移动。一个子窗口总是完全包含在它的父窗口里面，如果需要，它会被剪切。从属于同一个父窗口的多个子窗口相互间的关系称为“同胞”。

客户区

一个窗口的客户区简单地说是它的可使用区。如果一个窗口包括一个边框或标题栏，则客户区是内部的矩形区域。如果没有这样一个边框，则客户区等同于窗口本身。

剪切，剪切区域

剪切是一种限制窗口或它的部分输出的操作。

剪切区域是一个窗口的原有可见区域。由于被更高 Z 序列的同胞窗口遮挡，或者不在父

窗口可见区域范围之内的缘故，这些部分就会被剪切掉。

桌面窗口

桌面窗口由视窗管理器自动创建，总是覆盖整个显示区域。它始终是一个最底层的窗口。如果没有定义其它窗口，它就是默认（活动）窗口。所有窗口都是桌面窗口的继承窗口。

句柄

当一个新的窗口被创建，WM 会给它分配一个唯一的标识符，称为句柄。句柄将用于对特定窗口更进一步操作的执行。

隐藏/显示窗口

一个隐藏的窗口是不可见的，尽管它仍然存在（有句柄）。当创建一个窗口时，如果没有创建指定的标识的话，默认状态是隐藏。使一个窗口可见则可以将其显示；使其不可见则可以隐藏它。

透明

带有透明部分的窗口包括有窗口静止时不被重绘的区域。这些区域的操作就仿佛是下面的窗口可以透过它们显示出来。在这种情况下，下面的窗口在这个透明窗口之前重绘就显得很重要了。WM 能自动处理正确的重绘顺序。

有效/无效

一个有效的窗口是一个完全更新了的窗口，它不需要重绘。

一个无效的窗口不再对所有的更新有反应，因此需要完全或部分重绘。当改变影响一个特定的窗口时，WM 标记该窗口无效。下一次窗口重绘（手动或通过回调函数）后，它将有效。

Z-序，底层/顶层

尽管一个窗口是在一个只有 X 和 Y 坐标构成的二维的屏幕上显示，WM 也管理被认为是 Z-序，或者深度的座标——一个虚拟的三维的坐标，决定窗口从背景到前景放置。因此窗口可以在另一窗口的上面或下面显示。

将一个窗口设置到底层将会把它放在所有同胞窗口（如果存在）的下面；设置为顶层将会把它放在所有同胞窗口的上面。创建一个窗口时，如果没有指定创建标识符，默认情况下

它会被设置在顶层。

12.2 WM API 函数

下表列出了与 μC/GUI 视察管理器有关的函数，在各自的类型中按字母顺序进行排列。函数的详细描述在本章稍后列出。

函数	说明
基本函数	
WM_CreateWindow()	创建一个窗口。
WM_CreateWindowAsChild()	创建一个子窗口。
WM_DeleteWindow()	删除一个窗口。
WM_Exec()	通过执行回调函数（所有工作）重绘有效窗口。
WM_Exec1()	通过执行一个回调函数（仅一个工作）重绘有效窗口。
WM_GetClientRect()	返回活动窗口的。
WM_GetDialogItem()	返回一个对话框项目（控件）的窗口句柄。
WM_GetOrgX()	返回活动窗口的原点 X 坐标。
WM_GetOrgY()	返回活动窗口的原点 Y 坐标。
WM_GetWindowOrgX()	返回一个窗口的原点 X 坐标。
WM_GetWindowOrgY()	返回一个窗口的原点 Y 坐标。
WM_GetWindowRect()	返回活动窗口的屏幕坐标。
WM_GetWindowSizeX()	返回一个窗口的水平尺寸（宽度）。
WM_GetWindowSizeY()	返回一个窗口的垂直尺寸（高度）。
WM_HideWindow()	使一个窗口不可见。
WM_InvalidateArea()	使显示屏的某些部分无效。
WM_InvalidateRect()	使一个窗口部分无效。
WM_InvalidateWindow()	使一个窗口无效。
WM_MoveTo()	设置一个窗口的坐标。
WM_MoveWindow()	移动一个窗口到另一个位置。
WM_Paint()	立即绘制或重绘一个窗口。
WM_ResizeWindow()	改变一个窗口尺寸。
WM_SelectWindow()	设置用于绘图操作的活动窗口。
WM_ShowWindow()	使一个窗口可见。
高级特性	
WM_Activate()	激活视窗管理器。
WM_BringToBottom()	在其同胞窗口之后放置一个窗口。
WM_BringToTop()	在其同胞窗口之前放置一个窗口。
WM_ClrHasTrans()	清除 has 透明标识。
WM_Deactivate()	解除视窗管理器。
WM_DefaultProc()	处理信息的默认函数。

WM_GetActiveWindow()	返回活动窗口的句柄。
WM_GetDesktopWindow()	返回桌面窗口的句柄。
WM_GetFirstChild()	返回窗口的第一个子窗口的句柄。
WM_GetNextSibling()	返回窗口的下一相同胞窗口的句柄。
WM_GetHasTrans()	返回 has 透明标志的当前值。
WM_GetParent()	返回窗口的父窗口的句柄。
WM_Init()	初始化视窗管理器。不再需要由 GUI_Init()来完成。
WM_IsWindow()	判断一个指定的句柄是否一个有效句柄。
WM_SendMessage()	向一个窗口发送信息。
WM_SetDesktopColor()	设置桌面窗口颜色。
WM_SetCallback()	为一个窗口设置回调函数。
WM_SetCreateFlags()	当创建一个新窗口时设置一个默认标识符。
WM_SetHasTrans()	设置 has 透明标志。
WM_SetUserClipRect()	临时减小剪切区域。
WM_ValidateRect()	使一个窗口的部分有效。
WM_ValidateWindow()	使一个窗口有效。
存储设备支持（可选）	
WM_DisableMemdev()	禁止用于重绘一个窗口的存储设备的使用。
WM_EnableMemdev()	启用用于重绘一个窗口的存储设备的使用。

12.3 视窗管理器的回调机制

WM 可以在有或没有回调函数的情况下使用。在大多数情况下，使用回调函数更为可取。

回调机制后面的哲学

µC/GUI 为窗口和窗口对象（控件）提供的回调机制实质是一个事件驱动系统。正如在大多数视窗系统中一样，原则是控制流程不只是从用户程序到图形系统（用户程序调用图形系统函数来更新窗口），而且可以从用户程序到图形系统，同时也从图形系统回到用户程序，意思是图形系统也可以调用用户程序提供的回调函数来达到更新窗口的目的。这种机制——常常表现好莱坞法则的特点（“不要打电话给我们，我们会打电话给你们!”）——主要是视窗管理器为了启动窗口重绘的需要。与传统程序比较有差异，但它使对视窗管理器的无效逻辑开发成为可能。

不使用回调函数

你不一定非要用回调函数不可，但这样做，WM 在重绘窗口管理时会降低效率。也可以混合使用，例如，一些窗口使用回调而另一些却不使用。然而，如果一个窗口不使用回调机制，

你的应用程序必须负责更新窗口内容。

警告：当没有使用回调机制时，屏幕更新的管理就成了你的责任。

12.4 使用回调函数

为了使用一个回调函数创建窗口，你必需要有一个回调函数。函数的名称将与创建窗口时对应的回调函数指针参数名称相一致（即 WM_CreateWindow() 中的 cb 参数）。所有的回调函数必须具有以下函数原型：

函数原型

```
void callback(WM_MESSAGE* pMsg);
```

参　数	含　意
pMsg	消息的指针。

回调函数的执行行为依赖于它收到的消息类型。上面的函数原型通常带有一个开关声明，用于定义了对于不同的使用一个或更多的事件声明的消息所采用的不同的处理方式（典型的至少有对 WM_PAINT()的处理）。

范例

创建一个回调函数自动重绘一个窗口：

```
void WinHandler(WM_MESSAGE* pMsg)
{
    switch (pMsg->MsgId)
    {
        case WM_PAINT:
            GUI_SetBkColor(0xFF00);
            GUI_Clear();
            GUI_DispStringAt("Hello world",0,0);
            break;
    }
}
```

WM_MESSAGE 元素

MsgId	消息的类型（参照下表）
HWin	目标窗口。
hWinSrc	源窗口。
Data.p	数据指针。
Data.v	数据数值。

MsgId 元素使用的消息类型

WM_PAINT	重绘窗口（因为内容至少部分无效）。
WM_CREATE	一个窗口创建后产即发送。
WM_DELETE	告诉窗口释放它的数据结构（如果有的话），然后它将会被删除。
WM_SIZE	当一个窗口的大小改变后发送到它。
WM_MOVE	当一个窗口移动后发送到它。
WM_SHOW	当一个窗口收到显示命令后发送到它。
WM_HIDE	当一个窗口收到隐藏命令后发送到它。
WM_TOUCH	触摸屏消息。

应用程序可以为它自己的用途定义附加消息。为了保证它们使用的消息 ID 不会与μC/GUI 使用的消息 ID 同名，用户定义的消息的编号以 WM_USER 为开始。你应该像下面所展示的一样定义你自己的消息：

```
#define MY_MESSAGE_AAA  WM_USER+0
#define MY_MESSAGE_BBB  WM_USER+1
```

等等

背景窗口重绘及回调

在初始化视窗管理器期间，会创建一个包括整个 LCD 区域的窗口作为背景窗口（或称桌面窗口）。该窗口的句柄是 WM_HBKWIN。WM 不会自动重绘背景窗口的区域，因为没有默认的背景颜色。这意味着如果你进一步创建新一层窗口，然后删除它，被删除的窗口仍然是可见的。需要指定 WM_SetBkWindowColor() 函数设置重绘背景窗口的颜色。

你也可以设置一个回调函数处理这个问题。如果一个窗口被创建，然后象前面一样被删除，回调函数将触发 WM 去识别背景窗口已不再有效，并自动进行重绘。想了解关于创建和使用回调函数的信息，参考本章末的范例。

12.5 基本函数

WM_CreateWindow()

描述

在一个指定位置创建一个指定尺寸的窗口

函数原型

```
WM_HWIN WM_CreateWindow (   int x0, int y0,
                            int width, int height,
                            U8 Style,
                            WM_CALLBACK* cb,
                            int NumExtraBytes);
```

参　数	含　意
x0	左上角 X 轴坐标。
y0	左上角 Y 轴坐标。
width	窗口的 X 轴尺寸。
height	窗口的 Y 轴尺寸。
Style	窗口创建标识，如下一表格所示。
cb	回调函数的指针，如果没有使用回调函数则为 NULL。
NumExtraBytes	分配的额外字节数，通常为 0。

参数 Style 允许数值（或数值组合）

WM_CF_HASTRANS	Has 透明标志。必须由窗口定义哪些客户区不能完全填充。
WM_CF_HIDE	创建窗口后将它隐藏（默认）。
WM_CF_SHOW	创建窗口后显示它。
WM_CF_FGND	创建窗口后把它放到前面。（默认）。
WM_CF_BGND	创建窗口后把它放到后面。
WM_CF_STAYONTOP	确定窗口继续停留在所有创建的不带这个标志的同胞窗口上面。
WM_CF_MEMDEV	重绘时自动使用一个存储设备。这可能避免闪烁，同时在大多数情况下会提高输出速度，因为剪切简化了。注意为了能够使用这个标志，要求用到存储设备软件包（并且要在配置中启用）。如果存储设备没有启用，该标志被忽略。

返回数值

所创建窗口的句柄。

附加信息

几个创建标志可以通过“OR”操作进行组合。使用负的座标系。

范例

用回调函数创建一个窗口：

```
hWin2 = WM_CreateWindow(100, 10 ,180, 100, WM_CF_SHOW, &WinHandler, 0);
```

用回调函数创建一个窗口：

```
hWin2 = WM_CreateWindow(100, 10 ,180, 100,WM_CF_SHOW, NULL, 0);
```

WM_CreateWindowAsChild()

描述

以子窗口的形式创建一个窗口。

函数原型

```
WM_HWIN WM_CreateWindowAsChild( int x0, int y0,
                                int width, int height,
                                WM_HWIN hWinParent,
                                U8 Style,
                                WM_CALLBACK* cb,
                                int NumExtraBytes);
```

参数	含义
x0	相对于父窗口的左上角 X 轴坐标。
y0	相对于父窗口的左上角 Y 轴坐标。
width	窗口的 X 轴尺寸。如果为 0，为父窗口客户区 X 轴尺寸。
height	窗口的 Y 轴尺寸。如果为 0，为父窗口客户区 Y 轴尺寸。
hWinParent	父窗口的句柄。
Style	窗口创建标志（参阅 WM_CreateWindow ()）。
cb	回调函数的指针，没有使用回调函数的话，为 NULL。
NumExtraBytes	额外分配的字节数量，通常为 0。

返回数值

子窗口的句柄。

附加信息

如果 hWinParent 参数设为 0，背景窗口当作父窗口使用。默认情况下，一个子窗口会旋转在它的父客串和所有原先的同胞窗口的上面，所以，如果它们的 Z-序没有改变的话，这个“最年轻”的窗口将总会在最顶部。

同胞窗口的 Z-序可以改变，尽管不论它们的次序如何，它们总是保持在它们父窗口的上面。

WM_DeleteWindow()

描述

删除一个指定的窗口。

函数原型

```
void WM_DeleteWindow(WM_HWIN hWin);
```

参数	含义
hWin	窗口句柄。

附加信息

在窗口被删除之前，它收到一个 WM_DELETE 消息。该消息典型用于删除任何它使用的对象（控件）并释放分配给窗口的动态内存。

如果指定的窗口有子窗口，在窗口本身被删除之前，这些子窗口自动被删除。

WM_Exec()

描述

通过执行回调函数（所有工作）重绘无效窗口。

函数原型

```
int WM_Exec(void);
```

返回数值

0：如果没有工作执行；1：如果执行一项工作。

附加信息

该函数将自动重复调用 WM_Exec1()，直到它完成所有的工作——本质上是直到返回一个 0 数值。

推荐改为调用 GUI_Exec()。

通常该函数不需要由用户应用程序调用。它由 GUI_Delay ()函数自动调用。如果你在使用多任务系统，我们推荐使用一个单独的任务执行这个函数，如下所示：

```
void ExecIdleTask(void)
{
    while(1)
    {
        WM_Exec();
    }
}
```

WM_Exec1()

描述

通过执行一个回调函数（只有一项工作）重绘一个无效窗口。

函数原型

```
int WM_Exec1(void) ;
```

返回数值

0：如果没有工作执行；1：如果执行一项工作。

附加信息

该函数可能反复调用直到返回 0，这意思是所有的工作已经完成。

推荐改为调用 GUI_Exec1()。

该函数由 WM_Exec()函数自动调用。

WM_GetClientRect()

描述

返回活动窗口的客户区的座标。

函数原型

```
void WM_GetClientRect(GUI_RECT* pRect);
```

参数	含义
pRect	一个 GUI_RECT 的指针。

WM_GetDialogItem()

描述

返回一个对话话项目（控件）的窗口句柄。

函数原型

```
WM_HWIN WM_GetDialogItem(WM_HWIN hDialog, int Id);
```

参数	含义
hDialog	对话框的句柄。
Id	控件的窗口 ID。

返回数值

控件窗口的句柄。

附加信息

该函数总是在创建对话框时使用。在使用它之前，对话框使用的窗口 ID 必须转化成它的句柄。

WM_GetOrgX()，WM_GetOrgY()

描述

（分别）返回一个活动窗口的客户区的原点的 X 轴或 Y 轴坐标。

函数原型

```
int WM_GetOrgX(void); int WM_GetOrgY(void);
```

返回数值

以像素为单位的客户区原点的 X 轴或 Y 轴坐标。

WM_GetWindowOrgX()，WM_GetWindowOrgY()

描述

（分别）返回以像素为单位的指定窗口的客户区的原点的 X 轴或 Y 轴坐标。

函数原型

```
int WM_GetWindowOrgX(WM_HWIN hWin); int WM_GetWindowOrgY(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

以像素为单位的客户区的原点的 X 轴或 Y 轴坐标。

WM_GetWindowRect()

描述

返回活动窗口的座标。

函数原型

```
void WM_GetWindowRect(GUI_RECT* pRect);
```

参数	含义
pRect	一个 GUI_RECT 结构的指针。

WM_GetWindowSizeX()，WM_GetWindowSizeY()

描述

（分别）返回指定窗口的 X 轴或 Y 轴的尺寸。

函数原型 s

```
int WM_GetWindowSizeX(WM_HWIN hWin); int WM_GetWindowSizeY(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

以像素为单位的窗口 X 轴或 Y 轴的尺寸。

在水平方向定义为 x1-x0+1，垂直方向为 y1-y0+1，这里 x0，x1，y0，y1 是分别是窗口的最左边，最右边，最顶部，最底部坐标。

WM_HideWindow()

描述

使一个指定窗口不可见。

函数原型

```
void WM_HideWindow(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

调用该函数后，窗口并不会立即显现“不可见”效果。其它窗口的无效区域（那些位于窗口后面的应该隐藏的区域）在执行 WM_Exec ()函数时会被重绘。如果你需要立即隐藏一个窗口的话，你应当调用 WM_Paint ()函数重绘其它窗口。

WM_InvalidateArea()

描述

使显示屏的指定矩形区域无效。

函数原型

```
void WM_InvalidateArea(GUI_RECT* pRect);
```

参数	含义
pRect	一个 GUI_RECT 结构的指针。

附加信息

调用该函数将告诉 WM 指定的区域不要更新。

该函数可以用于使任何窗口或重叠可相交的部分窗口无效。

WM_InvalidateRect()

描述

使一个窗口的指定矩形区域无效。

函数原型

```
void WM_InvalidateRect(WM_HWIN hWin, GUI_RECT* pRect);
```

参数	含义
hWin	窗口的句柄。
pRect	一个 GUI_RECT 结构的指针。

附加信息

调用该函数将告诉 WM 指定的区域不要更新。

接下来调用 WM_Paint ()进行窗口重绘，该区域同样也能够被重绘。

WM_InvalidateWindow()

描述

使一个指定的窗口无效。

函数原型

```
void WM_InvalidateWindow(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

调用该函数告诉 WM 指定的窗口不更新。

WM_MoveTo()

描述

将一个指定的窗口移到某个位置。

函数原型

```
void WM_MoveTo(WM_HWIN hWin, int dx, int dy);
```

参数	含义
hWin	窗口的句柄。
x	新的 X 轴坐标。
y	新的 Y 轴坐标。

WM_MoveWindow()

描述

把一个指定的窗口移动一段距离。

函数原型

```
void WM_MoveWindow(WM_HWIN hWin, int dx, int dy);
```

参数	含义
hWin	窗口的句柄。
dx	移动的水平距离。
dy	移动的垂直距离。

WM_Paint()

描述

立即绘制或重绘一个指定窗口。

函数原型

```
void WM_Paint(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

窗口重绘，反映它最后一次绘制以来所有的更新。

WM_ResizeWindow()

描述

改变一个指定窗口的尺寸。

函数原型

```
void WM_ResizeWindow(WM_HWIN hWin, int XSize, int YSize);
```

参数	含义
hWin	窗口的句柄。
Xsize	窗口水平尺寸要修改的值。
YSize	窗口垂直尺寸要修改的值。

WM_SelectWindow()

描述

设置一个活动窗口用于绘制操作。

函数原型

```
WM_HWIN WM_SelectWindow(WM_HWIN hWin);_
```

参数	含义
hWin	窗口的句柄。

返回数值

选择的窗口。

范例

将活动窗口的句柄设为 hWin2，设置背景颜色，然后清除窗口：

```
WM_SelectWindow(hWin2);
GUI_SetBkColor(0xFF00);
GUI_Clear();
```

WM_ShowWindow()

描述

使一个指定窗口可见。

函数原型

```
void WM_ShowWindow(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

该调用以后，窗口不会立即可见。它在执行 WM_Exec ()更新。如果你需要立即显示（绘

制）这个窗口，你应当调用 WM_Paint ()函数。

12.6 高级函数

WM_Activate()

描述

激活视窗管理器。

函数原型

```
void WM_Activate (void) ;
```

附加信息

初始化后，默认情况下 WM 是激活的。该函数只有原先调用 WM_Deactivate ()函数，才需要调用。

WM_BringToBottom()

描述

将一个指定窗口放到它的同胞窗口下面。

函数原型

```
void WM_BringToBottom(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

该窗口会放在所有其它同胞窗口的下面，但是位置保持在它的父窗口前面。

WM_BringToTop()

描述

将一个指定窗口放到它的同胞窗口上面。

函数原型

```
void WM_BringToTop(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

该窗口会放在所有其它同胞窗口和父窗口的上面。

WM_ClrHasTrans()

描述

清除 has 透明标志（设为 0）。

函数原型

```
void WM_ClrHasTrans(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

当该标志被设置时，它告诉视窗口管理器一个窗口包含有不要重绘的部分，该部分因此而透明。然后 WM 知道背景重绘需要优先于窗口重绘，这是为了保证透明部分能正确地恢复。

该标志被 WM_ClrHasTrans ()清除后，在窗口重绘之前，WM 将不会自动重绘背景。

WM_Deactivate()

描述

使视窗管理器失效。

函数原型

```
void WM_Deactivate(void);
```

附加信息

该函数调用后，剪切区设为全部 LCD 区域，WM 将不执行窗口回调函数。WM_DefaultProc()

描述

WM_DefaultProc()

默认消息处理器。

函数原型

```
void WM_DefaultProc(WM_MESSAGE* pMsg)
```

参数	含义
pMsg	消息的指针。

附加信息

在下面的范例中，使用该函数处理未加工过的消息：

```
static WM_RESULT cbBackgroundWin(WM_MESSAGE* pMsg)
{
    switch (pMsg->MsgId)
    {
        case WM_PAINT:
            GUI_Clear();
        default:
            WM_DefaultProc(pMsg);
    }
}
```

WM_GetActiveWindow()

描述

返回用于绘图操作的活动窗口的句柄。

函数原型

```
WM_HWIN WM_GetActiveWindow(void);
```

返回值

活动窗口的句柄

WM_GetDesktopWindow()

描述

返回桌面窗口的句柄。

函数原型

```
WM_HWIN WM_GetDesktopWindow(void);
```

返回数值

桌面窗口的句柄。

附加信息

桌面窗口总是最底层的窗口，任何进一步创建的窗口都是它的子窗口。

WM_GetFirstChild()

描述

返回指定窗口的第一个子窗口的句柄。

函数原型

```
void WM_GetFirstChild(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

窗口的第一个子窗口的句柄；如果没有子窗口，则为 0 。

附加信息

窗口的第一个子窗口是基于特定的父窗口创建的第一个子窗口。如果窗口的 Z-序列没有改变，它将直接放在指定的父窗口上面。

WM_GetNextSibling()

描述

返回指定窗口的下一个同胞窗口的句柄。

函数原型

```
void WM_GetNextSibling(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

窗口的下一个同胞窗口的句柄，如果不存在，则为 0。

附加信息

窗口的下一个同胞窗口是基于同一个父窗口创建的下一个子窗口。如果窗口的 Z-序列没有改变，它将直接放在指定的窗口上面。

WM_GetHasTrans()

描述

返回 has 透明标志当前的值。

函数原型

```
U8 WM_GetHasTrans(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

0：非透明；1：窗口有透明部分。

附加信息

当该标志被设置时，它告诉视窗口管理器一个窗口包含有不要重绘的部分，该部分因此而透明。然后 WM 知道背景重绘需要优先于窗口重绘，这是为了保证透明部分能正确地恢复。

WM_GetParent()

描述

返回指定窗口的父窗口的句柄。

函数原型

```
void WM_GetParent(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

窗口父窗口的句柄，如果不存在则为 0。

附加信息

没有父窗口存在的唯一情况是桌面窗口的句柄用作参数。

WM_Init()

描述

初始化视窗管理器。

函数原型

```
void WM_Init (void) ;
```

附加信息

不再需要通过用户程序调用该函数，它通过 GUI_Init ()调用。

WM_IsWindow()

描述

确定一个指定句柄是否一个有效的窗口句柄。

函数原型

```
void WM_IsWindow(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

返回数值

0：句柄不是一个有效的窗口句柄男；1：句柄是一个有效的窗口句柄。

WM_SendMessage()

描述

向一个指定窗口发送一个消息。

函数原型

```
void WM_SendMessage(WM_HWIN hWin, WM_MESSAGE* pMsg)
```

参数	含义
hWin	窗口的句柄。
pMsg	消息的指针。

WM_SetDesktopColor()

描述

设置桌面窗口的颜色。

函数原型

```
GUI_COLOR WM_SetDesktopColor(GUI_COLOR Color);
```

返回数值

原先选择的桌面窗口颜色。

附加信息

桌面窗口的默认设置是不需重绘它自己本身。如果该函数没有调用，桌面窗口根本就不会重绘；因此其它窗口会保持可见，甚至它们已经被删除。

一旦这个函数指定一种颜色，桌面窗口将重绘自身。为了恢复默认值，调用这个函数并指定 GUI_INVALID_COLOR。

WM_SetCallback()

描述

设置为视窗管理器执行的回调函数。

函数原型

```
WM_CALLBACK* WM_SetCallback (WM_HWIN hWin, WM_CALLBACK* cb)
```

参数	含义
hWin	窗口的句柄。
cb	回调函数的指针。

返回数值

原先回调函数的指针。

WM_SetCreateFlags()

描述

创建一个新的窗口时设置用作默认值的标志。

函数原型

返回数值

该参数原先的值。

附加信息

这里指定的标志是二进制数 ORed ，和 WM_CreateWindow () 及 WM_CreateWindowAsChild () 函数指定的标志一致。标志 WM_CF_MEMDEV 常常用于在所有窗口上启用存储设备。

范例

```
WM_SetCreateFlags(WM_CF_MEMDEV);      /* 在所有窗口上自动使用存储设备 */
```

WM_SetHasTrans()

描述

设置 has 透明标志（将其设为 1）。

函数原型

```
void WM_SetHasTrans(WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

当设置时，该标志告诉视窗管理器一个窗口包括没有重绘的部分，因而以透明模式显示。然后 WM 知道背景的重绘要先于窗口的重绘，这了为了保证透明部分能正确的重建。

WM_SetUserClipRect()

描述

临时缩小当前窗口的剪切区为一个指定的矩形。

函数原型

```
const GUI_RECT* WM_SetUserClipRect(const GUI_RECT* pRect);
```

参数	含义
pRect	一个定义剪切区域的 GUI_RECT 结构的指针。

返回数值

原先剪切矩形的指针。

附加信息

可以传递一个空指针以恢复默认调协。在回调函数使用时，剪切矩形会被 WM 复位。

给出的矩形必须与当前窗口有关。你不能将剪切矩形面积放大超过当前窗口矩形。

你的应用中必须保证指定的矩形保留它的值直到它不再被使用。也就是，直到一个不同的剪切区被指定或一个空指针被传递。这意思是，如果剪切矩形保持活动直到返回以后，当作参数传递的矩形结构不应是一个自动变量（通常位于堆栈上）。既然这样，应该使用一个静态变量。

范例

这个范例来自进度指示器的绘制函数。进度指示器必须在进度条上写上文字，该处文字颜色必须与进度条左右两部分不同。这意思是一个数字的一半可能是一种颜色，而另一半是另一种颜色。最好的处理办法是在绘制如下所示的进度条进临时减小剪切区：

```
/* 绘制进度条的左边部分 */

r.x0=0; r.x1=x1-1; r.y0=0; r.y1 = GUI_YMAX;
WM_SetUserClipRect(&r);
GUI_SetBkColor(pThis->ColorBar[0]);
GUI_SetColor(pThis->ColorText[0]);
GUI_Clear();
GUI_GotoXY(xText,yText);
GUI_DispDecMin(pThis->v);
GUI_DispChar('%');

/* 绘制进度条的右边部分 */

r.x0=r.x1; r.x1=GUI_XMAX;
```

```
WM_SetUserClipRect(&r);
GUI_SetBkColor(pThis->ColorBar[1]);
GUI_SetColor(pThis->ColorText[1]);
GUI_Clear();
GUI_GotoXY(xText,yText);
GUI_DispDecMin(pThis->v);
GUI_DispChar('%');
```

进度条外观的屏幕截图

WM_ValidateRect()

描述

使一个指定的窗口矩形区域有效。

函数原型

```
void WM_ValidateRect (WM_HWIN hWin, GUI_RECT* pRect);
```

参数	含义
hWin	窗口的句柄。
pRect	一个 GUI_RECT 结构的指针。

附加信息

调用该函数会告诉 WM 指定区域被更新。通常该函数在内部调用，不需由用户应用程序调用。

WM_ValidateWindow()

描述

使一个指定的窗口有效。

函数原型

```
void WM_ValidateWindow (WM_HWIN hWin);
```

参数	含义
hWin	窗口的句柄。

附加信息

调用该函数会告诉 WM 指定窗口被更新。通常该函数在内部调用，不需由用户应用程序调用。

12.7 存储设备支持（可选）

当一个存储设备用于一个窗口的重绘，所有绘制操作自动发送到一个存储设备上下文并在内存中执行。所有绘制操作执行之后的唯一结果是窗口在 LCD 上重绘立即反映所有的更新。使用存储设备的优点是避免了任何闪烁现象（通常发生在绘制操作执行时屏幕不断更新的时候）。

要了解更多关于如何操作存储设备的内容，请参阅第 10 章“存储设备”。

WM_DisableMemdev()

描述

禁止用于重绘一个窗口的存储设备的使用。

函数原型

```
void WM_EnableMemdev (WM_HWIN hWin)
```

参数	含义
hWin	窗口的句柄。

WM_EnableMemdev()

描述

启用用于重绘一个窗口的存储设备的使用。

函数原型

```
void WM_EnableMemdev (WM_HWIN hWin)
```

参数	含义
hWin	窗口的句柄。

12.8 范例

下面的范例展示了使用或不使用回调函数进行重绘之间的区别。它也展示了如何设置你自己的回调函数。范例文件是随μC/GUI 一起发布的范例当中的 WM_Redraw.c：

```
/*----------------------------------------------------------------------
文件:       WM_Redraw.c
目的:       展示视窗管理器的重绘机制
----------------------------------------------------------------------*/

#include "GUI.H"

/*********************************************************************
*                          背景窗口的回调函数                         *
*********************************************************************/

static void cbBackgroundWin(WM_MESSAGE* pMsg)
{
    switch (pMsg->MsgId)
    {
        case WM_PAINT:
            GUI_Clear();
        default:
            WM_DefaultProc(pMsg);
    }
}

/*********************************************************************
*                          前景窗口的回调函数                         *
*********************************************************************/

static void cbForegroundWin(WM_MESSAGE* pMsg)
{
```

```
    switch (pMsg->MsgId)
    {
        case WM_PAINT:
            GUI_SetBkColor(GUI_GREEN);
            GUI_Clear();
            GUI_DispString("Foreground window");
            break;
        default:
            WM_DefaultProc(pMsg);
    }
}

/*********************************************************************
*                          μC/GUI 重绘机制展示                       *
*********************************************************************/

static void DemoRedraw(void)
{
    GUI_HWIN hWnd;
    while(1)
    {
    /* 创建前景窗口 */
        hWnd = WM_CreateWindow( 10, 10, 100, 100,
                                WM_CF_SHOW,
                                cbForegroundWin, 0);    /* 显示前景窗口 */
        GUI_Delay(1000);
    /* 删除前景窗口 */
        WM_DeleteWindow(hWnd);
        GUI_DispStringAt("Background of window has not been redrawn", 10, 10);
    /* 等待一会，背景不会重绘 */
        GUI_Delay(1000);
        GUI_Clear();
    /* 设置背景窗口的回调函数 */
        WM_SetCallback(WM_HBKWIN, cbBackgroundWin);
    /* 创建前景窗口 */
        hWnd = WM_CreateWindow( 10, 10, 100, 100,
                                WM_CF_SHOW,
```

```
                                    cbForegroundWin, 0);     /* 显示前景窗口 */
        GUI_Delay(1000);
    /* 删除前景窗口 */
        WM_DeleteWindow(hWnd);
    /* 等待一会，背景不会重绘 */
        GUI_Delay(1000);
    /* 删除背景窗口的回调函数 */
        WM_SetCallback(WM_HBKWIN, 0);
    }
}

/*********************************************************************
*                                主函数                              *
*********************************************************************/

void main (void)
{
    GUI_Init ();
    DemoRedraw();
}
```