延伸外挂模组

使用手册

2009/04/16

延伸外挂模组

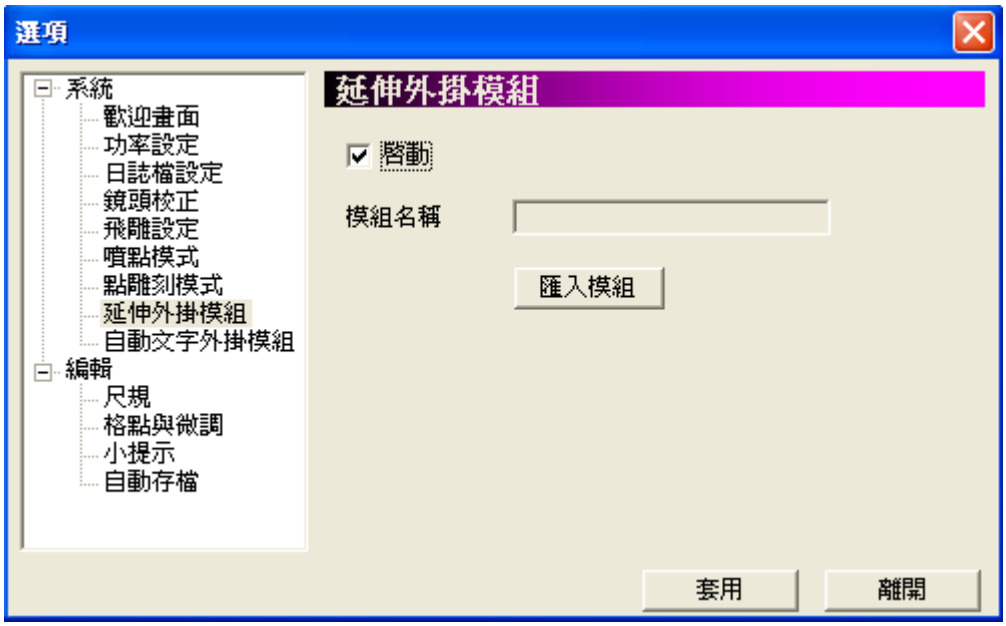
系统厂商可自行开发的一种Win32 DLL，依照实际流程，在特定的函式中，进行额外的流程；而打标系统会在特定的时机，呼叫这些函式。藉由此种机制，进行特殊的流程，快速制作出专用的打标系统。

1. 启动方法

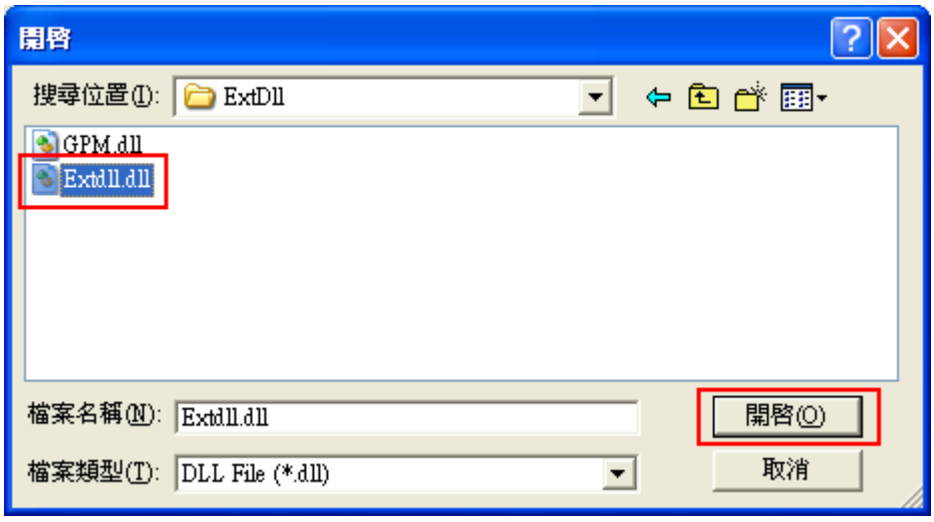
启动一个已完成的延伸外挂模组。

方法如下：

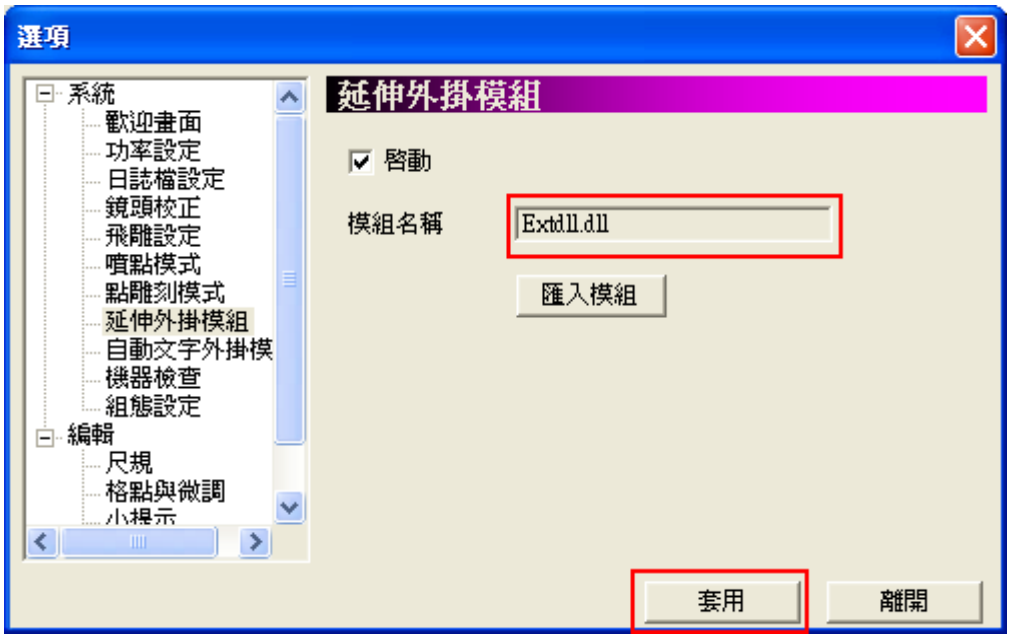
步骤1：选取功能表中的「选项」功能，会出现「选项」对话盒，接着选择左方的「延伸外挂模组」项目，最后在右方勾选「启动」。如下图所示：



步骤2：按下「汇入模组」按钮，接着出现「开启」对话盒，接着在上方的列表中选取所需的延伸外挂模组（副档名为.dll），最后按下「开启」按钮，完成「汇入模组」，如下图所示：

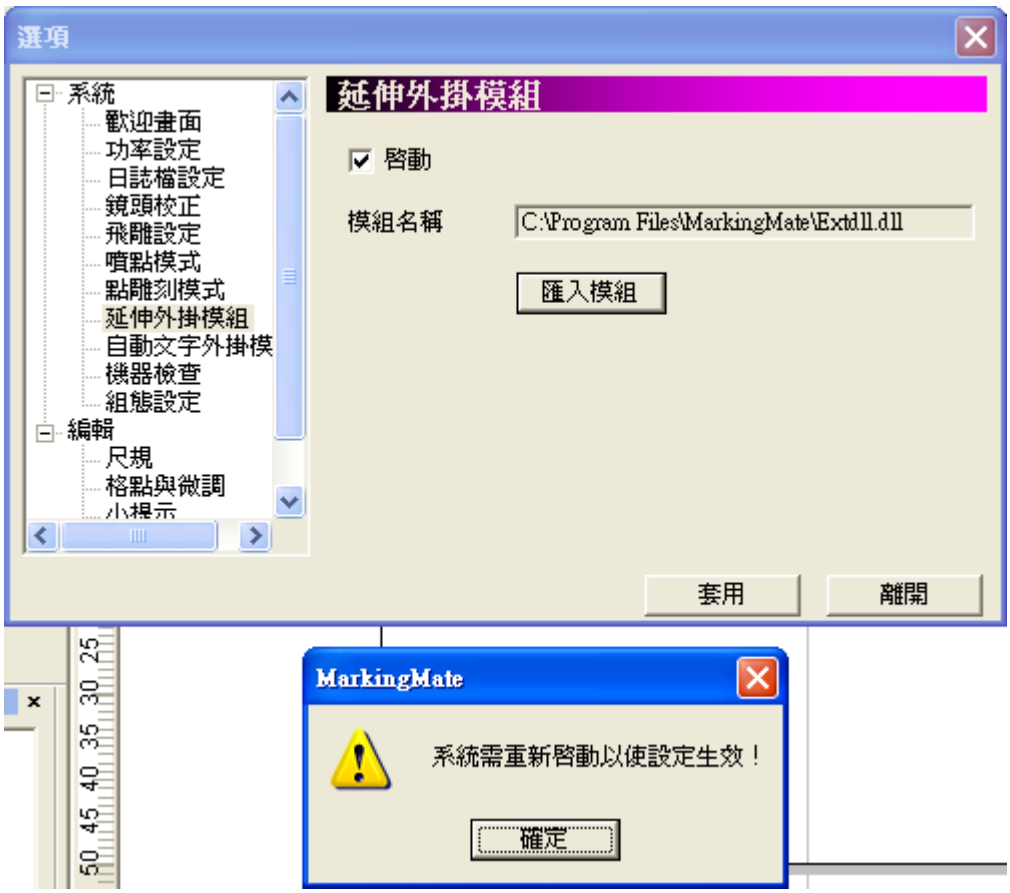


步骤3：请确认「模组名称」右方的名称，然后按下「套用」按钮，完成「启动延伸外挂模组」。



延伸外挂模组

步骤4：因打标系统必须重新载入所指定的延伸外挂模组，请离开打标系统，再重新开启打标系统，即会透过延伸外挂模组，来进行额外的流程。



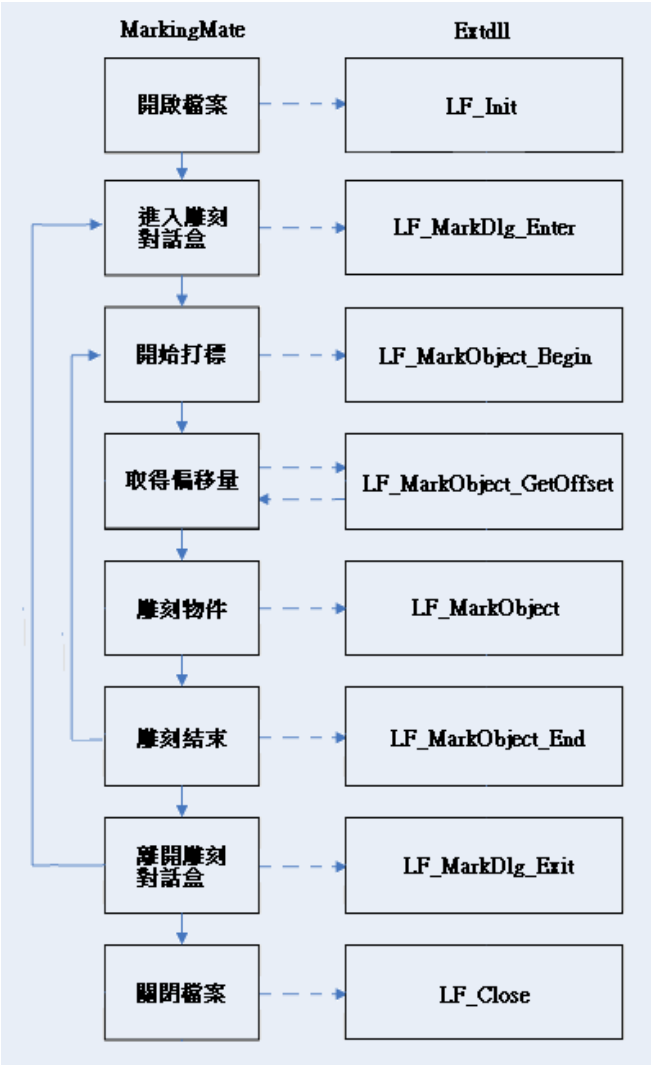
2. 函式说明

说明所有已支援的函式名称及用法。请依照实际流程，选择合适的函式并撰写所需的流程。

点选：「开始→所有程式→MarkingMate System → SDK → Extdll」，会弹出延伸外挂模组的范例程式的目录。范例中，建构了延伸外挂模组的每一个函式，并在进入每个函式时，都会显示一个讯息对话盒。正在摸索的软体工程师，可以将这个范例所编译出的DLL，用上节介绍的方式，安装到打标系统上，然后载入一个图档，进行打标作业。由画面上显示的对话盒以及次序，便可清楚的理解打标系统呼叫这些函式的时机。

范例为Microsoft Visual C++ 6.0所建立的Win32 DLL专案，若熟悉该编译器的软体工程师，以此为基础，扩充适合的函式，并将不需使用的函式予以删除，即成一个适用的延伸外挂模组。

主程式呼叫外挂模组的流程示意图：



延伸外挂模组

以下列出所有已支援的函式：

函式名称	**LF_Program_Init**
函式型态	int PASCAL LF_Program_Init (void)
呼叫时机	开启打标系统时，会呼叫此函式。
输入参数	无。
传回值	0: 成功，其他值：失败。

函式名称	**LF_Init**
函式型态	void PASCAL LF_Init (void)
呼叫时机	开启旧档及开启新档时，会呼叫此函式。 注意：若同时有多个档案被开启，此函式会被逐一呼叫。
输入参数	无。
传回值	无。

函式名称	**LF_MarkDlg_Enter**
函式型态	void PASCAL LF_MarkDlg_Enter (LPCTSTR pDocName)
呼叫时机	开启打标对话盒时，会呼叫此函式。 传入目前准备打标的档案名称。
输入参数	LPCTSTR pDocName：目前打标的档案名称。
传回值	无。

函式名称	**LF_MarkObject_Begin**
函式型态	void PASCAL LF_MarkObject_Begin (void)
呼叫时机	收到脚踏开关讯号，或按下打标对话盒的”执行”按钮时，会呼叫此函式。
输入参数	无。
传回值	无。

函式名称	**LF_MarkObject_GetOffset**
函式型态	void PASCAL LF_MarkObject_GetOffset (double* pOffX, double* pOffY, double* pR)
呼叫时机	呼叫MarkObject_Begin之后，会呼叫本函式。 传回工件位置的偏移值与旋转角度，以便修正实际的打标位置。 如 *pR = 45，则先将原图以(0, 0)为旋转中心并旋转45度后，再行打标。
输入参数	double* pOffX：X方向偏移量。单位：公厘。 double* pOffY：Y方向偏移量。单位：公厘。 double* pR：旋转角度。单位：度。
传回值	无。

函式名称	**LF_MarkObject**
函式型态	void PASCAL LF_MarkObject (int iType, LPCTSTR pTypeName, LPCTSTR pNickName, LPCTSTR pAutoTextStr)
呼叫时机	当一个物件（即档案中的图形）打标完成后，在下一个物件打标开始前，会呼叫本函式。 传入打标物件的资讯。

输入参数	int iType：物件类型编号。 LPCTSTR pTypeName：物件类型。定义如下： 1：点 2：线 3：圆 4：弧 6：文字 7：影像 9：曲线 LPCTSTR pNickName：物件名称。 LPCTSTR pAutoTextStr：物件内容（此类仅对文字类型的物件有效）。
传回值	无。

函式名称	**LF_MarkObject_End**
函式型态	void PASCAL LF_MarkObject_End (void)
呼叫时机	打标作业结束时，会呼叫本函式。
输入参数	无。
传回值	无。

函式名称	**LF_MarkDlg_Exit**
函式型态	void PASCAL LF_MarkDlg_Exit (void)
呼叫时机	离开打标对话盒时。
输入参数	无。
传回值	无。

函式名称	**LF_Close**
函式型态	void PASCAL LF_Close (void)
呼叫时机	关闭档案时，会呼叫此函式。
输入参数	无。
传回值	无。

函式名称	**LF_Program_Close**
函式型态	int PASCAL LF_Program_Close (void)
呼叫时机	关闭打标系统时，会呼叫此函式。
输入参数	无。
传回值	0: 成功，其他值：失败。

3. 应用实例：

以下是用 CCD（电脑视觉装置）计算工件位置补正值的应用实例。可参考以下的实例来制作此延伸外挂模组。

CCD 的动作如下：

A. 在打标系统开启时（呼叫 LF_Program_Init），开启 CCD 的 Modeless 对话盒。
B. 在打标系统关闭时（呼叫 LF_Program_Close），关闭 CCD 的 Modeless 对话盒。
C. 每当执行一个打标作业时（呼叫 LF_MarkObject_GetOffset），可透过 CCD 取得工件位置的偏移及旋转量，以修正打标位置。

以下是此实例的拟程式码，请注意：

A. 可参考后作适当修改，即可使用。
B. 所有 CCD_XXXX 的函式需自行撰写，并依照所用的 CCD 与实际流程来规划。

拟程式码如下：

```
EXPORT void PASCAL LF_Program_Init(void)
{
	CCD_INIT ();			// 将 CCD 初始化
	CCD_DLG_OPEN ();		// 开启 CCD 设定对话盒
}

EXPORT void PASCAL LF_MarkObject_GetOffset(double* pOffX, double* pOffY, double* pR)
{
	double dOffX, dOffY, dR;
	CCD_GETOFFSET (&dOffX, &dOffY, &dR);	// 从 CCD 取得偏移量
	*pOffX = dOffX;			// 填入偏移量
	*pOffY = dOffY;			// 填入偏移量
	*pR = dR;			// 填入偏移量
}

EXPORT void PASCAL LF_Program_Close(void)
{
	CCD_DLG_CLOSE ();		// CCD 设定对话盒关闭
	CCD_DESTROY ();			// 结束 CCD 功能。
}
```