第18章 输入设备

μC/GUI 提供触摸屏，鼠标，和键盘支持。 基本μC/GUI 程序包包括一个用于模拟触摸屏驱动程序和一个 PS2 鼠标驱动程序，不过别的种类的触摸板和鼠标装置在适当的驱动程序下也可以使用。 任何类型的键盘驱动程序都适合μC/GUI。

用于输入设备的软件位于子目录 GUI\Core 中。

18.1 指针光标输入设备

指针光标输入设备包括鼠标和触摸屏。 它们共用一组通用的指针光标输入设备（PID）函数使得鼠标和触摸屏能同时起作用。 该函数一般由视窗管理器自动地调用，如先前所描述的那样，起刷新显示屏的作用。 如果视窗管理器未使用，你的应用程序要负责调用 PID 函数。

数据结构

GUI_PID_STATE 类型的结构通过程序使用当前值填入的参数 pState 所引用。 该结构如下所术定义：

```
typedef struct {
  int x, y;
  unsigned char Pressed;
}GUI_PID_STATE;
```

指针光标输入设备API函数

下表按字母顺序列出指针光标输入设备函数。 函数的详细说明在稍后给出。

函　数	说　明
GUI_PID_GetState()	返回 PID 的当前状态。
GUI_PID_StoreState()	存储 PID 的当前状态。

GUI_PID_GetState()

描述

返回指针光标输入设备的当前状态。

函数原型

```
void GUI_PID_GetState(const GUI_PID_STATE *pState);
```

参数	含意
pState	指向一个 GUI_PID_STATE 类型的结构的指针。

返回值

如果输入设备当前被按下为 1；如果未按下为 0。

GUI_PID_StoreState()

描述

存储指针光标输入设备的当前状态。

函数原型

```
int GUI_PID_StoreState(GUI_PID_STATE *pState);
```

参数	含意
pState	指向一个 GUI_PID_STATE 类型的结构的指针。

18.1.1 鼠标输入驱动程序

鼠标支持由两个“层”组成：一个通用层和一个鼠标驱动程序层。 通用程序参考那些始终存在的函数，不管你使用什么样的鼠标驱动程序。 另一方面，该有效的鼠标驱动程序，会根据需要调用适当的通用函数，这些函数只能够用于μC/GUI 提供的 PS2 鼠标驱动程序。 如果你自己写驱动程序，在程序中你要负责调用这些通用函数。

通用鼠标函数会依次调用对应的 PID 函数。

通用鼠标API

下表按字母顺序列出了通用鼠标函数。 这些函数可以用于任何类型的鼠标驱动程序。 函数的详细说明在稍后给出。

函　数	说　明
GUI_MOUSE_GetState()	返回鼠标的当前状态。
GUI_MOUSE_StoreState()	存储鼠标的当前状态。

GUI_MOUSE_GetState()

描述

返回鼠标的当前状态。

函数原型

```
int GUI_MOUSE_GetState(GUI_PID_STATE *pState);
```

参数	含意
pState	指向一个 GUI_PID_STATE 类型的结构的指针。

返回值

如果鼠标当前被按下为 1；如果未按下为 0。

附加信息

该函数会调用 GUI_PID_GetState ()。

GUI_MOUSE_StoreState()

描述

存储鼠标的当前状态。

函数原型

```
void GUI_MOUSE_StoreState(const GUI_PID_STATE *pState) ;
```

参数	含意
pState	指向一个 GUI_PID_STATE 类型的结构的指针。

附加信息

该函数会调用 GUI_PID_StoreState ()。

PS2鼠标驱动程序API

下表按字母顺序列出了有效的鼠标驱动函数。 这些函数仅仅在你使用包括在μC/GUI 中的 PS2 鼠标驱动程序时才应用。

函　数	说　明
GUI_MOUSE_DRIVER_PS2_Init()	初始化鼠标驱动程序。
GUI_MOUSE_DRIVER_PS2_OnRx()	从接收中断程序调用。

GUI_MOUSE_DRIVER_PS2_Init()

描述

初始化鼠标驱动程序。

函数原型

```
void GUI_MOUSE_DRIVER_PS2_Init(void);
```

GUI_MOUSE_DRIVER_PS2_OnRx()

描述

必须从接收中断程序调用。

函数原型

```
void GUI_MOUSE_DRIVER_PS2_OnRx(unsigned char Data);
```

参数	含意
Data	通过中断服务程序（ISR）接收到的数据字节数。

附加信息

该 PS2 鼠标驱动程序是一种串行驱动程序，意思是它每次接受一个字节。

你需要保证这些函数从你的接收中断程序调用，每次接收一个字节（字符）。

18.1.2 触摸屏输入驱动程序和配置

触摸屏支持也是由一个通用层和一个驱动程序层组成。 通用函数用于任何类型的驱动程序（模拟，数字，等等.）。 该有效的模拟触摸屏驱动程序，会根据需要调用适当的通用函数，这些函数只能够用于μC/GUI 提供的模拟触摸屏驱动程序。 就象鼠标支持一样，如果你自己写驱动程序，在程序中你要负责调用这些通用函数。

通用触摸屏函数会调用对应的 PID 函数。

驱动程序层同时包括一个可能需要修改的配置模块。

通用触摸屏API

下表按字母顺序列出了通用触摸屏函数。 这些函数可以用于任何类型的触摸屏驱动程序。 函数的详细说明在稍后给出。

函 数	说 明
GUI_TOUCH_GetState()	返回触摸屏的当前状态。
GUI_TOUCH_StoreState()	存储触摸屏的当前状态。

GUI_TOUCH_GetState()

描述

返回触摸屏的当前状态。

函数原型

```
int GUI_TOUCH_GetState(GUI_PID_STATE *pState);
```

参数	含意
pState	指向一个 GUI_PID_STATE 类型的结构的指针。

返回值

如果触摸屏当前被按下为 1；如果未按下为 0。

GUI_TOUCH_StoreState()

描述

存储触摸屏的当前状态。

函数原型

```
void GUI_TOUCH_StoreState(int x int y);
```

参数	含意
x	X 轴坐标。
y	Y 轴坐标。

附加信息

该函数会调用 GUI_PID_StoreState()。

用于模拟触摸屏驱动程序API

µC/GUI 触摸屏驱动程序处理模拟输入（来自一个 8 位或更好的 A/D 转换器），对触摸屏进行去抖动和校准处理。

该触摸屏驱动程序通过使用函数 GUI_TOUCH_Exec ()连续地监视和刷新触摸板，该函数在它辨认出一个动作已经执行或者情况有所变化时，调用适当的通用触摸屏 API 函数。

下表按字母顺序列出了有效的模拟触摸屏驱动程序函数。 这些函数仅在你使用包括在 µC/GUI 中的驱动程序时才应用。

函　数	说　明
GUI_TOUCH_Calibrate()	更改刻度。
GUI_TOUCH _Exec()	激活 X 轴和 Y 轴的测量;需要大约每秒 100 次的调用。
GUI_TOUCH_SetDefaultCalibration()	恢复默认刻度。

TOUCH_X 函数

如果你使用µC/GUI 提供的驱动程序，下列四个与硬件相关函数需要加到你工程当中，而在轮询触摸板时，它们通过 GUI_TOUCH_Exec ()函数调用。 一个建议的位置是在文件 GUI_X.C 中。 这些函数在下表列出：

函　数	说　明
TOUCH_X_ActivateX()	准备 Y 轴的测量。
TOUCH_X_ActivateY()	准备 X 轴的测量。
TOUCH_X_MeasureX()	返回 A/D 转换器 X 轴的结果。
TOUCH_X_MeasureY()	返回 A/D 转换器 Y 轴的结果。

GUI_TOUCH_Calibrate()

描述

在运行时更改刻度。

函数原型

```
int GUI_TOUCH_Calibrate(int Coord, int Log0, int Log1, int Phys0, int Phys1);
```

参数	含意
Coord	用于 X 轴是 0，用于 Y 轴是 1。
Log0	以像素为单位逻辑值 0。
Log1	以像素为单位逻辑值 1。
Phys0	在 Log0 时 A/D 转换器的值。
Phys1	在 Log1 时 A/D 转换器的值。

附加信息

该函数把要校正的轴，两个用于该轴的以像素为单位的逻辑值和两个对应 A/D 转换器的物理值作为参数。

GUI_TOUCH_Exec()

描述

通过调用该 TOUCH_X 函数对触摸屏进行轮询，以激活 X 和 Y 轴的测量。 你必须保证这些函数大约每秒钟被调用 100 次。

函数原型

```
void GUI_TOUCH_Exec(void);
```

附加信息

如果你在使用一个实时操作系统，确定这些函数被调用的最轻松的方式是创建一个单独的任务。 当没有使用一个多任务系统时，你可以使用一个中断服务程序来做这项工作。

GUI_TOUCH_SetDefaultCalibration()

描述

将刻度复位为配置文件中设置的默认值。

函数原型

```
void GUI_TOUCH_SetDefaultCalibration(void);
```

附加信息

如果在配置文件中没有设置数值，该刻度将恢复到原始的默认值。

TOUCH_X_ActivateX()，TOUCH_X_ActivateY()

描述

从 GUI_TOUCH_Exec()调用这些函数以激活 X 和 Y 轴的测量。 TOUCH_X_ActivateX () 接通 X 轴的测量电压；TOUCH_X_ActivateY()接通 Y 轴的电压。接通 X 轴电压意思是 X 轴的值能够测量了，反之亦然。

函数原型

```
void TOUCH_X_ActivateX(void); void TOUCH_X_ActivateY(void);
```

TOUCH_X_MeasureX()，TOUCH_X_MeasureY()

描述

从 GUI_TOUCH_Exec ()调用这些函数以返回来自 A/D 转换器的 X 和 Y 轴的测定值。

函数原型

```
int TOUCH_X_MeasureX(void); int TOUCH_X_MeasureY(void);
```

配置触摸屏模块

在你的配置文件夹中需要有一单独的配置文件，命名为 GUITouchConf.h。 下表展示用于 µC/GUI 提供的模拟触摸屏驱动程序的所有配置宏：

类型	宏	默认	说明
B	GUI_TOUCH_SWAP_XY	0	设为 1 则 X 轴和 Y 轴相互交换。
B	GUI_TOUCH_MIRROR_X	0	X 轴镜像。
B	GUI_TOUCH_MIRROR_Y	0	Y 轴镜像。
N	GUI_TOUCH_AD_LEFT	30	由 A/D 转换器返回的最小值。
N	GUI_TOUCH_AD_RIGHT	220	由 A/D 转换器返回的最大值。
N	GUI_TOUCH_AD_TOP	30	由 A/D 转换器返回的最小值。
N	GUI_TOUCH_AD_BOTTOM	220	由 A/D 转换器返回的最大值。
N	GUI_TOUCH_XSIZE	LCD_XSIZE	被触摸屏覆盖的水平区域。

N	GUI_TOUCH_YSIZE	LCD_YSIZE	被触摸屏覆盖的垂直区域。

18.2 键盘输入

一个键盘输入设备使用 ASCII 字符编码，为了能够区别不同的字符。 例如，只有一个“A”键在键盘上，但是一个大写的“A”和一个小写的“a”的 ASCII 编码是不一样的（分别是 0x41 和 0x61）。

µC/GUI预定义字符代码

µC/GUI 也能定义字符代码，用于其它的“虚拟”键盘操作。 这些代码在下表列出，它们在 GUI.h 的一个标识符表中定义。 因此，一个在µC/GUI 中字符代码能够可以是 ASCII 字符值的任何扩展值或任一个下列的µC/GUI 预定义值。

预定义的虚拟键代码	描述
GUI_KEY_BACKSPACE	退格键。
GUI_KEY_TAB	TAB 键。
GUI_KEY_ENTER	回车/返回键。
GUI_KEY_LEFT	左箭头键。
GUI_KEY_UP	向上箭头键。
GUI_KEY_RIGHT	右箭头键。
GUI_KEY_DOWN	向下箭头键。
GUI_KEY_HOME	Home 键（移到当前行的开头）。
GUI_KEY_END	End 键（移到当前行的末端）。
GUI_KEY_SHIFT	换档键。
GUI_KEY_CONTROL	控制键。
GUI_KEY_ESCAPE	换码键。
GUI_KEY_INSERT	插入键。
GUI_KEY_DELETE	删除键。

18.2.1 驱动程序层 API

键盘驱动程序层操作键盘信息函数。 当具体的键（或者键组合）已经按下或释放时，程序会通知视窗管理器。

下表按字母顺序列出了驱动程序层键盘处理函数。 函数的详细描述在稍后给出。

函 数	说 明
GUI_StoreKeyMsg()	在一个指定键中存储一个状态消息。
GUI_SendKeyMsg()	向一个指定的按键发送一个状态消息。

GUI_StoreKeyMsg()

描述

在一个指定键中存储一个状态消息。

函数原型

```
void GUI_StoreKeyMsg(int Key, int Pressed);
```

参数	含意
Key	可以是任何可扩展的 ASCII 字符（在 0x20 和 0xFF 之间）或者任何预定义的μC/GUI 信息码。
Pressed	键的状态，参考下面说明。

参数 pressed 允许的值

1： 按下状态。
0： 释放（未按下）状态

GUI_SendKeyMsg()

描述

向一个指定的按键发送一个状态消息。

函数原型

```
void GUI_SendKeyMsg(int Key, int Pressed);
```

参数	含意
Key	可以是任何可扩展的 ASCII 字符（在 0x20 和 0xFF 之间）或者任何预定义的μC/GUI 信息码。
Pressed	键的状态（参 GUI_StoreKeyMsg()）

18.2.2 应用层 API

下表按字母顺序列出了应用层键盘处理函数。 函数的详细说明在稍后给出。

函 数	说 明
GUI_ClearKeyBuffer()	清除键缓冲区。

GUI_GetKey()	返回键缓冲区的内容。
GUI_StoreKey()	在缓冲区中存储一个键。
GUI_WaitKey()	等待一个键被按下。

GUI_ClearKeyBuffer()

描述

清除键缓冲区。

函数原型

```
void GUI_ClearKeyBuffer(void);
```

GUI_GetKey()

描述

返回键缓冲区的当前内容。

函数原型

```
int GUI_GetKey(void);
```

返回值

在键缓冲区中的字符代码；如果没有键在缓冲区中为 0。

GUI_StoreKey()

描述

在缓冲区中存储一个键。

函数原型

```
void GUI_StoreKey(int Key);
```

参数	含意
Key	可以是任何可扩展的 ASCII 字符（在 0x20 和 0xFF 之间）或者任何预定义的μC/GUI 信息码。

附加信息

通常该函数通过驱动程序调用而不是通过应用程序本身调用。

GUI_WaitKey()

描述

等待一个键被按下。

函数原型

```
int GUI_WaitKey(void);
```

附加信息

该应用是“锁定”的，意思是它在一个按键被按下之前是不会返回的。