

目录

第 1 章 键 盘相关操作.....	5
1、按键对应值列表.....	5
2、key_led_init.....	5
3、get_key.....	6
第 2 章 基本界面操作函数.....	7
1、tft_init.....	7
2、lcd_160_clr.....	7
3、ds_init.....	7
4、dis_mod_set.....	8
5、printf_str.....	9
6、line.....	9
7、rectangle.....	10
8、box.....	10
9、bmp_display.....	11
10、lcd_160_upd.....	11
11、fs_print_dat.....	12
12、fs_print_status.....	13
13、box_bak.....	14
第 3 章 控件操作函数.....	14
1、piclist_init.....	15
2、property_set.....	15
3、piclist_add.....	16
4、piclist_props_set.....	16
5、piclist_show.....	17
6、piclist_key_manege.....	17
7、piclist_destory.....	18
8、list_init.....	20
9、list_property_set.....	20
10、list_item_add.....	21
11、list_props_set.....	22
12、list_show.....	22
13、list_key_manege.....	23
14、list_exit.....	23
15、input_init.....	25
16、input_property_set.....	26
17、input_props_set.....	26
18、input_dis.....	26
19、input_key_manege.....	27
20、input_val_get.....	27
21、input_str_get.....	27
22、input_destory.....	28
23、label_init.....	30

24、label_property_set.....	31
25、label_props_set.....	32
26、label_dis.....	32
27、label_destory.....	32
28、choise_init.....	34
29、choice_property_set.....	35
30、choice_props_set.....	35
31、choise_key_manege.....	35
32、choise_destory.....	36
33、beep.....	36
34、led_updat.....	36
35、dis_buf_tran.....	37
第4章 卡类函数说明.....	37
1、dc_init_rf.....	38
2、dc_exit_rf.....	38
3、dc_reset.....	38
4、dc_config_card.....	38
5、dc_request.....	39
6、dc_anticoll.....	39
7、dc_select.....	39
8、dc_card.....	40
9、dc_authentication_pass.....	40
10、dc_readval.....	41
11、dc_decrement.....	41
12、dc_increment.....	41
13、dc_initval.....	42
14、dc_restore.....	42
15、dc_write.....	42
16、dc_read.....	43
17、dc_pro_reset.....	43
18、dc_pro_command.....	43
19、dc_int_sam.....	44
20、dc_setcpu.....	44
21、dc_setcpupara.....	45
22、dc_cpureset.....	45
23、dc_cpuapdu.....	46
24、dc_exit_sam.....	46
25、maginit.....	46
26、magdata.....	47
第5章 GPRS 模块函数说明.....	47
1、gprs_set_baud.....	47
2、gprs_test.....	47
3、gprs_close_return.....	48
4、gprs_sleep.....	48

5、gprs_check_signal.....	48
6、gprs_check_version.....	49
7、gprs_close.....	49
8、gprs_set_function.....	49
9、gprs_init_apn.....	50
10、gprs_init_tcpip.....	50
11、gprs_open_tcp.....	50
12、gprs_querfy_data.....	51
13、gprs_set_datadeletmode.....	51
14、gprs_set_datamode.....	51
15、gprs_send_tcpdata.....	52
16、gprs_receive_tcpdata.....	52
17、gprs_close_tcp.....	53
18、modem_gprs_sem_init.....	53
19、gprs_sel_sem_wait.....	53
20、modem_gprs_sem_end.....	53
21、gprs_initapn.....	54
22、gprs_inittcpip.....	54
23、gprs_buff_check.....	54
24、gprs_pin_rst.....	55
25、power_pin_set.....	55
第 6 章 CDMA 模块函数说明.....	55
1、cdma_close_return.....	55
2、cdma_check_signal.....	56
3、cdma_check_sim.....	56
4、cdma_set_user.....	56
5、cdma_build_ppp.....	57
6、cdma_closed_ppp.....	57
7、cdma_inquire_ppp.....	57
8、cdma_build_tcpip.....	58
9、cdma_cancel_tcp.....	58
10、cdma_check_tcp.....	59
11、cdma_close_tcpip.....	59
12、cdma_send_tcpdata.....	59
13、cdma_tcp_recv.....	60
第 7 章 TCP/IP 通讯模块函数说明.....	60
1、connecttcpserver.....	60
2、sendtcpmessage.....	60
3、recvtcpmessage.....	61
4、tcpclose.....	61
第 8 章 常用功能函数.....	62
1、fun_hextoascii.....	62
2、fun_asciitohex.....	62
3、fun_clear_buff.....	62

4、fun_find_byte.....	63
5、fun_find_nbyte.....	63
6、fun_ascii_bcd.....	63
7、fun_bcd_ascii.....	64
8、fun_exp2_n.....	64
9、fun_inversion_bit.....	64
第9章 附录.....	65
1、数据类型原型列表.....	65

第1章键 盘相关操作

1、按键对应值列表

按键值

```
#define F1_KEY -64
#define F2_KEY -72
#define F3_KEY -80
#define F4_KEY -88

#define ENT_KEY 92
#define ESC_KEY 84

#define KEY_1 65
#define KEY_2 66
#define KEY_3 67
#define KEY_4 73
#define KEY_5 74
#define KEY_6 75
#define KEY_7 81
#define KEY_8 82
#define KEY_9 83
#define KEY_0 90
#define KEY_DOT 89
#define KEY_UP 68
#define KEY_DN 76

#define KEY_SHIFT 91
#define KEY_12315 96
```

2、key_led_init

函数原型	int32 key_led_init(void)
参数说明	无
返回	0: 成功 -1: 失败

功能说明	初始化按键和 LED ，一般在应用程序开始时候调用
实例	key_led_init; //键盘、显示模块初始化

3、get_key

函数原型	int32 get_key(uint8 *key_ptr , uint32 mod , uint32 dtim)
参数说明	uint8 *key_ptr:按键指针值 uint32 mod: 寻键模式 uint32 dtim: 等待时间 mod 寻键模式，有如下三种 enum get_key_mod { BLOCK_MOD, //阻塞模式，有按键才返回 NOBLOCK_MOD, //查询模式，查询一下就返回 TIM_DLY_MOD, //延迟模式，在 dtim 指定的时间内等待按键，有则立即返回，无则超时退出 }; dtim 时间值以 ms 为单位，mod 为 TIM_DLY_MOD 有效
返回	0: 有按键 -1: 无按键
功能说明	获得键盘输入的一个按键值
实例	在一秒内等待按键，及处理程序 ret = get_key(&key_val,TIM_DLY_MOD , 1000) ; if(0 == ret) //有按键 { if(ESC_KEY == key_val) { } Else { } else //超时无按键 { } } }

第2章 基本界面操作函数

说明：P16 中将液晶和打印机，合并为对相应的内存操作。共用相同的界面操作函数。

1、tft_init

函数原型	int32 tft_init(void)
参数说明	无
返回	0: 成功 -1: 失败
功能说明	初始化液晶，才可进行后续液晶操作
实例	<pre>if(0 != tft_init()) { printf("open lcd_160 error \n") ; return(-1) ; }</pre>

2、lcd_160_clr

函数原型	Void lcd_160_clr(void)
参数说明	无
返回	无
功能说明	液晶清屏
实例	<pre>lcd_160_clr() ;</pre>

3、ds_init

函数原型	dis_map * ds_init(uint16 lin , uint16 vol , uint16 mod)
参数说明	lin 行数 vol 列数 mod 模式—指定 enum DIS_MOD { BIT_L, // 打印机

	<pre> BIT_V, BIT_4, // 160*160 液晶 CLOR16 , CLOR24 , MOD_EN , } ; </pre>
返回	返回 dis_map 地址
功能说明	为液晶和打印机，申请相关操作内存
实例	<p>为液晶申请显存</p> <pre> dis_map *mianwin_ptr = NULL ; mianwin_ptr = ds_init(480,272,TFT_16) ; if(NULL == mianwin_ptr) { printf("ds init error \n") ; exit(1) ; } </pre>

4、dis_mod_set

函数原型	void dis_mod_set(dis_map * ptr ,uint16 x , uint16 y)
参数说明	<p>ptr 显示设备 X 汉字字体 Y asc 字体 系统支持的字体类型如下：</p> <pre> enum { ASC_5DZ , //asc ASC_8DZ , ASC_12DZ , ASC_16DZ , ASC_24DZ , ASC_32DZ , HZ_12DZ , //汉字 HZ_16DZ , HZ_24DZ , HZ_32DZ , } ; </pre>
返回	无

功能说明	设置显示字体 ， 后续在显示设备 ptr， 上用此字体输出
实例	设置汉字 16 点阵 asc 12 点阵显示字体 dis_mod_set(mianwin_ptr ,HZ_16DZ ,ASC_12DZ) ；

5、printf_str

函数原型	void printf_str(dis_map * dptr, uint16 x, uint16 y, uint8 *ptr , uint32 f_color)
参数说明	dptr 显示设备 x 行数 y 列数 ptr 要显示的字符串 color 颜色
返回	无
功能说明	在指定的显示设备上 ， 显示相关的字体
实例	dis_mod_set(mianwin_ptr ,HZ_16DZ ,ASC_12DZ) ； printf_str(mianwin_ptr, 50 ,60, ” 请刷卡” , 0) ；

6、line

函数原型	int32 line(dis_map * ptr , int32 x0 , int32 y0 , int32 x1 , int32 y1 ,uint32 color)
参数说明	ptr 显示设备 X0 开始行 Y0 开始列 X1 结束行 Y1 结束列 color 颜色
返回	int（可不关注）
功能说明	在指定的显示设备上 ， 画线
实例	Line(mianwin_ptr , 5 ,46,155,46,1) ；

7、rectangle

函数原型	<code>int32 rectangle(dis_map * ptr ,int32 x0 , int32 y0 , int32 x1 , int32 y1 ,uint32 color)</code>
参数说明	ptr 显示设备 X0 开始行坐标 Y0 开始列坐标 X1 结束行坐标 Y1 结束列坐标 color 颜色
返回	int（可不关注）
功能说明	在指定的显示设备上，画线
实例	<code>rectangle (mianwin_ptr , 5 ,46,155,46,1) ;</code>

8、box

函数原型	<code>int32 box(dis_map * ptr , uint16 x0 , uint16 y0 , uint16 x1 , uint16 y1 ,uint32 color)</code>
参数说明	ptr 显示设备 X0 开始行坐标 Y0 开始列坐标 X1 结束行坐标 Y1 结束列坐标 color 颜色
返回	int（可不关注）
功能说明	用指定颜色填充显示设备的矩形区域，一般用做清除显示设备一小块区域。也可用来产生正反显示效果
实例	<code>box(mianwin_ptr, 0,20,160,160, 1) ; //填充黑色 dis_mod_set(mianwin_ptr ,HZ_16DZ ,ASC_12DZ) ; printf_str(mianwin_ptr, 5, 30, cptr, 0) ; //字体反显 line(mianwin_ptr , 5 ,46,155,46,0) ; //直线反显</code>

9、bmp_display

函数原型	int32 bmp_display(dis_map *tstptr, uint16 x, uint16 y , uint8 *fptr)
参数说明	tstptr 显示设备，如液晶，打印机 x 行位置 y 列位置 fptr bmp 图片的文件名路径
返回	0: 正常 -1: 失败
功能说明	在相关显示设备上显示
实例	<p>在消费小票上打上 logo 图标</p> <pre> #define logo_path "../res/bmp/dc_log.bmp" int32 fs_print_val(uint32 snr , uint32 val) { dis_map *fstptr = NULL ; uint8 disbuf[32] ; int32 ret ; uint8 step ; fstptr = ds_init(FS_DOT_NUM, 450 ,BIT_L) ; if(NULL == fstptr) { printf("ds init error \n") ; exit(1) ; } bmp_display(fstptr, 1 , 1, logo_path) ; //显示 logo 到打印内存 dis_mod_set(fstptr , HZ_32DZ ,ASC_24DZ) ; printf_str(fstptr, 120, 1 + 120 , "消费小票", 1) ; fs_print_dat(fstptr->ptr, fstptr->buf_size) ; // 打印输出 } </pre>

10、lcd_160_upd

函数原型	int32 lcd_160_upd(void)
------	-------------------------

参数说明	无
返回	int（可不关注）
功能说明	更新显示缓冲区的数据到液晶上
实例	<pre> box(mianwin_ptr, 0, 20, 160, 160, 1) ; //填充黑色 dis_mod_set(mianwin_ptr ,HZ_16DZ ,ASC_12DZ) ; printf_str(mianwin_ptr, 5, 30, cptr, 0) ; //字体反显 line(mianwin_ptr , 5 , 46, 155, 46, 0) ; //直线反显 lcd_160_upd() ; //更新显示 </pre>

11、fs_print_dat

函数原型	int32 fs_print_dat(uint8 *ptr , uint32 lenth)
参数说明	ptr 内存 Lenth 长度
返回	int（可不关注）
功能说明	更新显示缓冲区的数据到打印机上
实例	<p>打印一次消费小票</p> <pre> int32 fs_print_val(uint32 snr , uint32 val) { dis_map *fstptr = NULL ; //打印设备 uint8 disbuf[32] ; time_t timep; struct tm *p; int32 ret ; uint8 step ; fstptr = ds_init(FS_DOT_NUM, 320 ,BIT_L) ;//申请打印显示内存 f(NULL == fstptr) { printf("ds init error \n") ; exit(1) ; } dis_mod_set(fstptr ,HZ_32DZ ,ASC_24DZ) ;//设置字体为汉字 32, asc24 printf_str(fstptr, 120, 1, "消费小票", 1) ; //在打印内存上显示信息 dis_mod_set(fstptr ,HZ_24DZ ,ASC_24DZ) ; printf_str(fstptr, 30, 50, "消费金额:", 1) ; </pre>

	<pre> val_tran_asc(val ,disbuf); printf_str(fstptry, 150, 50, disbuf, 1) ; printf_str(fstptry, 30, 80, "消费时间:", 1) ; time(&timep); p = gmtime(&timep); sprintf(disbuf,"%d-%d-%d%d :%d", (1900+p->tm_year), (1+p->tm_mon), (p->m_mday), p->tm_hour, p->tm_min) ; printf_str(fstptry, 150, 80, disbuf, 1) ; sprintf(disbuf,"%u", snr); printf_str(fstptry, 30, 110, "卡 号:", 1) ; printf_str(fstptry, 150, 110, disbuf, 1) ; printf_str(fstptry, 30, 160, "**** 欢迎惠顾 ****", 1) ; fs_print_dat(fstptry->ptr, fstptry->buf_size) ; //打印输出 } </pre>
--	---

12、fs_print_status

函数原型	int32 fs_print_status(int*step)
参数说明	int*step:打印进度
返回	0: 打印完成 1: 打印中 -1: 缺纸
功能说明	监视打印情况
实例	<pre> while(1) { ret = fs_print_status(&step) ; printf("fs_print_status :%d \n", ret) ; if(1 == ret) { plan_step_dis(mianwin_ptr, pptr, lcd_160_upd , step) ; lcd_160_upd() ; } else if(-1 == ret) { plan_step_dis(mianwin_ptr, pptr, lcd_160_upd , step) ; lcd_160_upd() ; break ; } } </pre>

	<pre> } else if(0 == ret) { plan_step_dis(mianwin_ptr, pptr, lcd_160_upd , 100) ; lcd_160_upd() ; break ; } } </pre>
--	---

13、box_bak

函数原型	int32 box_bak(dis_map * ptr , uint16 x0 , uint16 y0 , uint16 x1 , uint16 y1)
参数说明	<p>ptr 显示设备</p> <p>X0 开始行坐标</p> <p>Y0 开始列坐标</p> <p>X1 结束行坐标</p> <p>Y1 结束列坐标</p>
返回	int（可不关注）
功能说明	用指定颜色填充显示设备的矩形区域，一般用做清除显示设备一小块区域。也可用来产生正反显示效果
实例	<pre> mianwin_ptr = ds_init(480,272,TFT_16) ; box_bak(mianwin_ptr, 0,0,480,272) ; //显示背景 dis_mod_set(mianwin_ptr ,HZ_16DZ ,ASC_12DZ) ; printf_str(mianwin_ptr, 5, 30, cptr, 0) ; //字体反显 line(mianwin_ptr , 5 ,46,155,46,0) ; //直线反显 </pre>

第3章控件操作函数

基本概念

- 1) P16 针对图形显示设备，提供如下常用控件， 图片列表选择框(pic_lst)，列表选择框(list)， 输入框(input)，标题框 (label)，选择

框（choise）等基本常用的界面元素，便于用户快速二次开发。

2) 控件的使用一般为如下步骤，

- <1>申请控件元素
- <2>设置显示属性
- <3>添加控件元素
- <4>显示控件
- <5>键盘托管
- <6>销毁控件

3) 控件使用上述基本元素编制，用户也可自行编制
图片控件

1、piclist_init

函数原型	struct pic_lst *piclist_init(void)
参数说明	无
返回	成功返回新控件指针，失败返回 NULL
功能说明	初始化一个图片列表选择控件
实例	<pre>if((main_menu_ptr = piclist_init()) == NULL) { exit(1) ; }</pre>

2、property_set

函数原型	int32 piclist_property_set(struct pic_lst *ptr ,int comm , ...)
参数说明	<p>ptr 图片列表选择控件指针</p> <p>Comm. 命令</p> <p>enum pic_lst_comm //命令列表</p> <pre>{ BMP_STAT_SET , //显示开始位置 BMP_SIZE_SET , //显示大小尺寸 BMP_INTE_SET , //图片间隔 BMP_DMOD_SET , //设置显示属性（显示与不显示） BMP_CAPEN_SET , //有无提示 BMP_CAPMOD_SET , //提示模式设置（参见标题框控件） BMP_CAPSTA_SET , //提示位置设置 BMP_CAPSIZ_SET , //提示框大小设置 BMP_CAPFON_SET , //提示字体设置 }</pre>

	};
返回	0: 正常 -1: 失败
功能说明	设置一个图片列表选择控件显示属性
实例	<pre> struct pic_lst * ptr1 ; ptr1 = piclist_init () ; piclist_property_set (ptr1,BMP_STAT_SET,2 ,60) ; </pre>

3、piclist_add

函数原型	int32 piclist_add(struct pic_lst *ptr , uint8 *cptr ,uint8 *cptr1)
参数说明	ptr 图片列表选择控件 cptr 图片的路径 cptr1 图片的选择提示信息（可有可无，无的时候象最上面的电量提示，无信息）
返回	成功返回新控件指针，失败返回 NULL
功能说明	初始化一个图片列表选择控件
实例	<pre> piclist_add(ptr1, "../res/bmp/logoff.bmp", " 关机"); </pre>

4、piclist_props_set

函数原型	void piclist_props_set(struct pic_lst *ptr , int8 *cptr[] , proptey *pptr)
参数说明	ptr 图片列表选择控件 cptr piclist_add 的多个参数表 cptr1 piclist_property_set 的多个参数表
返回	无
功能说明	初始化一个图片列表选择控件，向图片列表选择控件添加元素
实例	<pre> int8 *main_sel_tab[] = { PIC1, "射频卡消费", PIC2, "接触卡消费", PIC3, "磁卡消费", PIC4, "参数设置", PIC5, "状态查询", </pre>

	<pre> PIC6, "关机", NULL, NULL }; propkey main_sel_proty[] = { {BMP_STAT_SET , 2 , 60}, //显示的开始位置 {BMP_SIZE_SET , 36 , 40}, //显示大小尺寸 {BMP_INTE_SET , 4 , 0}, //图片的间隔 {BMP_CAPMOD_SET , 4 , STA_CENTER}, //设置提示模式 {BMP_CAPEN_SET , 1 , 0}, //有无提示 {BMP_CAPSIZ_SET , 160 , 16}, //提示框大小 {BMP_CAPSTA_SET , 0 , 40}, //提示框位置 {0xffff , 0 , 0} } ; piclist_props_set(main_menu_ptr, main_sel_tab , main_sel_proty) ; </pre>
--	---

5、piclist_show

函数原型	piclist_show(dis_map *dptr ,struct pic_lst *ptr)
参数说明	dptr 显示设备 Ptr 图片列表选择控件
返回	成功返回新控件指针，失败返回 NULL
功能说明	将一个图片列表选择控件更新到显示设备内存
实例	<pre> piclist_show(mianwin_ptr,main_menu_ptr) ; lcd_160_upd() ; </pre>

6、piclist_key_manege

函数原型	int32 piclist_key_manege(dis_map *dptr ,struct pic_lst *ptr, int32(* funp) (void))
参数说明	dptr 显示设备 ptr 图片列表选择控件 funp 跟新方式
返回	-1 取消按键 >0 当前选择的项 F1 (0) F2(1) F3 (2) F4 (3) 快捷选项
功能说明	键盘托管，让图片列表选择控件，管理键盘消息

实例	<pre> ret = piclist_key_manege(mianwin_ptr, main_menu_ptr , lcd_160_upd) ; if(0 == ret) { RF_consume_func() ; break; } if(1 == ret) { test_choise() ; break; } else if(3 == ret) { sys_set() ; break; } else { } </pre>
----	---

7、piclist_destory

函数原型	void piclist_destory(struct pic_lst *ptr)
参数说明	Ptr: 要销毁的控件
返回	无
功能说明	销毁一个图片列表选择控件内存及数据
实例	piclist_destory(main_menu_ptr) ;

综合示例

```

#define TEST_BMP        "../res/bmp/test.bmp"
#define TEST_BMP1       "../res/bmp/test_hand24.bmp"
#define TEST_BMP2       "../res/bmp/test_hand2316_24.bmp"
#define TEST_BMP3       "../res/bmp/test_hand256.bmp"
#define XINHAO4_BMP     "../res/bmp/xinhao4.bmp"
#define DIANCIO_BMP     "../res/bmp/dianci0.bmp"
#define DIANCII_BMP     "../res/bmp/dianci1.bmp"
#define DIANCII2_BMP    "../res/bmp/dianci2.bmp"
#define DIANCII3_BMP    "../res/bmp/dianci3.bmp"
#define DIANCII4_BMP    "../res/bmp/dianci4.bmp"
#define LIANJIE_BMP     "../res/bmp/lianjie.bmp"

```

```

#define PIC1          "../res/bmp/mediaplayer.bmp"
#define PIC2          "../res/bmp/logoff.bmp"
#define PIC3          "../res/bmp/windows.bmp"
#define PIC4          "../res/bmp/cogwheels.bmp"
#define PIC5          "../res/bmp/search_v2.bmp"
#define PIC6          "../res/bmp/Shutdown.bmp"

int8  *main_sel_tab[] =
{
    PIC1,  "射频卡消费",
    PIC2,  "接触卡消费",
    PIC3,  "磁卡消费",
    PIC4,  "参数设置",
    PIC5,  "状态查询",
    PIC6,  "关机",
    NULL,  NULL
};

proptey  main_sel_proty[] =
{
    {BMP_STAT_SET ,   2 , 60},
    {BMP_SIZE_SET ,   36 , 40},
    {BMP_INTE_SET ,    4 , 0},
    {BMP_CAPMOD_SET ,    4 , STA_CENTER},
    {BMP_CAPEN_SET ,    1 , 0},
    {BMP_CAPSIZ_SET ,   160 , 16},
    {BMP_CAPSTA_SET ,    0 , 40},
    {0xffff ,          0 , 0}
} ;

if((main_menu_ptr = piclist_init()) == NULL)
{
    exit(1) ;
}
piclist_props_set(main_menu_ptr, main_sel_tab , main_sel_proty) ;
box(mianwin_ptr, 0,20,160,160, 0) ;
piclist_show(mianwin_ptr,main_menu_ptr) ;
lcd_160_upd() ;

ret = piclist_key_manege(mianwin_ptr, main_menu_ptr , lcd_160_upd) ;
if(0 == ret)
{
    RF_consume_func() ;
}

```

```

if(1 == ret)
{
    test_choise() ;
}
else if(3 == ret)
{
    sys_set() ;
}
else if(4 == ret)
{
    sys_mange() ;
}
else if(5 == ret)
{
    power_off() ;
    break ;
}
else
{
}
}

```

列表框控件

8、list_init

函数原型	list_dst *list_init(void)
参数说明	无
返回	成功返回新控件指针，失败返回 NULL
功能说明	申请列表控件
实例	<pre> list_dst *ptr ; ptr = list_init() ; if(NULL == ptr) { exit(1) ; } </pre>

9、list_property_set

函数原型	int list_property_set(list_dst *ptr ,int comm , ...)
参数说明	<p>ptr 列表框控件</p> <p>Comm. 命令</p>

	<pre>enum LST_PROPER { LST_STA_STATION , //显示位置 LST_LST_TYPE , //带滚动条与否 LST_WIDTH , //宽度 LST_ROW_HIGH , //行高 LST_SEL_COLOR , //选择颜色（彩色液晶而言） LST_FON_COLOR , //字体颜色（彩色液晶而言） LST_BAK_COLOR , //背景颜色（彩色液晶而言） LST_FONT_TYPE , //字体类型 LST_ROW_LINE , //行列数 LST_BAR_TYP , //带滚动条与否 LST_SEL_MOD , //被选字体颜色（彩色液晶而言） } ; ... 参数</pre>
返回	0: 成功 -1: 失败
功能说明	设置一个列表框控件显示属性
实例	list_property_set (ptr1, LST_STA_STATION , 2 ,60) ;

10、list_item_add

函数原型	Void list_item_add(list_dst *ptr , ...)
参数说明	ptr 图片列表选择控件 ... 根据列的数目决定，字符串个数
返回	无
功能说明	向列表选择控件添加元素
实例	<pre>1. list_item_add(ptr, “文件1” ,” 大小”, ” 100.2K”) ; 2. list_item_add(ptr , ”1>时间参数设置”) ; list_item_add(ptr , ”2>网络参数设置”) ; list_item_add(ptr , ”3>黑名单管理”) ; list_item_add(ptr , ”4>M1 卡钱包充值”) ; list_item_add(ptr , ”5>gprs 测试”) ; list_item_add(ptr , ”6>射频卡测试”) ; list_item_add(ptr , ”7>磁卡测试”) ; list_item_add(ptr , ”8>接触卡测试”) ;</pre>

11、list_props_set

函数原型	Void list_props_set(list_dst *ptr , propkey *pptr)
参数说明	ptr 图片列表选择控件 ... 根据列的数目决定，字符串个数
返回	无
功能说明	ptr 列表控件 pptr list_property_set 的多个参数表
实例	<pre> propkey sys_set_proty[] = { {LST_LST_TYPE , LST_NOTILE , 0}, //带标题框与否 {LST_WIDTH , 125,0}, //宽度 {LST_STA_STATION , 20 ,65}, //显示位置 {LST_FONT_TYPE , HZ_16DZ , ASC_12DZ}, //字体类型 {LST_BAR_TYP , 1 , 0}, //带滚动条与否 {LST_ROW_LINE , 4 , 1}, //行列数 {LST_SEL_MOD , 1 , 0}, //被选字体颜色 {LST_ROW_HIGH , 16 , 1}, //行高 {0xffff , 0 , 0} } ptr = list_init() ; if(NULL == ptr) { exit(1) ; } list_props_set(ptr , sys_set_proty) ; </pre>

12、list_show

函数原型	void list_show(dis_map *dptr , list_dst *ptr)
参数说明	dptr 显示设备 Ptr 列表选择控件
返回	无
功能说明	将一个列表选择控件更新到显示设备内存
实例	<pre> list_show(mianwin_ptr , lptr) ; lcd_160_upd() ; </pre>

13、list_key_manege

函数原型	int32 list_key_manege(dis_map *dptr ,list_dst *ptr, int32(*funp) (void))
参数说明	dptr 显示设备 ptr 列表选择控件 funp 跟新方式
返回	-1 取消按键 >0 当前选择的项
功能说明	让列表选择控件，管理键盘信息
实例	<pre>ret = list_key_manege(mianwin_ptr, lptr, lcd_160_upd) ; if(ret < 0) { list_exit(lptr) ; return ; } else if(0 == ret) { dec_val_manege() ; } else if(1 == ret) { read_val_manege() ; }</pre>

14、list_exit

函数原型	void list_exit(list_dst *ptr)
参数说明	Ptr: 销毁的控件
返回	无
功能说明	销毁一个列表选择控件内存及数据
实例	list_exit (lptr) ;

综合示例

```
proptey sel_lst_proty[] =
{
    {LST_LST_TYPE , LST_NOTILE , 0},
```

```

    {LST_WIDTH ,      90, 0},
    {LST_STA_STATION ,  40 , 77},
    {LST_FONT_TYPE ,  HZ_16DZ , ASC_12DZ},
    {LST_BAR_TYP ,      1 , 0},
    {LST_ROW_LINE ,    2 , 1},
    {LST_SEL_MOD ,      1 , 0},
    {LST_ROW_HIGH ,    18 , 1},
    {0xffff ,          0 , 0}
} ;

void RF_consume_func(void)
{
    uint8      *cptr = "德卡 P16 小额消费 POS 机";
    uint8      *cptr1 = "欢迎使用";
    uint8      *cptr2 = "功能选择";

    uint8      *cptr4 = "请刷卡....";
    uint8      *cptr5 = "深圳德卡科技";

    list_dst   *lptr ;
    int32      ret ;

    lptr = list_init() ;
    if(NULL == lptr)
    {
        exit(1) ;
    }

    list_props_set(lptr , sel_lst_proty) ;
    list_item_add(lptr , "1 卡片消费") ;
    list_item_add(lptr , "2 余额查询") ;

    while(1)
    {
        box(mianwin_ptr, 0, 20, 160, 160, 0) ;

        dis_mod_set(mianwin_ptr , HZ_16DZ , ASC_12DZ) ;
        printf_str(mianwin_ptr, 5, 30, cptr, 1) ;
        line(mianwin_ptr , 5 , 46, 155, 46, 1) ;

        printf_str(mianwin_ptr, 50, 50, cptr2, 1) ;
        printf_str(mianwin_ptr, 50 , 122, cptr1, 1) ;
        line(mianwin_ptr , 5 , 140, 155, 140, 1) ;
    }
}

```



```

dis_mod_set(mianwin_ptr ,HZ_12DZ ,ASC_12DZ) ;
printf_str(mianwin_ptr, 90 ,146, cptr5, 1) ;

list_show(mianwin_ptr ,lptr) ;
lcd_160_upd() ;

ret = list_key_manege(mianwin_ptr, lptr, lcd_160_upd) ;
if(ret < 0)
{
    list_exit(lptr) ;
    return ;
}
else if(0 == ret)
{
    dec_val_manege() ;
}
else if(1 == ret)
{
    read_val_manege() ;
}
}
}

```

输入框控件

15、input_init

函数原型	input_dst * input_init(uint8 mod)
参数说明	mod: 输入框模式 enum INPUT_MOD { LITTLE_MONEY_MOD , //小金额模式 FREE_MOD, //自由输入模式 PASSWORD_MOD //密码输入模式 } ;
返回	成功返回新控件指针，失败返回 NULL
功能说明	初始化一个输入框，返回输入框控件
实例	input_dst *iptr ; iptr = input_init(0) ;

16、input_property_set

函数原型	int input_property_set(input_dst *ptr ,int comm , ...)
参数说明	<p>ptr 输入框</p> <p>Comm. 命令</p> <p>enum INPUT_PROPER</p> <p>{</p> <p> INPUT_HIN_STA , //提示信息开始位置</p> <p> INPUT_TYPE , //输入的类型</p> <p> INPUT_HIND , //提示信息</p> <p> INPUT_RET_STA , //输入框位置</p> <p> INPUT_RET_SIZ , //输入框大小</p> <p> INPUT_HIND_FONT , //提示信息字体</p> <p> INPUT_MAX_LEN //最大输入文本长度（最多 15 个字）</p> <p>} ;</p> <p>... 参数</p>
返回	<p>0: 成功</p> <p>-1: 失败</p>
功能说明	设置输入框属性
实例	<pre>input_dst *iptr ; iptr = input_property_set (ptr1, INPUT_HIND , “充值金额”) ;</pre>

17、input_props_set

函数原型	void input_props_set(input_dst *ptr , proptey *pptr)
参数说明	<p>ptr 输入框控件</p> <p>pptr 参数列表</p>
返回	成功返回新控件指针，失败返回 NULL
功能说明	设置输入控件的多个参数
实例	input_props_set(iptr_id, iptr_id_proty) ;

18、input_dis

函数原型	int32 input_dis(dis_map *dptr , input_dst *ptr)
参数说明	<p>dptr 显示设备</p> <p>Ptr 输入框控件</p>
返回	成功返回新控件指针，失败返回 NULL

功能说明	将一个输入框跟新到显示内存
实例	input_dis(mianwin_ptr, iptr) ;

19、input_key_manege

函数原型	int32 input_key_manege(dis_map *dptr, input_dst *ptr, int32(*funp)(void))
参数说明	dptr 显示设备 Ptr 输入控件 Funp 跟新方式
返回	0 回车确认 -1 取消按键
功能说明	让输入控件，管理键盘消息
实例	<pre>ret = input_key_manege(mianwin_ptr, iptr, lcd_160_upd) ; if(ret < 0) { return ; } else { ... }</pre>

20、input_val_get

函数原型	uint32 input_val_get(input_dst *ptr)
参数说明	Ptr: 输入控件
返回	获取的数值
功能说明	获得小金额模式下的输入数值
实例	val = input_val_get(iptr) ;

21、input_str_get

函数原型	void input_str_get(input_dst *ptr, uint8 *buf)
------	--

参数说明	ptr 输入控件 buf 输出指针
返回	无
功能说明	获得其他模式下的输入字符串
实例	input_str_get(iptr_id, buf) ;

22、input_destory

函数原型	void input_destory(input_dst *ptr)
参数说明	ptr 输入控件
返回	无
功能说明	销毁一个输入控件内存及数据
实例	input_destory(ptr);

综合示例 1

```

input_dst *iptr ;
iptr = input_init(0) ;
if(NULL == iptr)
{
    exit(1) ;
}
input_dis(mianwin_ptr, iptr) ;
lcd_160_upd() ;
ret = input_key_manege(mianwin_ptr, iptr , lcd_160_upd) ;
if(ret < 0)
{
    return ;
}
else
{
    ....
}

```

综合示例 2

```

proptey    iptr_id_proty[] =
{
    {INPUT_HIN_STA , 2 , 85},
    {INPUT_RET_STA , 75, 83},
    {INPUT_RET_SIZ , 17 ,78},

```

```

        {INPUT_HIND_FONT ,    HZ_16DZ , ASC_12DZ},
        {INPUT_MAX_LEN ,   10 , 0},
        {0xffff ,          0 , 0}
    } ;

proptey    iptr_pass_proty[] =
{
    {INPUT_HIN_STA ,   2 , 110},
    {INPUT_RET_STA ,   75, 108},
    {INPUT_RET_SIZ ,   17 ,78},
    {INPUT_HIND_FONT ,    HZ_16DZ , ASC_12DZ},
    {INPUT_MAX_LEN ,   10 , 0},
    {0xffff ,          0 , 0}
} ;

void test_input_new(void)
{
    input_dst *iptr_id ;
    input_dst *iptr_pass ;
    int32      ret ;
    uint32     val ;
    uint8      buf[32] ;

    box(mianwin_ptr, 0,20,160,160, 0) ;
    dis_mod_set(mianwin_ptr ,HZ_16DZ ,ASC_12DZ) ;
    printf_str(mianwin_ptr, 50, 30, "签到管理", 1) ;
    printf_str(mianwin_ptr, 0 ,55, "身份验证", 1) ;
    line(mianwin_ptr , 5 ,47,155,47,1) ;

    dis_mod_set(mianwin_ptr ,HZ_12DZ ,ASC_12DZ) ;
    printf_str(mianwin_ptr, 90 ,146, "深圳德卡科技", 1) ;
    line(mianwin_ptr , 5 ,140,155,140,1) ;

    iptr_id = input_init(FREE_MOD) ;
    if(NULL == iptr_id)
    {
        exit(1) ;
    }

    iptr_pass = input_init(PASSWORD_MOD) ;
    if(NULL == iptr_pass)
    {
        exit(1) ;
    }
}

```

```

input_props_set(iptr_id, iptr_id_proty) ;
input_props_set(iptr_pass, iptr_pass_proty) ;
input_property_set(iptr_id, INPUT_HIND, "员工代码:") ;
input_property_set(iptr_pass, INPUT_HIND, "密 码:") ;

input_dis(mianwin_ptr, iptr_id) ;
input_dis(mianwin_ptr, iptr_pass) ;
lcd_160_upd() ;

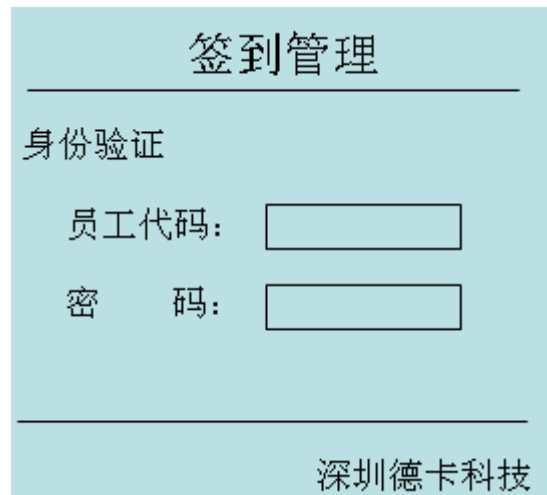
ret = input_key_manege(mianwin_ptr, iptr_id , lcd_160_upd) ;
ret = input_key_manege(mianwin_ptr, iptr_pass ,
lcd_160_upd) ;

input_str_get(iptr_id, buf) ;
printf("buf id %s \n" , buf) ;
input_str_get(iptr_pass, buf) ;
printf("buf pass %s \n" , buf) ;

input_destory(iptr_id) ;
input_destory(iptr_pass) ;
}

```

显示如下效果：



标题控件

23、label_init

函数原型	label_dst *label_init(void);
参数说明	无

返回	成功返回新控件指针，失败返回 NULL
功能说明	初始化一个标题控件
实例	<pre> if(NULL == (timlab_ptr = label_init())) { exit(1) ; } </pre>

24、label_property_set

函数原型	int label_property_set(label_dst *ptr ,int comm , ...)
参数说明	<p>ptr 列表框</p> <p>Comm. 命令</p> <p>enum DIS_STA //位置参数</p> <pre> { STA_LEFT , //居左 STA_RIGHT , //居右 STA_CENTER , //居中 }; </pre> <p>enum DIS_LABMOD //显示模式</p> <pre> { DIS_NORMAL , //正常 DIS_BACKCOLOR , //反显 DIS_BACKDIS , //反显 DIS_ADD , //彩色液晶参数 }; </pre> <p>enum LBL_PROPER</p> <pre> { LBL_STA_STATION , //开始位置 LBL_STATE , //显示对齐位置 LBL_WIDTH , //宽 LBL_FON_COLOR , //字体颜色（彩色液晶） LBL_BAK_COLOR , //背景颜色（彩色液晶） LBL_FONT_TYPE , //字体设置 LBL_DIS_MOD , //显示模式 LBL_DIS_CAPTION , //待显示的标题 }; </pre> <p>... 参数</p>
返回	<p>0: 成功</p> <p>-1: 失败</p>

功能说明	设置一个标题控件显示属性
实例	list_property_set (ptr1, LABL_STATE , STA_CENTER) ; list_property_set (ptr1, LABL_DIS_CAPTION, “系统功能”) ;

25、label_props_set

函数原型	void label_props_set(label_dst *ptr , propkey *pptr)
参数说明	ptr 标题控件 pptr 属性列表
返回	无
功能说明	标题控件多属性设置
实例	propkey tim_lab_proty[] = { {LABL_STA_STATION , 80 , 3}, //开始位置 {LABL_STATE , STA_RIGHT , 0}, //显示对齐位置 {LABL_WIDTH , 80 , 10}, //宽 {LABL_FONT_TYPE , HZ_12DZ , ASC_8DZ}, //字体设置 {0xffff , 0 , 0} } ; label_props_set(timlab_ptr, tim_lab_proty) ; label_property_set(timlab_ptr, LABL_DIS_CAPTION, 19:05/2009-02-19”) ;

26、label_dis

函数原型	void label_dis(dis_map *dptr , label_dst *ptr)
参数说明	dptr 显示设备 ptr 标题控件
返回	无
功能说明	将一个标题控件更新到显示设备内存
实例	label_dis(hindwin_ptr, timlab_ptr) ; lcd_160_upd() ;

27、label_destory

函数原型	Void label_destory(label_dst *ptr)
------	------------------------------------

参数说明	ptr 标题控件
返回	无
功能说明	销毁一个标题控件内存及数据
实例	label_destory(lptra) ；

综合示例：液晶右上角时间显示

```

propety    tim_lab_proty[] =
{
    {LABL_STA_STATION , 80 , 3},//开始位置
    {LABL_STATE ,      STA_RIGHT , 0},//对齐位置
    {LABL_WIDTH ,      80 , 10},//宽度
    {LABL_FONT_TYPE ,  HZ_12DZ , ASC_8DZ},//字体设置
    {0xffff ,          0 , 0}
} ;

void *hind_thread(void *sig)
{
    struct          pic_lst *hind_icon_ptr ;
    dis_map          *hindwin_ptr = NULL ;
    label_dst *timlab_ptr ;
    time_t           timep;
    struct            tm *p;
    uint8            disbuf[32] ;
    uint32           i = 0;

    sig = sig ;

    hindwin_ptr = ds_init(160,20,BIT_4) ;
    if(NULL == hindwin_ptr)
    {
        printf("ds init error \n") ;
        exit(1) ;
    }

    if(NULL == (timlab_ptr = label_init()))
    {
        exit(1) ;
    }
    label_props_set(timlab_ptr, tim_lab_proty) ; //设置位置其他参数
值
    for(;;)
    {

```

```

time(&timep);          //获得时间
p = gmtime(&timep);    //将时间转变为闪烁的字符串
if(i%2)
{
    sprintf(disbuf, "%d-%d-%d :%d", (1900+p->tm_year), (1+p->t
    m_mon), (p->tm_mday), p->tm_hour, p->tm_min) ;
}
else
{
    sprintf(disbuf, "%d-%d-%d%d%d", (1900+p->tm_year), (1+p->t
    m_mon), (p->tm_mday), p->tm_hour, p->tm_min) ;
}

label_property_set(timlab_ptr, LABL_DIS_CAPTION, disbuf) ;//
label_dis(hindwin_ptr, timlab_ptr) ; //添加到显示

lcd_160_upd() ;//显示更新
i++ ;
sleep(1);    //延时 1 秒
}
label_destory(timlab_ptr) ;
ds_destory(hindwin_ptr) ;
}

```

选择对话框

28、choise_init

函数原型	choise_dst *choise_init(uint8 mod)
参数说明	3. 确认和取消选择 1. 重试和取消选择
返回	成功返回新控件指针，失败返回 NULL
功能说明	初始化一个选择对话框控件
实例	<pre> ptr = choise_init(1) ; if(NULL == ptr) { exit(1) ; } </pre>

29、choice_property_set

函数原型	int choice_property_set(choise_dst *ptr ,int comm , ...)
参数说明	ptr 控件指针 comm. 命令 enum CHO_PROPER { CHO_STA, //开始位置 CHO_MOD , //显示模式 CHO_FON_TYPE , //字体 CHO_WIDTH , //宽高 CHO_HIND , //提示信息 } ;
返回	成功返回新控件指针，失败返回 NULL
功能说明	设置选择对话框显示属性
实例	choice_property_set(ptr, CHO_STA, 10,10)

30、choice_props_set

函数原型	void choice_props_set(choise_dst *ptr , proptey *pptr)
参数说明	ptr 控件指针 pptr. 属性表
返回	无
功能说明	同时设置选择控件显示属性
实例	choice_props_set(ptr , proptey-lable);

31、choise_key_manege

函数原型	int32 choise_key_manege(dis_map *dptr ,choise_dst *ptr, int32(*funp) (void))
参数说明	dptr 显示设备 Ptr 选择控件指针 funp
返回	0 回车确认 -1 取消按键

功能说明	让选择控件，管理键盘消息
实例	ret = choise_key_manege(mianwin_ptr, ptr, lcd_160_upd) ;

32、choise_destory

函数原型	void choise_destory(choise_dst *ptr)
参数说明	ptr 选择控件指针
返回	无
功能说明	销毁一个选择对话控件控件内存及数据
实例	choise_destory (lptr) ;

其他

33、beep

函数原型	void beep(uint32 dtim)
参数说明	uint32 dtim 10*dtim ms 时间 蜂鸣
返回	无
功能说明	蜂鸣器函数
实例	beep(10);

34、led_updat

函数原型	int32 led_updat(uint8 *ptr)
参数说明	ptr 12 字节的缓冲区
返回	无
功能说明	将 led 缓冲区的数据输出到 LED 屏上
实例	dis_buf_tran(led_buf, 6) ; led_updat(led_buf) ;

35、dis_buf_tran

函数原型	void dis_buf_tran(uint8 *ptr , uint8 len)
参数说明	ptr 字符串 Len 长度
返回	无
功能说明	led 显示转换缓冲区内容
实例	dis_buf_tran(led_buf, 6) ; led_updat(led_buf) ;

PS: static uint8 led_tran_tab[] = {

```

        '0', 0x3f, '1', 0x06, '2', 0x5b, '3', 0x4f, '4', 0x66,
        '5', 0x6d, '6', 0x7d, '7', 0x07,
        '8', 0x7f, '9', 0x6f, 'A', 0x77, 'b', 0x7c, 'c', 0x58,
        'd', 0x5e, 'E', 0x79, 'F', 0x71,
        ' ', 0x00, '-', 0x40, '=', 0x41, '[', 0x39, ']', 0x0f,
        '_', 0x08, 'H', 0x76, 'L', 0x38,
        'P', 0x73, 'C', 0x39, 'r', 0x72, 'o', 0x5c, 'W', 0x00
    } ;

```

说明：大于 0x80 的数据，转换为字符加小数点，转换表如上

第4章 卡类函数说明

类型定义说明：

```

typedef signed char      int8;
typedef unsigned char    uint8;

typedef signed short     int16;
typedef unsigned short   uint16;

typedef signed int       int32;
typedef unsigned int     uint32;

typedef signed long long int64;
typedef unsigned long long uint64;

```

射频卡

1、dc_init_rf

函数原型	int32 dc_init_rf (void)
参数说明	无
返回	成功则返回句柄 -1: 失败
功能说明	初始化射频模块，才可进行后续射频操作
实例	<pre>dev = dc_init_rf(); if(dev<0) { printf("open rf53l_open error \n") ; return(-1) ; }</pre>

2、dc_exit_rf

函数原型	void dc_exit_rf(int fd)
参数说明	int fd 句柄
返回	无
功能说明	关闭射频模块
实例	<pre>dc_exit_rf(fd);</pre>

3、dc_reset

函数原型	void dc_reset(dev,int mon);
参数说明	dev 射频模块句柄 mon 0 关闭; 1 打开
返回	无
功能说明	复位射频模块 连续寻卡是必须调用
实例	<pre>dc_reset(dev, 0); dc_reset(dev, 1);</pre>

4、dc_config_card

函数原型	int dc_config_card(int fd,unsigned char cardtype)
------	---

参数说明	fd: 打开端口返回的句柄; cardtype :卡类型, 'A' TYPEA 卡 'B' TYPEB 卡
返回	无
功能说明	设置卡类型
实例	dc_config_card (dev,'A');

5、dc_request

函数原型	int32 dc_request(int fd,uint8 _Mode , uint8 *TagType)
参数说明	_Mode: 寻卡模式 0——表示 IDLE 模式, 一次只对一张卡操作; Tagtype: 卡类型值 返回卡类型, MIFARE 1 特征值 4
返回	0: 成功 其他失败
功能说明	寻卡请求
实例	dc_request(dev, 0, type);

6、dc_anticoll

函数原型	int32 dc_anticoll(int fd,uint8 _Bcnt , uint32 *_Snr)
参数说明	fd: 打开端口返回的句柄 _Bcn: 设为 0 _Snr: 返回的卡序列号地址
返回	0: 成功 其他失败
功能说明	防卡冲突, 返回卡的序列号
实例	dc_anticoll(fd, 0 , _Snr)

7、dc_select

函数原型	int32 dc_select(int fd,uint32 _Snr , uint8 *_Size)
参数说明	fd: 打开端口返回的句柄 _Snr: 卡序列号 _Size: 指向返回的卡容量的数据

返回	0: 成功 其他失败
功能说明	选取一个给定序列号的卡
实例	dc_anticoll(fd, _Snr, _Size);

8、dc_card

函数原型	int32 dc_select(int fd,uint32 _Snr , uint8 *_Size)
参数说明	fd: 打开端口返回的句柄 _Mode: 寻卡模式 _Snr: 返回的卡序列号
返回	0: 成功 其他失败
功能说明	寻卡，能返回在工作区域内某张卡的序列号（该函数包含了 dc_request,dc_anticoll,dc_select 的整体功能）
实例	st=dc_card(dev, 0, &cardsnr);

PS：射频卡寻卡部分 M1 卡，CPU 卡通用
M1 卡

9、dc_authentication_pass

函数原型	int32 dc_authentication_pass(int fd,uint8 _Mode , uint8 addr, uint8 *password)
参数说明	fd: 打开端口返回的句柄 _Mode: 密码验证模式 Addr: 要验证密码的扇区块号（扇区号） passbuff:密码字符串
返回	0: 成功 其他失败
功能说明	验证 M1 卡密码
实例	st= dc_authentication_pass(dev,0, 0, pass) ;

PS：密码验证模式

对于 M1 卡的每个扇区，在读写器中均对应有三套密码（KEYSET0、KEYSET1、KEYSET2），每套密码包括 A 密码（KEYA）和 B 密码（KEYB），共六个密码，用 0～2、4～6 来表示这六个密码：

- 0——KEYSET0 的 KEYA
- 1——KEYSET1 的 KEYA
- 2——KEYSET2 的 KEYA
- 4——KEYSET0 的 KEYB
- 5——KEYSET1 的 KEYB
- 6——KEYSET2 的 KEYB

10、dc_readval

函数原型	int32 dc_readval(int fd,uint8 _Adr , uint32 *_Value)
参数说明	fd: 打开端口返回的句柄 _Adr: 块地址 _Value: 读出值
返回	0: 成功 其他失败
功能说明	读块值
实例	st= dc_authentication_pass(dev,0, 0, pass) ;

11、dc_decrement

函数原型	int32 dc_decrement(int fd,uint8 _Adr , uint32 _Value)
参数说明	fd : 打开端口返回的句柄 _Adr: 块地址 _Value: 要减的值
返回	0: 成功 其他失败
功能说明	块减值
实例	dc_decrement(dev,4 , decval) ;

12、dc_increment

函数原型	int dc_increment(int fd,unsigned char _Adr, unsigned long _Value)
参数说明	fd : 打开端口返回的句柄 _Adr: 块地址 _Value: 要加的值

返回	0: 成功 其他失败
功能说明	块加值
实例	dc_increment (dev,4 , decval) ;

13、dc_initval

函数原型	int32 dc_initval(int fd,uint8 _Adr , uint32 *_Value)
参数说明	fd: 打开端口返回的句柄 _Adr: 块地址 _Value: 初始值
返回	0: 成功 其他失败
功能说明	初始化块值
实例	dc_initval(int fd,4 , 1000);

14、dc_restore

函数原型	int32 dc_restore(int fd,uint8 _Adr1 , uint8 _Adr2)
参数说明	fd: 打开端口返回的句柄 _Adr: 原块地址 _Adr2: 目标块地址
返回	0: 成功 其他失败
功能说明	将块 1 的内容传送到块 2
实例	dc_restore (int fd,4 , 8)

15、dc_write

函数原型	int32 dc_write(int fd,uint32 n , uint8 databuff)
参数说明	fd: 打开端口返回的句柄 n: 绝对块号 (0-63) databuff : 要写入的数据
返回	0: 成功

	其他失败
功能说明	写数据
实例	st = dc_write(dev, j, wrData);

16、dc_read

函数原型	int32 dc_read (int fd, uint32 n , uint8 databuff)
参数说明	fd: 打开端口返回的句柄 n: 绝对块号 (0-63) databuff : 读出来的数据
返回	0: 成功 其他失败
功能说明	读数据
实例	st = dc_read (dev, j, reData);

非接触式 CPU 卡

17、dc_pro_reset

函数原型	int dc_pro_reset(int fd, unsigned char *rlen, unsigned char* buff)
参数说明	int fd 打开端口返回的句柄 unsigned char *rlen 返回复位信息的长度 unsigned char * rbuff 存放返回的复位信息
返回	0: 成功 <0 错误, 其绝对值为错误号
功能说明	卡复位函数
实例	dc_pro_reset(fd, &retlen, rdata);

18、dc_pro_command

函数原型	int dc_pro_command(int fd, unsigned int slen, unsigned char *sdata, unsigned int *rlen, unsigned char* rbuf, unsigned char timeout)
参数说明	int fd 打开端口返回的句柄

	unsigned char slen 发送的信息长度 unsigned char * sdata 存放要发送的信息 unsigned char *rlen 返回信息的长度 unsigned char * rbuf 存放返回的信息 unsigned char timeout 延迟时间，单位为：10ms
返回	0：成功 <0 错误，其绝对值为错误号
功能说明	应用协议数据单元信息交换函数
实例	<pre> slen=5; sdata[0]=0x00; sdata[1]=0x84; sdata[2]=0x00; sdata[3]=0x00; sdata[4]=0x08; st=dc_pro_command(fd, slen, sdata, &rlen, rdata1, 100); </pre>

接触式 CPU 卡

19、dc_int_sam

函数原型	int dc_int_sam(void)
参数说明	无
返回	成功则返回句柄 -1：失败
功能说明	打开端口，返回操作句柄
实例	<pre> icdev=dc_int_sam(); if(icdev<0) { printf("dc_int_sam error. %d\n", icdev); return -1; } </pre>

20、dc_setcpu

函数原型	int dc_setcpu(int fd, unsigned char _Byte)
参数说明	fd：端口句柄 _Byte：设置要操作的卡座号, 0x0c 为大卡座，0x0d 0x0e 0x0f 各为 SAM1 SAM2 SAM3
返回	0：成功 <0 错误

功能说明	设置当前卡座
实例	dc_setcpu(icdev, 0x0c);

21、dc_setcpupara

函数原型	int dc_setcpupara(int fd,unsigned char cputype,unsigned char cpupro,unsigned char cputetu)
参数说明	fd 端口句柄 unsigned char cputype ---- 卡座类型, 12=主卡座 13=SAM1 卡座 14=SAM2 卡座 15=SAM3 卡座 (十进制) unsigned char cpupro 卡的协议类型 =0 表示 T=0 协议 =1 表示 T=1 协议 unsigned char cputetu 卡操作时候的延时数据 (十进制) 不同波特率的卡, 此参数的值不同, 对于 9600 波特率的卡 cputetu 设置成 0x11 对于 38400 波特率的卡 cputetu 设置成 0x13
返回	0: 成功 <0 错误, 绝对值为错误号
功能说明	设置 CPU 卡的参数, 上电后的默认参数是 cpupro=0(T=0 协议) cputetu=0x11(波特率 9600)
实例	st= dc_setcpupara(icdev, 0x0c, 0, 0x11);

22、dc_cpureset

函数原型	int dc_cpureset(int fd,unsigned char *rlen,unsigned char *databuffer)
参数说明	int fd 端口句柄 unsigned char *rlen 返回复位信息的长度 unsigned char * databuffer 存放返回的复位信息
返回	0: 成功 <0 错误, 绝对值为错误号
功能说明	CPU 卡上电复位函数, 复位后自动判断卡片协议
实例	dc_cpureset(icdev, &retlen, data);

23、dc_cpuapdu

函数原型	int dc_cpuapdu(int fd,unsigned int slen,unsigned char * sendbuffer,unsigned int *rlen,unsigned char * databuffer)
参数说明	int fd 端口句柄 unsigned int slen 发送的信息长度 unsigned char * sendbuffer 存放要发送的信息 unsigned int *rlen 返回信息的长度 unsigned char * databuffer 存放返回的信息
返回	0: 成功 <0 错误, 绝对值为错误号
功能说明	CPU 卡 APDU (应用协议数据单元) 信息交换函数
实例	dc_cpureset(icdev,&retlen,data);

24、dc_exit_sam

函数原型	int dc_exit_sam(int fd)
参数说明	int fd 端口句柄
返回	无
功能说明	关闭 CPU 卡端口句柄
实例	<pre> st=dc_cpureset(icdev,&retlen,data); if(st!=0) { dc_exit_sam(icdev); return -11; } </pre>

磁条卡

25、maginit

函数原型	int maginit(void)
参数说明	无
返回	返回: <0 失败 成功返回句柄
功能说明	初始化磁条卡

实例	fd=maginit();
----	---------------

26、magdata

函数原型	int magdata(int fd,unsigned char* ver,unsigned char* track1,unsigned char* track2,unsigned char* track3)
参数说明	track1 返回一轨数据; track2 返回二轨数据; track3 返回三轨数据;
返回	0: 成功 -1: 失败
功能说明	获取磁条卡数据
实例	fd=maginit();

第5章 GPRS 模块函数说明

1、gprs_set_baud

函数原型	uchar gprs_set_baud(uint baudsel)
参数说明	baudsel: 波特率, 115200、57600、38400、19200、14400、9600、4800、2400、1200、600、300
返回	0: 波特率设置成功 1: 波特率设置失败
功能说明 实例	设置通讯波特率, 默认波特率为 9600 uchar temp; uint baud=115200; temp= gprs_set_baud (baud); //设置当前 GPRS 模块通讯波特率为 115200

2、gprs_test

函数原型	uchar gprs_test(void)
参数说明	无
返回	0: 测试成功 1: 测试失败 2: 超时退出

功能说明	AT 指令测试，发送命令“AT”，当返回“OK”时，表示当前模块可以正常通讯
实例	<pre> uchar temp; temp= gprs_test(); // AT 指令测试 </pre>

3、gprs_close_return

函数原型	uchar gprs_close_return(uchar mode)
参数说明	mode=0: 关闭回显 1: 开启回显
返回	0: 回显方式开启/关闭成功 1: 回显方式开启/关闭失败 2: 超时退出
功能说明 实例	回显方式开启/关闭 <pre> uchar mode=0; uchar temp; temp= gprs_close_return(mode); //关闭回显 </pre>

4、gprs_sleep

函数原型	uchar gprs_sleep(uchar mode)
参数说明	mode=0: 休眠模式关闭 1: 休眠模式开启
返回	0: 模块休眠模式设置成功 1: 模块休眠模式设置失败 2: 超时退出
功能说明 实例	设置模块是否休眠状态 <pre> uchar mode=0; uchar temp; temp= gprs_sleep (mode); //关闭休眠模式 </pre>

5、gprs_check_signal

函数原型	uchar gprs_check_signal(uchar *signal)
参数说明	signal: 信号强度，0~31
返回	0: 检测信号强度成功 1: 检测信号强度失败 2: 超时退出
功能说明	检测信号强度

实例	<pre> uchar sig; uchar temp; temp= gprs_check_signal(&sig); //检测当前环境的信号强度 </pre>
----	--

6、gprs_check_version

函数原型	uchar gprs_check_version(uchar *hversion, uchar *sversion)
参数说明	hversion: 硬件版本号 sversion: 软件版本号
返回	0: 检测模块版本号成功 1: 检测模块版本号失败 2: 超时退出
功能说明 实例	检测模块版本号 <pre> uchar hver; uchar sver[10]; uchar temp; temp= gprs_check_version(&hver, sver); //检测模块软硬件版本号 </pre>

7、gprs_close

函数原型	void gprs_close(void)
参数说明	无
返回	无
功能说明 实例	关闭 GPRS 模块 <pre> gprs_close(); //GPRS 模块直接下电 </pre>

8、gprs_set_function

函数原型	uchar gprs_set_function(uchar mode)
参数说明	mode=0: 最小功能, 先注销网络, 之后去激活 SIM 卡 1: 最大功能, 首先激活 SIM 卡, 之后进行自动搜网 4: 禁用手机发送和接收 RF 电路
返回	0: 功能设置成功 1: 功能设置失败 2: 超时退出
功能说明 实例	功能设置 <pre> uchar mode=0; uchar temp; temp= gprs_set_function(mode); </pre>

	//设置模块为最小功能模式，此模式下功耗很小。
--	-------------------------

9、gprs_init_apn

函数原型	uchar gprs_init_apn(void)
参数说明	无
返回	0: 初始化 PDP 成功 1: 初始化 PDP 失败 2: 超时退出
功能说明 实例	初始化 PDP 激活参数，（公网时使用） uchar temp; temp= gprs_init_apn(); //初始化 PDP，确认模块开机并搜网成功

10、gprs_init_tcpip

函数原型	uchar gprs_init_tcpip(void)
参数说明	无
返回	0: 初始化 TCPIP 成功 1: 初始化 TCPIP 失败 2: 超时退出
功能说明 实例	初始化 TCPIP（公网时使用） uchar temp; temp= gprs_init_tcpip(); //初始化 TCPIP

11、gprs_open_tcp

函数原型	uchar gprs_open_tcp(uchar *ip, uchar iplen, uchar *port, uchar portlen)
参数说明	ip: IP 地址，ASCII 码格式，如： ip[15]={ '1', '2', '1', '.', '3', '5', '.', '1', '2', '8', '.', '7', '4' } iplen: IP 地址长度。 port: 端口号，ASCII 码格式 portlen: 端口号长度
返回	0: 打开 TCP 链接成功 1: 打开 TCP 链接失败 2: 超时退出
功能说明 实例	打开一条 TCP 链接 uchar ip[15]={ '1', '2', '1', '.', '3', '5', '.', '1', '2', '8', '.', '7', '4' };

	<pre> uchar iplen=15; uchar port[4]={ '5','6','6','6'}; uchar portlen=4; uchar temp; temp= gprs_open_tcp(ip, iplen, port, portlen); //打开 IP 地址为 121.35.128.74 端口号为 5666 的服务器 </pre>
--	---

12、gprs_querfy_data

函数原型	uchar gprs_querfy_data(uint *data_len)
参数说明	data_len: 缓冲区中数据长度
返回	0: 查询缓冲区数据成功 1: 查询缓冲区数据失败 2: 超时退出
功能说明 实例	查询缓冲区剩余数据 <pre> uint len; uchar temp; temp= gprs_querfy_data(&len); //查询缓冲区剩余数据 </pre>

13、gprs_set_datadeletemode

函数原型	uchar gprs_set_datadeletemode(uchar mode)
参数说明	mode=0: 自动模式, 读取第一个未读数据包然后再自动删除它 mode=1: 手动模式, 读取索引为 index 的数据包 (无论是已读还是未读) 然后再自动删除它
返回	0: 删除模式设置成功 1: 删除模式设置失败 2: 超时退出
功能说明 实例	数据删除模式设置 <pre> uchar mode=0; uchar temp; temp= gprs_set_datadeletemode(mode); //设置数据删除模式为自动模式 </pre>

14、gprs_set_datamode

函数原型	uchar gprs_set_datamode(uchar mode)
参数说明	mode = 0: 模式 0, 不需要对输入的数据编码 1: 模式 1, 需要对输入的数据编码
返回	0: 数据模式设置成功 1: 数据模式设置失败

	2: 超时退出
功能说明 实例	设置数据模式 uchar mode=1; uchar temp; temp= gprs_set_datamode(mode); //设置数据模式为编码模式

15、gprs_send_tcpdata

函数原型	uchar gprs_send_tcpdata(char *send_buff, uint send_len)
参数说明	send_buff: 发送数据缓冲区 send_len: 发送数据长度
返回	0: 数据发送成功 1: 数据发送失败 2: 超时退出
功能说明 实例	通过 TCPIP 模式发送一定长度数据 非编码模式: char s_buff[3]={ 'b' , 'a' , 'i' }; uint s_len=3; uchar temp; temp= gprs_send_tcpdata(s_buff, s_len); //通过 TCPIP 发送 3 个数据 “bai” 编码模式: char s_buff[4]={ '3' , '4' , '3' , '5' }; uint s_len=4; uchar temp; temp= gprs_send_tcpdata(s_buff, s_len); //通过 TCPIP 发送 2 个数据 “45”

16、gprs_receive_tcpdata

函数原型	uchar gprs_receive_tcpdata(uchar *data_buff, uint *data_len)
参数说明	data_buff: 接收数据缓冲 data_len : 接收数据长度
返回	0: 接收数据成功 1: 接收数据失败 2: 超时退出
功能说明 实例	接收缓冲区数据 uchar r_buff[200]; uint r_len; uchar temp; temp= gprs_receive_tcpdata(r_buff, r_len);

	//接收缓冲区数据
--	-----------

17、gprs_close_tcp

函数原型	uchar gprs_close_tcp(void)
参数说明	无
返回	0: 关闭链接成功 1: 关闭链接失败 2: 超时退出
功能说明 实例	关闭链接 uchar temp; temp= gprs_close_tcp(); //关闭 TCP 链接

18、modem_gprs_sem_init

函数原型	Void modem_gprs_sem_init (void)
参数说明	无
返回	
功能说明 实例	GPRS 通讯通道初始化 modem_gprs_sem_init();

19、gprs_sel_sem_wait

函数原型	Void gprs_sel_sem_wait (void)
参数说明	无
返回	
功能说明 实例	GPRS 申请通讯通道 gprs_sel_sem_wait ();

20、modem_gprs_sem_end

函数原型	Void modem_gprs_sem_end (void)
参数说明	无

返回	
功能说明	GPRS 关闭通讯通道
实例	modem_gprs_sem_end ();

21、gprs_initapn

函数原型	gprs_initapn(char *apn_buff)
参数说明	apn_buff: 字符型 ANP
返回	0: 初始化 PDP 激活参数成功 1: 初始化 PDP 激活参数失败
功能说明	设置 APN 接入点
实例	Int temp; uchar apn_buff[6]={ 'C', 'M', 'N', 'E', 'T' }; temp = gprs_initapn(apn_buff);

22、gprs_inittcpip

函数原型	gprs_inittcpip(uchar *user, uchar *pword)
参数说明	User: APN 的用户名 Pword: APN 的密码
返回	0: 初始化 TCP/IP 成功 1: 初始化 TCP/IP 失败
功能说明	初始化 TCP/IP
实例	uchar user[7]={0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; uchar pword[7]={0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00}; int temp; temp = gprs_inittcpip(user, pword);

23、gprs_buff_check

函数原型	uchar gprs_buff_check(uint *num)
参数说明	num: 缓冲区未读数据包个数
返回	0: 查询成功 1: 查询失败
功能说明	查询缓冲区未读数据包个数
实例	uint num=0; gprs_buff_check(num);

24、gprs_pin_rst

函数原型	gprs_pin_rst(uint32 val)
参数说明	0: 模块下电 1: 模块上电
返回	无
功能说明	gprs 模块复位
实例	<pre>gprs_pin_rst (0); sleep (1); gprs_pin_rst (1); sleep (2);</pre>

25、power_pin_set

函数原型	power_pin_set(int cg)
参数说明	cg: 为 0 是为 gprs 模块应用, 为 1 是为 cdma 模块应用
返回	无
功能说明	gprs/cdma 模块选用
实例	<pre>power_pin_set(0);</pre>

第6章 CDMA 模块函数说明

1、cdma_close_return

函数原型	uchar cdma_close_return(void)
参数说明	无
返回	0: 关闭成功 1: 关闭失败
功能说明	关闭回显
实例	<pre>uchar temp; temp = cdma_close_return(); //关闭回显</pre>

2、cdma_check_signal

函数原型	uchar cdma_check_signal(uchar *signal_buff, uchar *signal_len)
参数说明	signal_buff: 信号强度数据 signal_len: 信号强度数据长度
返回	0: 检测成功 1: 检测失败
功能说明	检测当地的网络信号强度, 31 最大, 0 最小
实例	<pre>uchar signal[2]; uchar len; uchar temp; temp = cdma_check_signal(signal, &len); //检测信号强度</pre>

3、cdma_check_sim

函数原型	uchar cdma_check_sim(uchar *pin_buff, ushort *pin_len)
参数说明	data_buff: PIN 状态数据 data_len: PIN 状态数据长度
返回	0: 查询成功 1: 查询失败 2: 超时
功能说明	查询 PIN 状态, SIM 卡是否处于就绪状态
实例	<pre>uchar pin[10]; uchar len; uchar temp; temp = cdma_check_sim(pin, &len); //检测当前 SIM 卡状态, 成功时返回 “READY”</pre>

4、cdma_set_user

函数原型	uchar cdma_set_user(uchar *user, uchar userlen, uchar *password, uchar passwordlen)
参数说明	user: 用户名 userlen: 用户名长度 password: 密码 passwordlen: 密码长度
返回	0: 设置成功 1: 设置失败 2: 超时
功能说明	设置 PPP 用户名和口令
实例	<pre>uchar un[4] = ['c', 'a', 'r', 'd'];</pre>

	<pre> uchar pd[4] = ['c' , 'a' , 'r' , 'd']; uchar unlen = 4; uchar pdlen = 4; uchar temp; temp = cdma_set_user(un, unlen, pd, pdlen); //设置用户密码（在未开通时用户名和口令默都认为 card） </pre>
--	--

5、cdma_build_ppp

函数原型	uchar cdma_build_ppp(uchar *status)
参数说明	Status: 连接成功返回的信息 0: 表示连接成功 1: 表示连接失败
返回	0: 连接 PPP 成功 1: 连接 PPP 失败 2: 超时
功能说明	该命令用来建立 PPP 连接，如果在执行该命令之前设置过正确的 PPP 用户名称和口令则可以正常拨号成功，否则拨号失败。
实例	<pre> uchar status; uchar temp; temp = cdma_build_ppp(&status); //建立 PPP 连接 </pre>

6、cdma_closed_ppp

函数原型	uchar cdma_closed_ppp(void)
参数说明	无
返回	0: 关闭 PPP 连接成功 1: 关闭 PPP 连接失败 2: 超时
功能说明	关闭 PPP 连接
实例	<pre> uchar temp; temp = cdma_closed_ppp(); //关闭 PPP 连接 </pre>

7、cdma_inquire_ppp

函数原型	uchar cdma_inquire_ppp(uchar *status)
参数说明	Status: PPP 拨号连接状态 0: 表示连接成功 1: 表示连接失败
返回	0: 查询成功

	1: 查询失败 2: 超时
功能说明	PPP 拨号连接状态查询
实例	<pre> uchar status; uchar temp; temp = cdma_inquire_ppp(&status); //查询当前的 PPP 连接状态 </pre>

8、cdma_build_tcpip

函数原型	uchar cdma_build_tcpip(uchar *ip, uchar iplen, uchar *port, uchar portlen)
参数说明	ip: IP 地址, ASCII 码格式, 如: ip[15]={ '1', '2', '1', '.', '3', '5', '.', '1', '2', '8', '.', '7', '4' } iplen: IP 地址长度。 port: 端口号, ASCII 码格式 portlen: 端口号长度
返回	0: 建立 TCP 链接成功 1: 建立 TCP 链接失败 2: 超时
功能说明	建立一条 TCP 链接
实例	<pre> uchar temp; uchar ip[15]={ '1', '2', '1', '.', '3', '5', '.', '1', '2', '8', '.', '7', '4' }; uchar iplen=13; uchar port[4]= { '5', '3', '3', '3' }; uchar portlen=4; temp = cdma_build_tcpip(Ip, iplen, port, portlen); //链接 ip 地址为 121.35.128.74, 端口号为 5333 的终端。 </pre>

9、cdma_cancel_tcp

函数原型	uchar cdma_cancel_tcp(void)
参数说明	无
返回	0: 关闭成功 1: 关闭失败
功能说明	强行关闭正在建立的 TCP 链接
实例	<pre> uchar temp; temp = cdma_cancel_tcp(); //关闭当前 TCP 链接 </pre>

10、cdma_check_tcp

函数原型	uchar cdma_check_tcp(uchar *data_buff, uchar *data_len)
参数说明	data_buff: 链接状态信息, 0: TCP 连接已成功; 1: TCP 连接已关闭 data_len: 链接状态信息长度
返回	0: 查询成功 1: 查询失败 2: 超时
功能说明	查询当前 TCP 链接状态
实例	uchar buff[20]; uchar len; uchar temp; temp = cdma_check_tcp(buff, &len); //查询当前 TCP 链接状态

11、cdma_close_tcpip

函数原型	uchar cdma_close_tcpip(void)
参数说明	无
返回	0: 关闭链接成功 1: 关闭链接失败
功能说明	关闭 TCP 链接
实例	uchar temp; temp = cdma_close_tcpip(); //关闭 TCP 链接

12、cdma_send_tcpdata

函数原型	uchar cdma_send_tcpdata(uchar *data, ushort data_len)
参数说明	data: 发送数据 data_len: 发送数据长度
返回	0: 发送成功 1: 发送失败
功能说明	发送 TCP 数据
实例	uchar buff[12]={ '0','0','0','0','0','0','0','0','0','0','1','1'}; uchar len=12; uchar temp; temp = cdma_send_tcpdata(buff, len); //发送 12 字节数据

13、cdma_tcp_recv

函数原型	uchar cdma_tcp_recv(uchar *data_buff, ushort *data_len)
参数说明	data_buff: 接收数据 data_len: 接收数据长度
返回	0: 接收成功 1: 接收失败 2: 超时
功能说明	接收 TCP 数据
实例	<pre> uchar buff[512]; ushort len; uchar temp; temp = cdma_tcp_recv(buff, &len); //接收 TCP 数据 </pre>

第7章 TCPIP 通讯模块函数说明

1、connecttcpserver

函数原型	int connecttcpserver(unsigned char * Ser_IP,int Ser_Port)
参数说明	Ser_IP 服务器 IP 地址 Ser_Port 服务器端口号
返回	0: 链接成功 -1: 链接失败
功能说明	TCPIP 链接服务器
实例	<pre> Result = connecttcpserver ("192.168.16.35", 5333); if(Result < 0) { printf("NET_Create error!\n"); dis_mod_set(mianwin_ptr ,HZ_12DZ ,ASC_8DZ) ; printf_str(mianwin_ptr, 2,90, " 以太网测试 creat error!", 1) ; lcd_160_upd() ; goto end; return -1; } </pre>

2、sendtcpmessage

函数原型	int sendtcpmessage(int msglen,unsigned char *msg ,int timeout)
------	--

参数说明	Smsglen 数据长度 msg 数据流 timeout 超时推出时间
返回	0: 发送成功 -1: 发送失败
功能说明	发送数据
实例	<pre> Result = sendtcpmessage (Length,Buffer ,5); if(Result < 0) { { printf("Net_Write, error!\n"); Tpclose(); dis_mod_set(mianwin_ptr ,HZ_12DZ ,ASC_8DZ) ; printf_str(mianwin_ptr, 2,90, " 以太网测试 write error!", 1) ; lcd_160_upd() ; goto end; return -1; } </pre>

3、recvtcpmessage

函数原型	int recvtcpmessage(unsigned char *msg, int timeout)
参数说明	msg 接收到的数据 timeout 超时退出时间
返回	数据长度: 接收成功 -1: 接收失败
功能说明	接收数据
实例	<pre> Result = recvtcpmessage (Buffer, 5); if(Result < 0) { printf("Net_Read, error!\n"); Tpclose(); dis_mod_set(mianwin_ptr ,HZ_12DZ ,ASC_8DZ) ; printf_str(mianwin_ptr, 2,90, " 以太网测试 read error!", 1) ; lcd_160_upd() ; goto end; return -1; } </pre>

4、tcpclose

函数原型	void tcpclose(void)
------	---------------------

参数说明	无
返回	无
功能说明	关闭 TCP/IP
实例	tcpclose();

第8章 常用功能函数

1、fun_hextoascii

函数原型	char fun_hextoascii(uchar hex_data)
参数说明	hex_data: 十六进制数据
返回	0: 十六进制数据超出范围, 转换失败 其他: 转换后数据
功能说明	将小于 16 的十六进制数据转换为大写 ASCII 码
实例	uchar h_data=0x09; char a_data; a_data = fun_hextoascii(h_data); //将十六进制数据 0x09 转换为 ASCII 码 '9'

2、fun_asciitohex

函数原型	uchar fun_asciitohex(char ascii_data)
参数说明	ascii_data: ASCII 码数据
返回	0xFF: ASCII 码数据格式错误, 转换失败 其他: 转换后数据
功能说明	将 ASCII 码转换为十六进制数据
实例	char a_data=' A' ; uchar h_data; h_data= fun_asciitohex(a_data); //将字符 ' A' 转换为十六进制数据 0x0A;

3、fun_clear_buff

函数原型	void fun_clear_buff(uchar *buff, uint bufflen)
参数说明	buff: 数据缓冲区 bufflen: 缓冲区大小
返回	无
功能说明	清空缓冲区, 使得缓冲区数据为 0x00
实例	uchar buff[20]; uchar len = 20;

	<pre>fun_clear_buff(buff, len); //清空缓冲区 buff 全部数据为 0x00</pre>
--	---

4、fun_find_byte

函数原型	uint fun_find_byte(uchar *buff, uint buff_len, uchar findbyte)
参数说明	buff: 数据指针 buff_len: 数组长度 findbyte: 寻找数据
返回	0: 数组中没有指定数据 其他: 数据在数组中的位置
功能说明	在数组中检测指定数据，并返回在数组中的位置
实例	<pre>uchar buff[5]={0x02, 0x45, 0x34, 0x34, 0x70}; uint len = 5; uchar fchar=0x45; uint temp; temp = fun_find_byte(buff, len, fchar); //在数组中检测数据 0x45，返回在数组中的位置 2</pre>

5、fun_find_nbyte

函数原型	uint fun_find_nbyte(uchar *str1, uint len1, uchar *str2, uint len2)
参数说明	str1: 原始数据 str2: 当前数据 len1: 原始数据长度 len2: 当前数据长度
返回	0: 没有检测到数据 其他: 检测到的数据的位置
功能说明	在原始数据中查找当前数据，并返回查找到当前数据后一字节的位置
实例	<pre>uchar buff1[5]={0x02, 0x45, 0x34, 0x34, 0x70}; uint len1 = 5; uchar buff2[2]={0x34, 0x34}; uint len2 = 2; uint temp; temp = fun_find_nbyte(buff1, len1, buff2, len2); //在数组中检测数据 0x34, 0x34，返回在数组中的位置 4</pre>

6、fun_ascii_bcd

函数原型	void fun_ascii_bcd(uchar *Ascii, uchar *Bcd, uint len)
参数说明	Ascii: ASCII 码数据

	Bcd: BCD 数据 len: ASCII 码数据长度
返回	无
功能说明	将 ASCII 码转换为 BCD 数据, 转换后的数据长度为转换前数据长度的一半
实例	<pre> uchar ascii_buff[6]={ '3' , ' 4' , ' 5' , ' 6' , ' 7' , ' 8' }; uchar len = 6; uchar bcd_buff[3]; fun_ascii_bcd(ascii_buff, bcd_buff, len); //将 6 位 ASCII 码转换为 3 位 BCD 码 0x34, 0x56, 0x78 </pre>

7、fun_bcd_ascii

函数原型	void fun_bcd_ascii(uchar *Bcd, uchar *Ascii, uint len)
参数说明	Bcd: BCD 数据 Ascii: ASCII 码数据 len: BCD 数据长度
返回	无
功能说明	将 BCD 码转换为 ASCII 数据, 转换后的数据长度为转换前数据长度的 2 倍
实例	<pre> uchar bcd_buff[3]{ 0x34, 0x56, 0x78}; uchar len = 3; uchar ascii_buff[6]; fun_ascii_bcd(bcd_buff, ascii_buff, len); // 将 3 位 BCD 码 0x34, 0x56, 0x78 转换为 6 位 ASCII 码 '3' , ' 4' , ' 5' , ' 6' , ' 7' , ' 8' </pre>

8、fun_exp2_n

函数原型	uchar fun_exp2_n(uchar en)
参数说明	en: 输入数据, 范围 0~7
返回	0xFF: 输入参数错误 其他: 输入数据的 2 次方值
功能说明	求数据的 2 次方值
实例	<pre> uchar n = 2; uchar temp; temp = fun_exp2_n(n); //求 2 的 2 次方值 </pre>

9、fun_inversion_bit

函数原型	uchar fun_inversion_bit(uchar datain)
------	---------------------------------------

参数说明	datain: 需转换的数据
返回	转换后数据
功能说明	将输入 8 位数据进行倒置转换取反并返回新值
实例	<pre> uchar data = 5; uchar temp; temp = fun_inversion_bit(data); //求数据 5 倒置取反后的数据 </pre>

第9章 附录

1、数据类型原型列表

数 据 类 型	类 型 说 明
unsigned char、uchar、uint8	无符号 8 位整型变量
signed char、char、int8	有符号 8 位整型变量
unsigned short、ushort、uint16	无符号 16 位整型变量
signed short、short、int16	有符号 16 位整型变量
unsigned int、uint、uint32	无符号 32 位整型变量
signed int、int、int32	有符号 32 位整型变量