基于 MMI 的一个 Demo 程序

实现功能：

1. 新建一个应用，并且添加到主菜单的“工具箱”菜单里；
2. 程序进入后响应左右键消息；
3. 能够操作屏幕，在屏幕上打印信息；
4. 能够操作串口，将收到的数据再发送出去，如果收到的数据含有小写字母，则转换为相应大写字母再发送。
5. 能够操作网络，能够打开网络并且收发数据。

在尝试编写的过程中，主要遇到了以下问题：

1. 编译后死机，发现是由于增加了资源以后没有重新编译资源文件造成的。
2. 串口开机后在打印大量数据，根本不能调试，发现是工程模式没有关闭串口的打印信息，关闭后正常。
3. 串口只有在第一次收到数据后能够将转换后的字符串回送，之后就再也不能“激活”相应的处理函数——最后了解到原因是没有对串口缓冲进行清理（包括 FIFO 和 Buffer），添加相应的清理函数后问题解决。
4. 网路不能进行异步方式传输，只能同步方式使用；原因是没有将网络传输方式手动切换到异步，添加响应的切换函数后问题解决。
5. Timer 不能使用，一运行就死机；研究发现原来是因为 id 号是需要在系统中进行定义的，使用系统常用的 id 号后 Timer 能够正常启动执行。

刘诗远

2009-4-20

Demo 代码如下：（略去了部分头文件引用）

```
// ZTestAppSrc.c Test for MTK
#include "ZTestAppProt.h"
#include "ZTestAppDefs.h"

#include "stdC.h"
#include "MMI_Features.h"

#ifdef _MMI_ZTEST_
////////////////////////////////////////////////////////////////////////////////
// print info
static int   print_x= 0;

static void print_reset()
{
    print_x = 0;
}
```

```
static void print_soc_info(UI_string_type info)
{
    gui_move_text_cursor(0,print_x);
    gui_set_text_color(UI_COLOR_RED);
    gui_print_text((UI_string_type)info);
    gui_BLT_double_buffer(0, 0, UI_device_width - 1, UI_device_height - 1);
    print_x += 30;
}

////////////////////////////////////////////////////////////////////////////////
// uart operate

module_type UART_Owner = 0;
kal_uint8 sm_handle;

void mmi_ztestapp_uart_readyToRead_ind_hdler(void *msg);

static void init_uart()
{
    print_reset();
    sm_handle = L1SM_GetHandle();
    L1SM_SleepDisable(sm_handle);
    UART_Owner = UART_GetOwnerID(uart_port1);

    UART_SetOwner(uart_port1, MOD_MMI);

    SetProtocolEventHandler(mmi_ztestapp_uart_readyToRead_ind_hdler,MS
G_ID_UART_READY_TO_READ_IND);

    UART_SetBaudRate(uart_port1, UART_BAUD_115200, MOD_MMI);

    UART_PutBytes(uart_port1, (kal_uint8 *)("Hello world!"),strlen("Hello
world!"), MOD_MMI);

}

static void exit_uart()
{
    UART_SetOwner(uart_port1, (kal_uint8) UART_Owner);
    L1SM_SleepEnable(sm_handle);
}

void mmi_ztestapp_uart_readyToRead_ind_hdler(void *msg)
```

```
{
    kal_uint16 returnReadLen = 0;
    kal_uint16 returnReadLen = 0;
    kal_uint16 AvailableLen = 0;
    kal_uint8 status =0;
    kal_uint8 RRingBuf[8];
    kal_bool IsServerFlag = KAL_FALSE;
    kal_uint8 pattern_len = 0;
    kal_uint8 i;
    uart_ready_to_read_ind_struct*  uart_rtr_ind  =
(uart_ready_to_read_ind_struct*)msg;

    //Check if MOD_MMI is the owner of this port or not:
    if(UART_GetOwnerID(uart_rtr_ind->port) != MOD_MMI) return;

    //Try to get 8 bytes at one time
    AvailableLen = UART_GetBytesAvail(uart_rtr_ind->port);
    while(AvailableLen > 0)
    {
        //Try to get 8 bytes at one time
        returnReadLen = AvailableLen > 8 ? 8 : AvailableLen;
        AvailableLen -= returnReadLen;

        returnReadLen  =  UART_GetBytes(uart_rtr_ind->port,  RRingBuf,
returnReadLen, &status, MOD_MMI);

        for(i=0;i<returnReadLen;i++)
        {
            if((RRingBuf[i] >= 'a') && (RRingBuf[i] <= 'z'))
            {
                RRingBuf[i] -= 0x20;
            }
        }

            UART_PutBytes(uart_rtr_ind->port,  (kal_uint8*)RRingBuf,
returnReadLen, MOD_MMI);
    }

    UART_Purge(uart_rtr_ind->port, RX_BUF, MOD_MMI);
    UART_Purge(uart_rtr_ind->port, TX_BUF, MOD_MMI);
    UART_ClrTxBuffer(uart_rtr_ind->port, MOD_MMI);
    UART_ClrRxBuffer(uart_rtr_ind->port, MOD_MMI);

        UART_PutBytes(uart_rtr_ind->port,  (kal_uint8*)RRingBuf,
```

```
returnReadLen, MOD_MMI);
        return;

}

////////////////////////////////////////////////////////////////////////////////
// socket operate

#include "soc_api.h"
#define SOCKET_TIMER 46
#define SOCKET_PACKAGE "POST / HTTP/1.0\r\nHost:10.0.0.172:80\r\nX-Online-Host:www.baidu.com\r\nConnection: Keep-alive\r\nAccept:*/*\r\n\r\n"

static kal_int8 socket_id = -1;
static kal_uint32 account_id = 14;
static kal_uint16 mod_mmi =12354;
static sockaddr_struct socket_addr;

void notify_socket(void *inMsg)
{
    char tmp[256];
    app_soc_notify_ind_struct *soc_notify;

    soc_notify = (app_soc_notify_ind_struct *)inMsg;

    print_soc_info(L"Notify Soket !");

    if (socket_id< 0)
        return;

    switch (soc_notify->event_type) {

        case SOC_WRITE:
        print_soc_info(L"---Notify Soket Write!");
        break;

        case SOC_READ:
        print_soc_info(L"---Notify Soket Resd!");
        memset((void*)tmp, 0 , 256);
        soc_recv( socket_id, (kal_uint8*)tmp, 256, 0);
        print_soc_info(L"!Data recv:!");
        print_soc_info((UI_string_type)tmp);
```

```
        break;

        case SOC_CONNECT:
        print_soc_info(L"---Notify Soket Connect!");
        soc_send(socket_id,                    (kal_uint8*)SOCKET_PACKAGE,
strlen(SOCKET_PACKAGE), 0);
        break;

        case SOC_CLOSE:
        print_soc_info(L"---Notify Soket Close!");

        break;

        default:
        print_soc_info(L"---Notify Soket Error!");
        soc_close(socket_id);
        socket_id = -1;

            break;
    }
}

static void init_socket()
{
    kal_int8 ret;
    kal_uint8 val = 1;
    print_reset();
    print_soc_info(L"Start Soket Create!");

    SetProtocolEventHandler(                                notify_socket,
MSG_ID_APP_SOC_NOTIFY_IND);
    socket_id    =    soc_create(PF_INET,    SOCK_STREAM,    0,    MOD_MMI,
account_id);
    if( socket_id < 0 )
    {
        print_soc_info(L"Socket Create Error!");
        return;
    }
    else
    {
                if (soc_setsockopt(socket_id, SOC_NBIO, &val, sizeof(val)) <
0)
                {
                print_soc_info(L"Set socket to nonblock mode error !!");
```

```
            return;
                }
                val = SOC_READ | SOC_WRITE | SOC_CLOSE |
SOC_CONNECT;
                if (soc_setsockopt(socket_id, SOC_ASYNC, &val, sizeof(val)) <
0)
                {
                    print_soc_info(L"Set socket to nonblock mode error !!");
                    return;
                }
    }

    print_soc_info(L"Start Soket Created, Connect!");
    /* China Mobile WAP default Gateway IP Address: 10.0.0.172; Port: 80 */
    socket_addr.addr_len = 4;
    socket_addr.addr[0] = 10;
    socket_addr.addr[1] = 0;
    socket_addr.addr[2] = 0;
    socket_addr.addr[3] = 172;
    socket_addr.port = 80;

    soc_connect(socket_id, &socket_addr);
    // send data

}

static void exit_socket()
{
    soc_close(socket_id);
    print_soc_info(L"Start Soket Close!");
}

////////////////////////////////////////////////////////////////////////////////
// app operator

static void end_app();

static void start_app()
{
//  StartTimer(MOD_MMI, 1, init_socket);
    init_uart();
//  init_socket();
    SetKeyHandler( end_app, KEY_LSK, KEY_EVENT_UP);
}
```

```
static void end_app()
{
//   StartTimer(MOD_MMI, 1, exit_socket);
     exit_uart();
//   exit_socket();
     SetKeyHandler( start_app, KEY_LSK, KEY_EVENT_UP);
}

void mmi_ztestapp_exit(void)
{
     end_app();
}

void mmi_ztestapp_entry(void)
{
     EntryNewScreen(SCR_ZTEST_MAIN, NULL , mmi_ztestapp_entry, NULL);
     entry_full_screen();
     clear_screen();
     gui_BLT_double_buffer(0, 0, UI_device_width - 1, UI_device_height - 1);
     SetKeyHandler( GoBackHistory, KEY_RSK, KEY_EVENT_UP);
     SetKeyHandler( start_app, KEY_LSK, KEY_EVENT_UP);
}

void mmi_ztestapp_hilite(void)
{
     SetLeftSoftkeyFunction(mmi_ztestapp_entry, KEY_EVENT_UP);
}

void mmi_ztestapp_init(void)
{
     SetHiliteHandler( MENU_ID_ZTEST , mmi_ztestapp_hilite);
}

#endif
```