gxTcp lib(for s2e) 使用指南

格芯单片机，您的网络方案专家!!!

www.tcp-mcu.com

目 录

第 1 章 gxTcp LIB(for s2e)使用指南.....1
1.1 概述.....1
1.2 gxTcpLIB 的运行平台.....2
1.3 gxTcpLIB 与用户程序空间分布.....3
1.4 gxTcpLIB 工程设置.....4
1.5 工程目录结构.....5
1.6 工程文件介绍.....6
1.6.1 UART 驱动功能组件 bsp_uart.c.....6
1.6.2 看门狗功能组件 bsp_wdt.c.....7
1.6.3 串口字符串打印功能组件 uartstdio.c.....8
1.6.4 字符串格式命令输出内存功能组件 ustdlib.c.....8
1.6.5 数据队列 queue.c.....9
1.6.6 系统配置 bsp.h.....10
1.6.7 目标板硬件配置和一些系统常用函数 bsp.c.....11
1.6.8 网络与 UART 数据处理功能组件 s2e.c.....12
1.6.9 其他组件.....13
1.7 gxTcpLIB 运行.....13
1.8 gxTcpLIB 中断优先级分配.....13
1.9 gxTcpLIB 数据结构.....14
1.10 TCP 数据的接受流程.....15
1.11 任务与任务优先级.....16
1.12 任务堆栈与堆栈测试.....17
1.13 系统状态查询与参数设置指令.....18
1.14 简易网页.....19
1.15 参数配置与虚拟串口软件.....20

第 1 章 gxTcp LIB(for s2e)使用指南

1.1 概述

随着家庭网络的兴起（即现在炒的很火的物联网），以后越来越多的设备都会纳入网络互联范围。

这就对工程师们提出新的要求，即基于 TCP/IP 协议的网络产品的开发。但是 TCP/IP 是一个庞大的协议群，协议种类非常的多，实现又非常复杂，即使是一个高级程序员要搞清楚，也非一年半载之功。再加上嵌入式领域，芯片内部的 RAM 和 ROM 等资源非常有限，需要根据各种情况对 TCP/IP 协议各项参数进行优化，就是使得网络产品的开发门槛非常之高。而且经常是即使实现了其功能，也经不起测试，一旦出问题，工程师只能对着庞大的 TCP/IP 协议代码叹气，却无从下手。

针对这种情况，格芯单片机工作室开发了 gxTcpLIB，gxTcpLIB 是一个库文件，是一个实现了 TCP/IP 协议的稳定的库文件，这样工程师不再为庞大的 TCP/IP 协议而苦恼。你可以在不需要掌握 TCP/IP 协议的情况下即可开发网络产品。

使用 gxTcpLIB 的芯片，目前已经在物联网，网络打印机，工业自动化，LED 信息显示，门禁考勤等众多领域广泛应用。

注意：gxTcpLIB 使用 ucos-ii 作为系统平台，ucos-ii 为一款开源，且免费学习的嵌入式系统，如果用于产品需要支付相关版权费用，用户如果使用 gxTcpLIB 开发产品，请自行支付 ucos-ii 相关版权费用。

1.2 gxTcpLIB 的运行平台

gxTcpLIB 需要运行在 TI 公司的 ARM CORTEX-M3 内核 MCU，我们目前主要是运行在 LM3S6911 与 LM3S9B92 两款芯片。当然，gxTcpLIB 适用于 TI 公司 ARM CORTEX-M3 内核所有集成网络功能的 MCU。

TI 公司 ARM CORTEX-M3 内核 MCU 集成网络的 MAC 和 PHY，是目前业界集成度最高的网络解决方案，其详细资源如下：

	LM3S6911	LM3S9B92
内核	32BIT ARM CORTEX-M3 架构	32BIT ARM CORTEX-M3 架构
主频	50M	80M
FLASH	256K	256K
SRAM	64K	96K
UART	3	3
ADC	无	10BIT 16 channel
CAN	无	2
Ethernet	集成 MAC and PHY	集成 MAC and PHY
是否可扩展 SDRAM	否	是
USB	无	OTG
封装	LQFP-100	LQFP-100

表 1.1 MCU 资源

由于芯片内部 MAC 和 PHY，且有合适的 FLASH 与 SRAM，所以 LM3S6911，LM3S9B92 不需要扩张任何外围即可实现网络功能，硬件原理图非常简单。

1.3 gxTcpLIB 与用户程序空间分布

如图 1.1 所示，使用 gxTcpLIB 的程序，整个 FLASH 空间被分成 3 部分，

USER CODE 为用户代码，可以用来存放用户代码和 BOOTLOAD 代码，

USER DATA 该区域可以用来保存一些用户数据，如 IP 地址等参数，

ROM LIB 起始地址是 0x3e000, 8k 空间，这部分为固化库函数，gxTcpLIB 会用到这里面的函数，如果没有这部分，gxTcpLIB 将不能正常运行。该部分被加密保护，不能被读写。

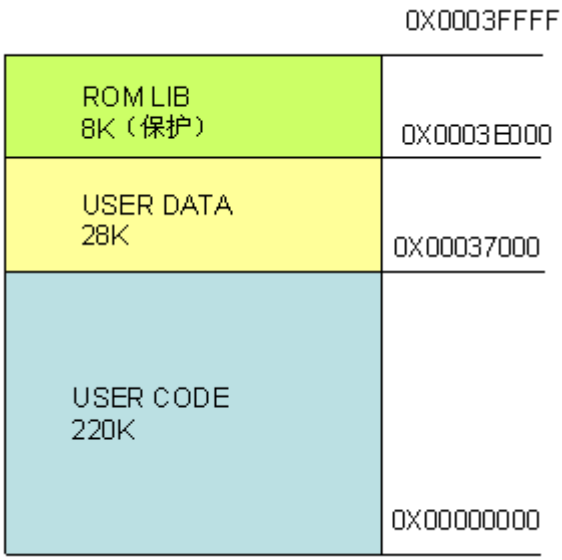

图 1.1 FLASH 空间分布

注意事项：如果是 TEMPEST 系列的芯片(如 LM3S9B92),在执行解锁操作时，会将芯片的 ROM LIB 擦除，ROM LIB 擦除后，gxTcpLIB 将不能正常运行。FURY 系列芯片（如 LM3S6911）不存在这个问题。

1.4 gxTcpLIB 工程设置

gxTcbLIB 的开发环境为 KEIL MDK。

由于 **ROM LIB** 占用高端 8K 地址空间，所以用户程序可用空间只有 248K，如图 1.2 所示，在 FLASH 空间定义里，需要将 SIZE 修改成 0x3E000

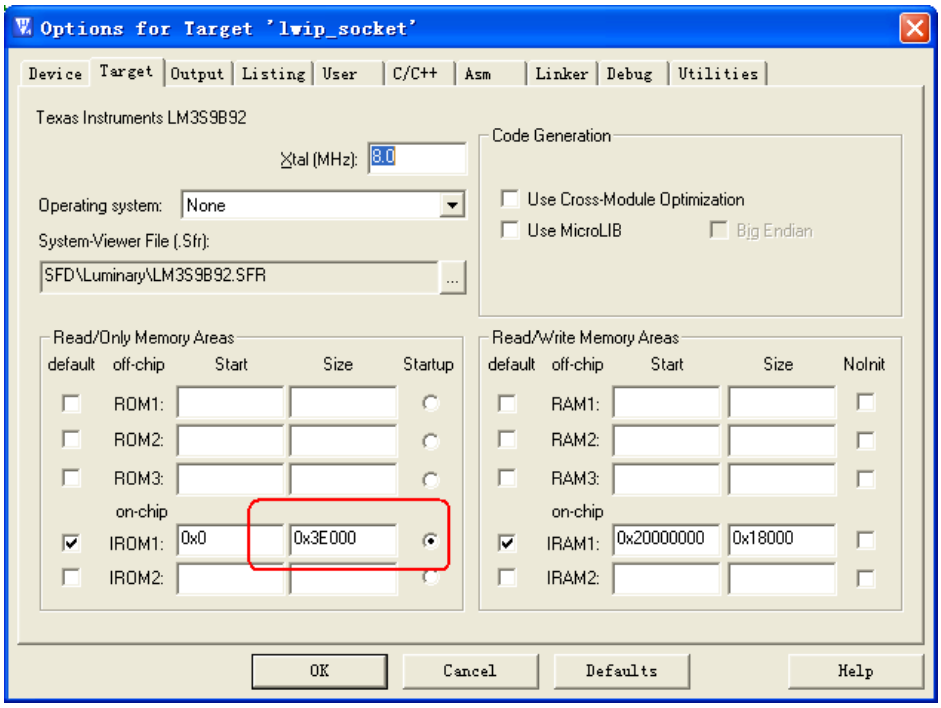

图 1.2 设置 FLASH 空间

如图 1.3 所示，使用默认的分散加载设置

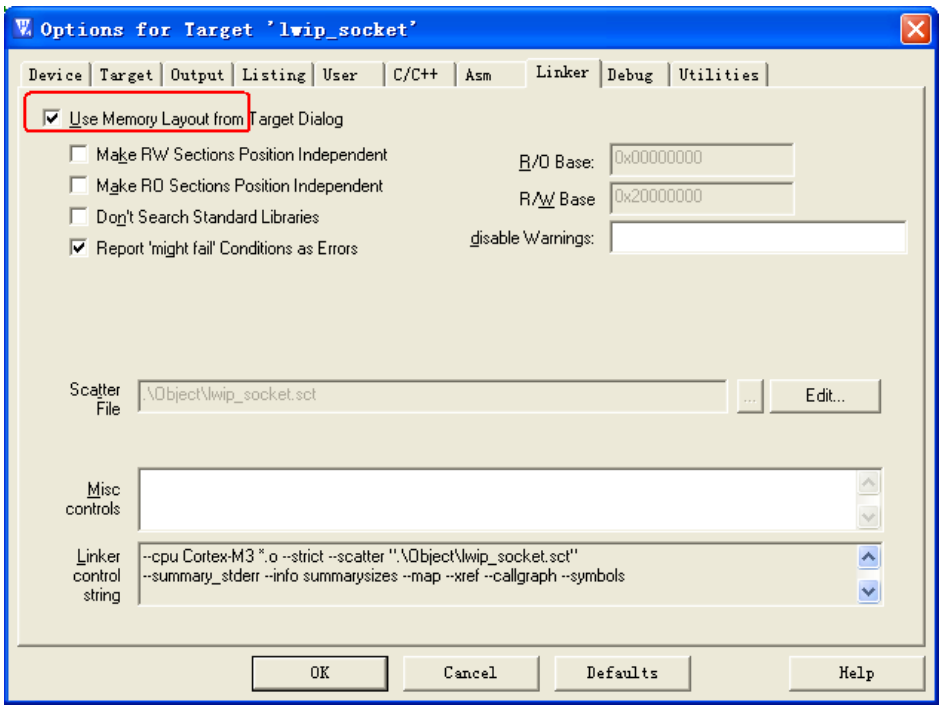

图 1.3 使用默认分散加载

1.5 工程目录结构

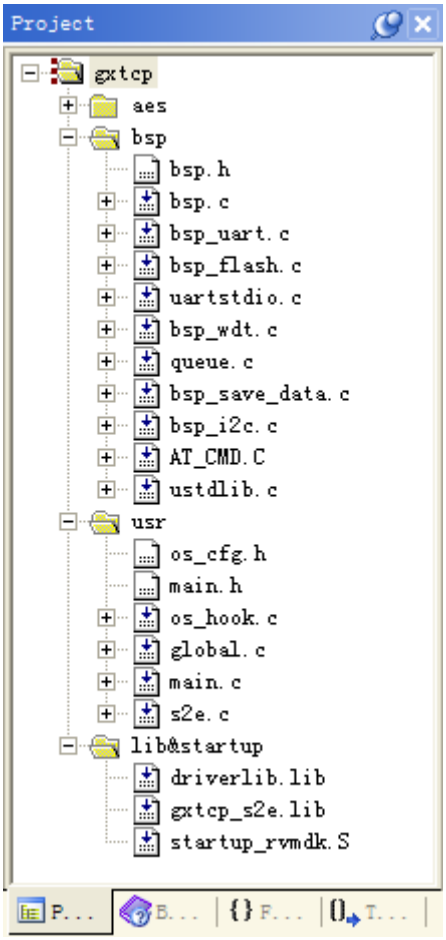

图 1.4 工程目录结构

如图 1.4 所示，gxTcp LIB 应用工程主要由 aes, bsp, usr, lib&startup 4 个文件夹组成。

aes 文件夹，aes 为高级加密算法标准，该文件夹内为 aes 加密算法的 C++源码实现。当使用 ENEC 协议进行数据传输时，在通讯前需要密码登录，密码内容使用 aes 加密。

bsp (board software pack)文件夹，顾名思义，即目标板软件开发支持包，主要包括目标板硬件驱动功能组件和一些通用功能组件。

usr 文件夹，主要放置 main.c 和用户应用程序。

lib&startup 文件夹，包括了 TI M3 外设驱动库——driverlib.lib；TI M3 启动配置文件——startup_rvmdk.s 和我们的 TCP/IP 应用库——gxtcp_pt.lib

1.6 工程文件介绍

接下来我们开始对 gxTcp LIB 应用工程的文件进行详细介绍。

1.6.1 UART 驱动功能组件 bsp_uart.c

bsp_uart.c 为 UART 驱动功能组件，且这是一个多任务下的 UART 驱动功能组件。

UART 驱动功能组件有一个自己独立的接受缓冲池。当 UART 发送接受中断时，UART 中断服务程序将数据从接受 FIFO 移入申请接受缓冲块；当调用 Uart_Rcv（）函数时，即可取出内存块中的数据。UART 接受缓冲池内存块的数目和内存块的大小是可以修改，如程序清单 1.1， UART_RCV_BLOCK_SIZE 定义了内存块的大小，UART_MSG_NUM 定义了内存块的数目。

```
#define     UART_RCV_BLOCK_SIZE     200     // UART 接受缓冲块大小
#define     UART_MSG_NUM            12      // UART 接收发送消息,最大条数
```

程序清单 1.1 定义 UART 接受缓冲

函数名称：int32 Uart_Open(STRUCT_UART *uart)
函数功能：对相应 UART 进行初始化，
输入参数：uart ---- UART 配置结构体指针。
返回值： 函数执行状态，0 --- 成功初始化

函数名称：int32 Uart_Rcv(STRUCT_UART *uart , uint16 headtime , void *rcvdata)
函数功能：从 UART 缓冲池中取数据。如果 UART 缓冲池中有数据，该函数立马返回数据 UART 接受的数据长度；如果 UART 缓存池中没有数据，该函数会阻塞等待数据，当等待时间超过设定超时时间时，该函数会返回 0。
输入参数：uart ---- UART 配置结构体指针。
headtime ---- 接受超时时间。
Rcvdata ---- 接收数据目的地址。
返回值：数据长度。

函数名称：uint8 Uart_Send(STRUCT_UART *uart , uint8 *pucBuffer, uint32 ulCount)
函数功能：将数据发送到 UART
输入参数：uart ---- UART 配置结构体指针。
pucBuffer ---- 需要发送的数据源地址
ulCount ---- 发送数据长度

函数名称：void UART0IntHandler(void)
函数功能：UART0 中断服务程序。该函数需要添加到 startup_rvmdk.s 文件的中断向量表中。UART1IntHandler， UART2IntHandler 分别为 UART1,UART2 的中断服务程序。

1.6.2 看门狗功能组件 bsp_wdt.c

bsp_wdt.c 为 WTD 驱动功能组件，且这是一个多任务下的看门狗功能组件。

如程序清单 1.2 所示，通过修改 MAX_DOG_NUM 的数值，即可定义相应数目的看门狗。如程序清单 1.2 所示，定义了 4 条看门狗，每个独立的看门狗都可以单独的使能，禁能，可以在 4 个不同的任务中起到防止程序进入死胡同的功能。

注意：看门狗 0，看门狗 1，保留给 gxTcp LIB 使用；看门狗 2，看门狗 3，保留给 ENEC 通讯协议处理；用户应用程序只能用编号为 4 以上的看门狗。

```
#define MAX_DOG_NUM        8
```

程序清单 1.2 定义看门狗数目

函数名称：void WD_HW_Init (uint32 cycle)

函数功能：初始化看门狗设置，并设置看门狗复位时间，如果超过设定时间，应用程序没有喂狗，看门狗将复位 MCU。

输入参数：cycle ----门狗复位时间

函数名称：void WD_Enable(int no)

函数功能：因为每条看门狗，都可以单独使能或禁能，该函数使能看门狗复位功能，看门狗只有在使能后才能发挥看家本领。

输入参数：no ---- 看门狗编号

函数名称：void WD_Disable(int no)

函数功能：因为每条看门狗，都可以单独使能或禁能，该函数禁能看门狗复位功能，禁能后，看门狗将不能起到看家的作用，注意看门狗功能模块在初始化的时候，默认把所有看门狗禁能。

输入参数：no ---- 看门狗编号

1.6.3 串口字符串打印功能组件 uartstdio.c

TI 有提供一个类似标准 C 中 prinf()函数的功能组件，该组件提供了函数 UARTprintf()，实现 print()函数一样的功能，可以完成数据的十进制字符串，十六进制字符串转换。

UARTprintf()函数数据默认打印窗口为 UART0，在调用调用初始化函数 UARTStdioInit()时设置。

函数名称：void UARTStdioInit(unsigned long ulPortNum)
函数功能：UARTprintf（）函数在使用前，必须调用本函数进行初始化。
调用本函数后，对应 UART 的波特率设置为默认的 115200
输入参数：ulPortNum --- UART 端口序列号， 0 代表使用 UART0 作为数据打印窗口。

函数名称：void UARTprintf(const char *pcString, ...)
函数功能：字符串打印，与标准 C 中 printf()函数功能完全一样，具体用法请参考 printf()函数。

1.6.4 字符串格式命令输出内存功能组件 ustdlib.c

与标准 C 中 sprintf()函数功能类似，该组件提供了功能函数 uvsnprintf(),实现 sprint()函数一样的功能。

函数 uvsnprintf()与函数 UARTprintf()功能十分类似，都是完成字符串格式命令转换，只是函数 UARTprintf()将转换后的数据输出到 UART，而函数 uvsnprintf()将转换后的数据输出到内存。gxtcp.lib 在 http 协议处理时有用到函数 uvsnprintf()。

函数名称：int uvsnprintf(char *pcBuf, unsigned long ulSize, const char *pcString,
va_list vaArgP)
函数功能：字符串格式命令转换，与 sprintf()函数功能一样，用法也可以参考 printf()函数。只是将格式转换后的数据输出到内存。该函数由于不牵涉到硬件，所以不需要初始化即可直接调用。

1.6.5 数据队列 queue.c

格芯单片机有提供给用户一个灵活、方便的数据队列功能模块。数据队列初始化后，即可方便的进行入队，出队操作。

数据队列在使用时需要申请一块内存空间，如程序清单 1.3 所示，一般可以定义一个全局数组。

程序清单 1.3 数据队列空间

```
uint8 queue_buf[MAX_RCV_NUM];
```

如程序清单 1.4 所示，我们可以设置数据队列空间的大小，S2E 接受需要开辟一个大的缓冲空间，建议一般在 8K 以上。

程序清单 1.4 数据队列空间大小

```
#define MAX_RCV_NUM                8000                // 定义缓冲大小
```

当然该数据队列可以使用 SDRAM 中的内存空间,如程序清单 1.5 所示，只需在初始化队列时，将使用一个指向 SDRAM 空间的指针即可。

程序清单 1.5 数据队列使用 SDRAM 空间

```
uint8 *queue_buf;
queue_buf=0x60000000;
queue_inital(queue_buf, 1000000);
```

函数名称： void queue_inital(void *buf, unsigned long buf_size)
函数功能： 队列初始化函数，队列使用之前必须初始化。
输入参数： buf --- 队列缓冲地址
size --- 为初始化时，传递给队列的缓冲区的最大长度

函数名称：long queue_in(void *buf, unsigned char *str, unsigned long len)
函数功能： 数据入队。
输入参数： buf --- 队列缓冲地址
str --- 入队数据指针
len --- 入队数据长度

函数名称： long queue_out(void *buf, unsigned char *pbuf,
unsigned long rcvlen, unsigned long timeover)
函数功能： 数据出队。
输入参数： buf --- 队列缓冲地址
pbuf --- 数据出队目的地址
rcvlen ---- 出对长度
timeover --- 超时时间，如果在指定时间队列中数据不够指定的出队长度
rcvlen，函数 queue_out 会超时退出，返回值为超时时出队数据长度。
当 timeover 为 0 时，代表函数 queue_out 永不超时退出。

1.6.6 系统配置 bsp.h

在 bsp.h 文件中，有一些重要的系统设置宏，如程序清单 1.6 所示。

DEBUG_MODE，为系统模式设置。DEBUG_MODE==1 时，设置系统为测试模式，测试模式为调试方便，禁能了看门狗，ENEC 协议通讯时也不需要密码登录。如果用户产品出厂时，为系统稳定性考虑，一定要记得设置成正常模式。

EN_UART1，为 UART1 使能配置。EN_UART1==1 时，使能 UART1 通讯。

PROTECT_CODE，为代码加密保护。当 PROTECT_CODE==1 时，系统在启动时会设置 JTAG 为普通 IO 功能，这样芯片就被加密了。JTAG 加密芯片后，必须解锁操作或在程序里恢复 JTAG 功能后，仿真器才能再次与 MCU 通讯。所以，JTAG 加密后，我们建议用户在程序留个恢复 JTAG 功能的后门。注意：如果是 TEMPEST 系列的芯片(如 LM3S9B92), 在执行解锁操作时，会将芯片的 ROM LIB 擦除，ROM LIB 擦除后，gxTcpLIB 将不能正常运行。FURY 系列芯片（如 LM3S6911）不存在这个问题。

TRANSPARENT,通讯协议选择。TRANSPARENT==1 时，TCP 数据位透明传输，即 TCP 收到什么数据，就往 UART 发送什么数据；当 TRANSPARENT==0 时，使用 ENEC 协议传输，TCP 的数据传输遵循 ENEC 协议。

EN_TCP_USR2,使能多用户登录。gxtcp.lib 支持多链接，即一个 TCP 端口，可以支持 2 台或 2 台以上远程登录。由于增加一个用户登录需要消耗很到 RAM，我们默认是只支持一个用户登录的。如果实在需要支持多用户登录功能，设置 EN_TCP_USR2==1， 即可支持 2 个用户同时登陆。

FURY，定义用户使用 MCU 所在的系列。因为用户选用 MCU 不同，一些系统的设置会有差异。LM3S6911 为 FURY 系列，LM3S9B92 为 TEMPEST 系列。

程序清单 1.6 系统设置宏

```
/*******************************************************************************************
    系统设置宏
*******************************************************************************************/
#define DEBUG_MODE              1       // 1 测试模式， 0 正常模式
#define EN_UART1                1       // 1 使用 UART1， 0 不使用 UART1
#define PROTECT_CODE            0       // 1 使能 JTAG 加密, 0 禁能 JTAG 加密
#define TRANSPARENT             1       // 1 透明传输, 0 ENEC 协议传输
#define EN_TCP_USR2             0       // 1 使能 TCP LINK 2
                                        // 0 禁能 TCP LINK 2
// #define TEMPEST                      // 定义当前为 TEMPEST 系列
#define FURY                            // 定义当前为 FURY 系列
```

在一些应用程序里经常需要开辟一些较大的数据缓冲，而有些缓冲使用的频率又很少，只有在某些特定的条件下才使用，如果分配静态的内存，有点不划算。为提供存的使用效率，我们使用 ucos 系统的内存管理，开辟了一个内存块 e_mem_pool，做动态内存使用。应用程序在使用内存前先申请，在使用完后释放。

MEM_BLOCK_SIZE，内存块的数据长度

MEM_MSG_NUM，内存块的数目

程序清单 1.7 动态内存池定义

```
/*********************************************************************************************
    定义内存池空间大小
*********************************************************************************************/
#define MEM_BLOCK_SIZE              1024                                //   1024 ,  TCP_MSS == 1000
#define MEM_MSG_NUM                  5                                  //   5

/*********************************************************************************************
    内存池空间
*********************************************************************************************/
INT8U          e_mem_pool[MEM_MSG_NUM][MEM_BLOCK_SIZE];      // 接收缓冲池
OS_MEM          *e_pool_ptr;                                 // 接收内存指针
```

1.6.7 目标板硬件配置和一些系统常用函数 bsp.c

bsp.c 为目标板硬件相关的配置和一些系统常用函数。

函数名称：void InitBSP(void)
函数功能：初始化目标板硬件，所有外设硬件的初始化都在此函数里设置。

函数名称：void DisableJTAG (void)
函数功能：禁能 JTAG 功能，通过将 JTAG 设置成普通 IO 功能，对程序加密。

函数名称：void EnableJTAG(void)
函数功能：恢复 JTAG 功能。

函数名称： uint16 t_ms(uint32 _tms)
函数功能：将将时间(单位：毫秒)转换成系统节拍(SYS_TICK)

函数名称：void tickISRHandler (void)
函数功能：系统定时器，中断服务程序。

1.6.8 网络与 UART 数据处理功能组件 s2e.c

最主要的文件之一，在这里主要完成以下两个功能。从网络接受数据，处理，然后发送给 UART；从 UART 接受数据，处理，然后发送给网络。这个文件创建了 3 个任务，App_ToEthTask, TcpUser1RcvTask 和 TcpUser1ToUartTask。

App_ToEthTask 任务，取出 UART 接受的数据，然后发送给网络。

TcpUser1RcvTask 任务，接受 TCP 链接数据并放入队列。

TcpUser1ToUartTask 任务，完成数据的出队，ENEC 协议的处理，然后将网络接受的数据发送给 UART。如果系统设置为透明传输，不需要进行 ENEC 协议处理，可以直接将队列的出队数据发送到 UART。

因为每一个用户登录，都是是一个独立的事件处理流程，如果使能多用户登录，即设置 EN_TCP_USR2==1， 那么在该文件里需要多创建两个任务 TcpUser2RcvTask 和 TcpUser2ToUartTask，分别处理用户 2 的数据接受，协议处理，数据发送等事件。

函数名称：void NetRawDeal(void *que, int socket_no)
函数功能：ENEC 协议处理
输入参数：que --- 数据队列指针
socket_no --- 用户 socket 信息结构体序号，每个用户都有一个 socket 信息结构体，如程序清单 1.8 所示，结构里存有 Sock ID，链接超时时间，是否链接，是否登录等信息。

程序清单 1.8 用户 socket 信息结构体

```
/*********************************************************************************************
    结构类型定义
*********************************************************************************************/
typedef struct _SockInfo
{
    uint32   ID;                                    // Sock ID 号
    uint16   time;                                  // 链接间隔时间
    uint8    link;                                  // 是否连接
    uint8    login;                                 // 是否登陆
    uint8    rcv_stop;                              // 停止接受标志。当队列满时，需要停止接受数据
    uint8    err;                                   // 错误号
    struct   lwip_socket *sock;                     //

}STRUCT_SOCK_INFO;
```

1.6.9 其他组件

AT_CMD.c 为 AT 命令修改模块参数设置实现。

bsp_i2c.c 为 I2C 协议读写 EEPROM 的代码。

global.c 里定义系统使用的全局变量，以及设备参数初始化设置。

os_hook.c 为 ucos 系统任务回调函数，为了减小功耗，在系统进入空闲任务后，在空闲任务回调函数里 CPU 执行休眠指令。

main.c 里放置了主程序入口，与启动任务，另外，可以在这里添加用户应用 。

driverlib.lib 为 TI 外设驱动库。

gxTcp_s2e.c 为 TCP/IP 库。

startup_rvmdk.S 为 MCU 启动汇编程序与中断向量表。

1.7 gxTcpLIB 运行

正确连接好硬件与仿真工具，打开 gxTcp 工程，点击 DEBUG 按钮  ，即可开始调试仿真你的程序。

如果程序你正常运行到 bsp.c 文件中 InitBSP（）函数中，UARTprintf("APP Start!\r\n")语句，打印 APP Start，说明网络硬件初始化成功，可以开始你的应用程序。

```
229     //=====================================
230     // 初始化TCP/IP协议
231     //=====================================
232     i = InitNic();
233     if (i)
234     {
235         UARTprintf("ERROR: %d\r\n", i);
236         while(1);
237     }
238     UARTprintf("APP Start!\r\n");
239
```

图 1.5 网络硬件初始化程序

1.8 gxTcpLIB 中断优先级分配

TI-M3 硬件支持 8 级中断嵌套，函数 IntPrioritySet(unsigned long ulInterrupt, unsigned char ucPriority)可以设置中断优先级，ulInterrupt 为中断号， ucPriority 为优先级。

注意：ucPriority 高 3 位有效，所以需要将优先级左移 5 位。且数值越小，代表优先级越高。

如程序清单所示，默认所有中断优先级为 1，或大于 1，优先级 0 保留给要求快速响应的特殊应用，优先级 0 保留给要求快速响应的特殊应用。

程序清单 1.9 系统中断优先级分配

```
IntPrioritySet(FAULT_PENDSV, (1<<5));          // 系统软中断
IntPrioritySet(FAULT_SYSTICK, (1<<5));         // 系统定时器中断
IntPrioritySet(INT_WATCHDOG, (1<<5));          // 看门狗中断
IntPrioritySet(INT_UART0, (1<<5));             // UART0 中断
IntPrioritySet(INT_ETH, (1<<5));               // Ethernet 中断
```

1.9 gxTcpLIB 数据结构

gxTcpLIB 的初始化参数，如 IP 地址，网关，通讯端口等系统主要参数，保存在设备通讯参数结构体 MdlCfg 和设备系统参数结构体 SysUtilities，其定义如程序清单 1.10 和程序清单 1.11 所示。

MdlCfg.IpPassword 为 TCP 通讯登录密码。

MdlCfg.IpAddr 为 IP 地址，子网掩码，网关，MAC 等参数。

MdlCfg.IpWorkMode 为通讯端口，TCP 链接超时时间等参数。

MdlCfg.IpWorkMode.over_time，over_time 大于 0，代表 TCP 链接超时时间，如果 TCP 链接超过设定时间没有接收到网络数据包，gxTcpLIB TCP 应用进程会主动断开 TCP 链接；over_time 等于 0，代表 TCP 应用进程永远不会主动断开 TCP 链接。

MdlCfg.UartNO 为 UART 波特率等参数。

SysUtilities.IsIDInit，ID 与 MAC 初始化标志，ID 与 MAC 只在第一次开机时初始化一次。

SysUtilities.CompName，可以存储用户公司名称。

SysUtilities.en_wdt，等于 1，代表 TCP 接受数据任务开启看门狗复位功能；等于 0，代表 TCP 接受数据任务没有开启看门狗复位功能。

SysUtilities.en_pt_stk，gxTcpLIB 开启了任务堆栈统计功能，当 en_pt_stk 等于 1，系统统计任务会在串口调试台打印堆栈使用情况；当 en_pt_stk 等于 0，系统虽然有堆栈统计，但是不会再串口调试台打印输出。en_pt_stk 一般默认为 0。当我们需要观看堆栈使用情况是，在 TCP 的 4000 端口，输入 es 指令即可设置 en_pt_stk 等于 1。

在调试程序时为了方便，建议关闭看门狗与超时主动断开 TCP 链接功能；在产品出厂时为了提供系统的可靠性，建议开启看门狗和超时主动断开 TCP 链接功能。

程序清单 1.10 设备通讯参数结构体

```
STRUCT_TCPIP_MODULE_CFG              MdlCfg;                          // 模块配置信息
/*******************************************************************************
                模块 参数
*******************************************************************************/
typedef struct _TcpIpModuleCfg
{
    STRUCT_PASSWORD      IpPassword;          // 只有 ID 和密码对了才能操作模块
    STRUCT_IP_ADDR       IpAddr;
    STRUCT_IP_WORKMODE IpWorkMode;          //
    STRUCT_UART_CFG      UartNO[2];
    // uint32                lrc;                  // 累加和校验
}STRUCT_TCPIP_MODULE_CFG;
```

程序清单 1.11 设备系统参数结构体

```
STRUCT_SYS_UTILITIES                 SysUtilities;                    // 系统杂项参数
typedef struct _SysUtilities
{
```

```
    uint32 IsIDInit;                    // ID 与 MAC 是否初始化，在出厂时需要初始化 ID 与 MAC
    uint8   IsIDSet;                    // ID 是否设置标志，ID 只允许设置一次
    uint8   IsCompNameSet;              // 公司名称是否设置
    uint8   SuperPWD[6];                // 超级密码
    uint8   CompName[20];               // 公司名称
    //=====================================================================
    // 追加系统控制参数
    //=====================================================================
    uint8   en_wdt;                     // 使能看门狗
    uint8   en_pt_stk;                  // 使能打印堆栈信息
    uint8   LinkNum;                    // TCP 通讯端口支持的链接数，最大为 2，最小为 1
}STRUCT_SYS_UTILITIES;
```

InitNic() 函数调用结构体 MdlCfg 中的参数，初始化网络。

MdlCfg 参数默认保存在 FLASH 中，在上电后，调用读 FLASH 获得参数。

1.10 TCP 数据的接受流程

网络初始化完成后，gxTcpLIB 会创建一个 TcpLinkTask 的任务，该任务会创建一个 TCP 链接，接受指定通讯端口的链接，在 TcpUser1RcvTask 任务接受 TCP 数据，TCP 数据接受流程如图 1.6 所示。

1. 准备工作，初始化系统与网络，初始化队列，队列的初始化对象为指针 queue_buf，queue_buf 可以是一个数组，也可以是一个指向某一内存块的指针。

2. 网络中断服务程序接受物理层过来的数据。

3. TCP APP Task 的任务接受 TCP 链接的数据，并调用函数 queue_in()数据入队，队列操作对象为 queue_buf 指针。

4. User Task 任务调用出队函数 queue_out（），获得网络接受的数据。

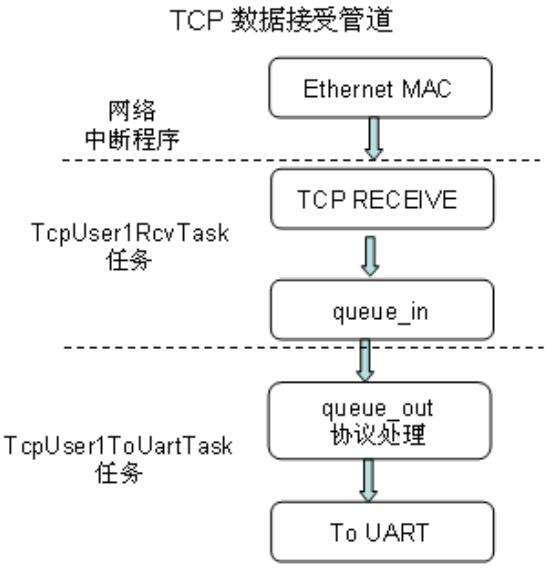

图 1.6 TCP 数据接受流程

1.11 任务与任务优先级

uC/OS-II 是一个多任务系统，也是一个抢占型系统，通过有先级仲裁，系统总是运行最高就绪态的最高优先级任务。

所以高优先级的任务在处理完相应事件后，必须调用系统的延时函数，释放 MCU 的使用权，否则低优先级的任务将得不到运行。

uC/OS-II 系统数字越小，优先级越高，0 为最高优先级任务。

gxTcpLIB 基于多任务的操作系统 ucos-ii, 如表 1.2 所示，gxTcpLIB 一共有 9 个任务。

表 1.2 系统任务列表

任务	优 先级	描述
uC/OS-II Stat	62	系统统计任务，统计堆栈使用情况等，如果 en_pt_stk==1，将在 UART 打印堆栈使用情况。
uC/OS-II Idle	63	系统空闲任务，也是最低优先级任务，当系统所有任务都处于等待状态时，系统将在空闲任务里调用休眠指令，以降低功耗。
tcpip_thread	2	TCP/IP 协议处理任务,在函数 InitNic() 中创建。
TCP_PACK Task	5	TCP/IP 协议处理任务,在函数 InitNic() 中创建。
Start Task	8	启动任务，在系统启动时初始化系统，创建其他任务，创建完其他任务后一直挂起。
TcpLinkTask	9	TCP&UDP 应用任务，创建 TCP 链接，接受相应通讯端口的链接处理。
App_LedTask	10	用户任务，LED 闪烁控制
TCP_USER1_RCV	15	TCP&UDP 应用任务，用户 1 接受 TCP 通讯端口数据
TCP_USER2_RCV	16	TCP&UDP 应用任务，用户 2 接受 TCP 通讯端口数据
TCP_USER1_TO_UART	17	用户任务，用户 1，TCP 接受数据协议处理，并发送给 UART。
TCP_USER2_TO_UART	18	用户任务，用户 2，TCP 接受数据协议处理，并发送给 UART。
APP_TO_ETH	20	用户任务，取出 UART 接受的数据，发送给以太网。
HTTP_PRIO	21	TCP&UDP 应用任务，WEB 登录处理。
QUERY_ST	30	TCP&UDP 应用任务，用于查询打印机状态，设置一些系统参数，通讯端口为 4000。
UDP FIND Task	31	TCP&UDP 应用任务，UDP 链接，用于广播查询 IP 地址。

如果用户应用程序需要增加任务，或进行任务优先级调整，请遵守以下规则：

1. 优先级 63,62 为 ucos 系统任务，不能动。
2. 优先级 10 以下任务保留给 TCP 协议处理与启动任务。

TCP/IP 协议处理任务与 TCP&UDP 应用任务都是基于信号量阻塞机制，当网络上没有数据包时，这些任务都处于挂起状态，不占用 CPU 的使用权。

1.12 任务堆栈与堆栈测试

每个任务都需要一个堆栈，堆栈空间分配是个很头痛的问题，堆栈开大一点吧，RAM 的资源非常有限和宝贵，怕浪费；堆栈开小一点吧，就怕堆栈溢出，导致系统崩溃。所以格芯单片机在创建任务时，先是给堆栈开一个较大空间，然后做堆栈测试，根据测试的堆栈使用情况再调整堆栈大小。

堆栈测试的一般原则是，先把所有的功能都运行一次，然后进行海量数据测试与长时间测试，这时得到的堆栈使用数据再上浮 30%，为最理想的堆栈大小。

gxTcpLIB 默认开启了堆栈测试功能，并且通过函数 StatInfoPrint()，将堆栈使用情况打印到串口。当然虽然默认开启了堆栈测试功能，但是不会往串口输出，如果需要输出，需要在 TCP 链接 4000 端口，连接并发送“es”字符串，使能堆栈测试输出。堆栈测试输出使能后，就可以通过串口数据看到堆栈的使用情况了。

格芯单片机提供的 gxTcpLIB 是做过堆栈测试的，测试数据如下：

```
tcpip_thread    PRIO: 2     STK used: 106   STK free: 150
TCP_PACK Task   PRIO: 5     STK used: 142   STK free: 58
Start Task      PRIO: 8     STK used: 96    STK free: 32
TCP LINK Task   PRIO: 9     STK used: 86    STK free: 82
LED blink       PRIO: 10    STK used: 24    STK free: 36
TCP USER1 RCV   PRIO: 15    STK used: 112   STK free: 68
TCP USER1 RCV   PRIO: 16    STK used: 112   STK free: 68
USER1 TO UART   PRIO: 17    STK used: 203   STK free: 53
USER1 TO UART   PRIO: 18    STK used: 203   STK free: 53
uart to Eth     PRIO: 20    STK used: 122   STK free: 28
http task       PRIO: 21    STK used: 106   STK free: 44
Query status    PRIO: 30    STK used: 114   STK free: 26
UDP FIND Task   PRIO: 31    STK used: 118   STK free: 32
uC/OS-II Stat   PRIO: 62    STK used: 56    STK free: 24
uC/OS-II Idle   PRIO: 63    STK used: 20    STK free: 30
```

1.13 系统状态查询与参数设置指令

我们前面有提到创建一个任务 Query status，该任务用于查询打印机状态，设置一些系统参数，TCP 通讯端口为 4000。

表 1.3 系统参数设置命令

指令	二进制码	功能描叙
0x1b v	0x1b 0x76	打印机状态查询指令。如果 Query status 在创建任务时，有设置打印机状态查询回调函数，则执行回调函数，并返回打印机状态；如果 Query status 在创建任务时，没有设置打印机状态查询回调函数，则返回状态信息无效。
es	0x65 0x73	enable print stk information 的缩写，使能堆栈信息输出，执行此命令后，设置 en_pt_stk==1，UART 窗口会有堆栈使用信息输出。
ds	0x64 0x73	disable print stk information 的缩写，禁能打印堆栈信息，执行此命令后，设置 en_pt_stk==0。
ed	0x65 0x64	enable watch dog 的缩写，使能 TCP APP Task 任务的看门狗，执行此命令后，设置 en_wdt==1。
dd	0x64 0x64	disable watch dog 的缩写，禁能 TCP APP Task 任务的看门狗，执行此命令后，设置 en_wdt==0。
st=ot	0x73 0x74	set over time 的缩写，设置 TCP APP Task 任务创建主通讯端口链接超时时间，设置 over_time = ot，单位为 100ms。如果设置命令为“st=1000”,即设置超时时间为 100 秒。如果主通讯端口在建立链接后，超过超时时间没有收到 TCP 数据，TCP APP Task 任务会主动关闭链接；如果 over_time==0， TCP APP Task 任务不会主动关闭链接。

1.14 简易网页

我们有提供一个简易网页，确保您的设备 IP 地址与您 PC 机 IP 地址在同一网段内，连接好网线，在 IE 浏览器中输入设备 IP，即可直至登录，查看一些系统参数。通过网页，也可以修改一些系统参数。

1.15 参数配置与虚拟串口软件

如果您使用 ENEC 协议，格芯单片机有提供一个基于 ENEC 协议的参数配置与虚拟串口功能的小软件 ModuleSerial，通过该软件可以完成设备查找，参数修改，虚拟串口通讯（即将网络虚拟成一个串口）等功能。

ModuleSerial 软件使用如下：

1. 安装软件.
2. 打开软件，弹出如下界面。

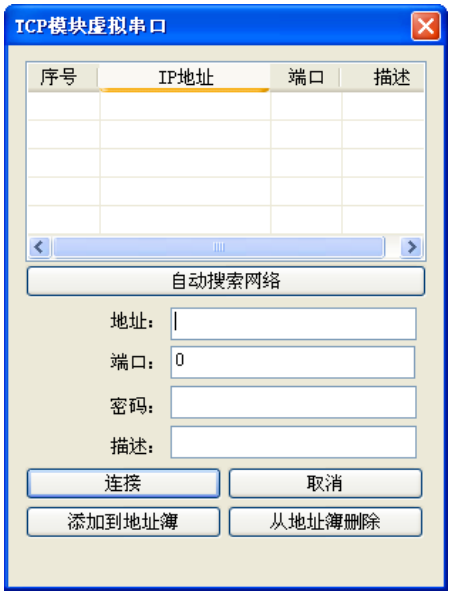

图 1.7 打开软件

3. 点击**"自动搜索网络"**，查找设备。

必须保证设备 IP 地址与 PC 机 IP 地址在同一网段内，如果找到设备，则软件会显示设备 IP 地址。

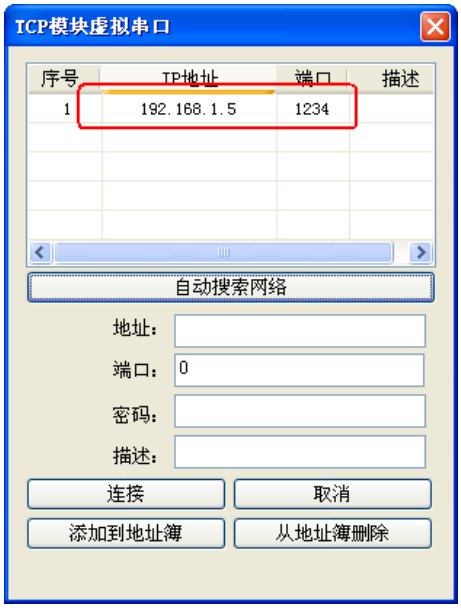

图 1.8 搜索设备

4. 点击搜索到的 IP 地址， 然后输入密码，点击**连接**进行登录。
登录成功后，会弹出如下对话框。

图 1.9 设备登录

5. 如图 1.9 所示，点击**设置**按钮，即弹出系统参数配置菜单。

图 1.10 设置参数

6. 虚拟串口的使用。如果是第一次登录，需要新建一个虚拟串口端口，如图 1.9 所示，点击**新建**，新建一个虚拟的串口号。

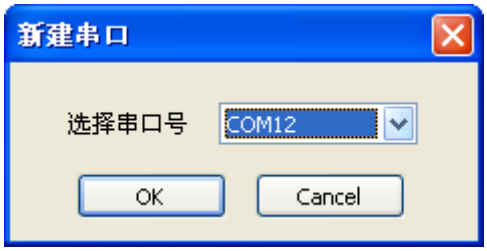

图 1.11 新建虚拟串口

7. 打开虚拟串口，如果虚拟串口端口已经建立，只需登录成功后，点击**打开**即可。

图 1.12 打开虚拟串口

8. 如果打开成功，在**串口助手**等串口软件里，就能看到并打开 TCP 虚拟出来的串口。

如图 1.13 所示，COM10 即为 TCP 虚拟出来的串口，你可以像使用串口一样使用 TCP 建立的虚拟网络链接。

图 1.13 TCP 虚拟串口