**密级：**

**文档编号：**

**版本号：V1.0**

# Task模型C函数接口

**北京华夏电通科技股份有限公司**

|  |  |
| --- | --- |
| 编制： | 生效日期： |
| 审核： | 批准： |

---------------------------------------------------------------------

北京华夏电通科技股份有限公司对本文件资料享受著作权及其它专属权利，未经书面许可，不得将该等文件资料（其全部或任何部分）披露予任何第三方，或进行修改后使用。

文件更改摘要：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **日期** | **版本号** | **修订说明** | **修订人** | **审核人** | **批准人** |
| 2013-04-26 |  | 首版 | 范仁亮,孙永 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## 目录

[Task模型C函数接口 1](#_Toc354733687)

[目录 3](#_Toc354733688)

[概述 3](#_Toc354733689)

[库使用方法： 4](#_Toc354733690)

[函数调用顺序 4](#_Toc354733691)

[函数说明 4](#_Toc354733692)

[线程部分 4](#_Toc354733693)

[容器部分 8](#_Toc354733694)

[文件部分 17](#_Toc354733695)

[网络部分 22](#_Toc354733696)

[Task部分 34](#_Toc354733697)

[常用工具函数 38](#_Toc354733698)

## 概述

为了方便C语言使用习惯的程序员，Task模型提供了C函数形式的接口。本文档着重描述Task模型C函数形式的接口，已确正确的使用Task模型。

Task模型包含OS层及Task层。



图（一）Task模型整体结构

Task模型分两层，第一层为平台无关基础库，第二层为Task模型。平台无关基础库包含了编程过程中将用到的常用技术。目前有线程部分，容器部分，文件部分，网络部分。Task模型使用基础库提供的功能实现了任务模型。所谓任务模型就是当任意事件发生时，有个任务来处理它。

## 库使用方法：

头文件：

#include “ChnsysTaskModel.h”

实际包含了ChnsysTypes.h, ChnsysFiles.h, ChnsysLog.h, ChnsysContainers.h, ChnsysThread.h, ChnsysTask.h, ChnsysSocket.h, ChnsysUtil.h。

库文件：

windows：

TaskModeld.lib（Debug，MTd），TaskModel.lib（Release，MT）

linux：

libTaskModel.a

## 函数调用顺序

暂略。见例子。

## 函数说明

### 线程部分

#### 创建线程实例

**函数**： OS\_THREAD\_HANDLE OS\_THREAD\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_THREAD\_Start等函数的句柄参数。

**说明**： 无

#### 销毁线程实例

**函数**： VOID OS\_THREAD\_DestroyInstance(OS\_THREAD\_HANDLE hOSThread);

**参数**： [in]hOSThread 线程实例句柄

**返回值**： 无

**说明**： 无

#### 线程回调函数

**函数**： typedef CHNSYS\_BOOL (\*OS\_THREAD\_CallBackFunc)(OS\_THREAD\_HANDLE hOSThread, int nThreadPhase, VOID \*pContext);

**参数**： [out]hOSThread 线程实例句柄

[out]nThreadPhase 线程回调阶段标志

[out]pContext 上下文参数

**返回值**： 当nThreadPhase=1时，返回FALSE表示结束线程。其他情况建议返回TRUE。

**说明**： 当多个线程对应一个回调函数时，可通过hOSThread参数来判断当前回调是否对应指定的线程。

nTreadPhase用于标识线程回调阶段。

nThreadPhase=1：表示线程进入循环前。如果在此阶段返回FALSE，则线程自动结束。可以在此阶段可以做一些初始化操作。  
nThreadPhase=2：表示线程循环中。Task模型内部本身是个while循环，所以在阶段2使用者不需要做while循环来防止线程退出。

nThreadPhase=3：表示线程结束循环后。此阶段可以做一些反初始化的操作。

pContext是一个上下文参数，使用者可以在设置回调时设置一个上下文指针，便于做更多业务相关的应用。

#### 设置线程回调函数

**函数**： VOID OS\_THREAD\_SetCallBack(OS\_THREAD\_HANDLE hOSThread, OS\_THREAD\_CallBackFunc cbf, VOID \*pContext);

**参数**： [in]hOSThread 线程实例句柄

[in]cbf 线程回调函数

[in]pContext 上下文参数

**返回值**： 无

**说明**： 无

#### 开启线程

**函数**： CHNSYS\_BOOL OS\_THREAD\_Start(OS\_THREAD\_HANDLE hOSThread);

**参数**： [in]hOSThread 线程实例句柄

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无

#### 停止线程

**函数**： CHNSYS\_BOOL OS\_THREAD\_Stop(OS\_THREAD\_HANDLE hOSThread, CHNSYS\_BOOL bWait = TRUE);

**参数**： [in]hOSThread 线程实例句柄

[in]bWait 等待线程退出标志

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 当bWait=TRUE时，函数会等到线程正常退出才返回。调用此函数时要确保线程未阻塞在回调函数上。

#### 创建互斥信号量实例

**函数**： OS\_MUTEX\_HANDLE OS\_MUTEX\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_MUTEX\_Lock等函数的句柄参数。

**说明**： 在多个线程应用中互斥信令量用于保护共享的变量，使各线程之间互斥使用共享变量。

#### 销毁互斥信号量实例

**函数**： VOID OS\_MUTEX\_DestroyInstance(OS\_MUTEX\_HANDLE hOSMutex);

**参数**： [in]hOSMutex 互斥信号量句柄

**返回值**： 无

**说明**： 无

#### 互斥信号量加锁

**函数**： VOID OS\_MUTEX\_Lock(OS\_MUTEX\_HANDLE hOSMutex);

**参数**： [in]hOSMutex 互斥信号量句柄

**返回值**： 无

**说明**： 加锁，解锁要配对，防止死锁。

#### 互斥信号量解锁

**函数**： VOID OS\_MUTEX\_Unlock(OS\_MUTEX\_HANDLE hOSMutex);

**参数**： [in]hOSMutex 互斥信号量句柄

**返回值**： 无

**说明**： 加锁，解锁要配对，防止死锁。

#### 创建条件信号量实例

**函数**： OS\_CONDITION\_HANDLE OS\_CONDITION\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_CONDITION\_Wait等函数的句柄参数。

**说明**： 条件信号量用于线程间同步

#### 销毁条件信号量实例

**函数**： VOID OS\_CONDITION\_DestroyInstance(OS\_CONDITION\_HANDLE hOSCondition);

**参数**： [in]hOSCondition 条件信号量句柄

**返回值**： 无

**说明**： 无

#### 等待条件信号量

**函数**： VOID OS\_CONDITION\_Wait(OS\_CONDITION\_HANDLE hOSCondition, OS\_MUTEX\_HANDLE hOSMutex, CHNSYS\_INT nTimeoutMilliSeconds);

**参数**： [in]hOSCondition 条件信号量句柄

[in]hOSMutex 互斥信号量句柄

[in]nTimeoutMilliSeconds 等待超时时间(毫秒)

**返回值**： 无

**说明**： linux下应用时，关于hOSMutex参数的描述，见pthread\_cond\_timedwait函数的解释。在windows下应用时，hOSMutex可填NULL。

#### 触发条件信号量

**函数**： VOID OS\_CONDITION\_Signal(OS\_CONDITION\_HANDLE hOSCondition);

**参数**： [in]hOSCondition 条件信号量句柄

**返回值**： 无

**说明**： 无

### 容器部分

#### 创建INT数组实例

**函数**： OS\_INTARRAY\_HANDLE OS\_INTARRAY\_CreateInstance(CHNSYS\_UINT nInitSize);

**参数**： [in]nInitSize 数组大小

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_INTARRAY\_Add等函数的句柄参数。

**说明**： 无

#### 销毁INT数组实例

**函数**： VOID OS\_INTARRAY\_DestroyInstance(OS\_INTARRAY\_HANDLE hIntArray);

**参数**： [in] hIntArray INT数组实例句柄

**返回值**： 无

**说明**： 无

#### 在INT数组末尾插入数据

**函数**： CHNSYS\_INT OS\_INTARRAY\_Add(OS\_INTARRAY\_HANDLE hIntArray, CHNSYS\_INT nValue);

**参数**： [in] hIntArray INT数组实例句柄

[in] nValue 插入的数据

**返回值**： -1表示执行失败，成功返回插入数据在数组的索引值。

**说明**： 无

#### 设置INT数组指定位置数据

**函数**： CHNSYS\_INT OS\_INTARRAY\_Set(OS\_INTARRAY\_HANDLE hIntArray, CHNSYS\_INT nIndex, CHNSYS\_INT nValue);

**参数**： [in] hIntArray INT数组实例句柄

[in] nIndex 索引位置

[in] nValue 数据值

**返回值**： -1表示执行失败，成功返回设定数据在数组的索引值。

**说明**： 无

#### 在INT数组指定位置插入数据

**函数**： CHNSYS\_INT OS\_INTARRAY\_Insert(OS\_INTARRAY\_HANDLE hIntArray, CHNSYS\_INT nIndex, CHNSYS\_INT nValue);

**参数**： [in] hIntArray INT数组实例句柄

[in] nIndex 索引位置

[in] nValue 数据值

**返回值**： -1表示执行失败，成功返回插入数据在数组的索引值。

**说明**： 无

#### 移除INT数组指定位置数据

**函数**： CHNSYS\_INT OS\_INTARRAY\_Remove(OS\_INTARRAY\_HANDLE hIntArray, CHNSYS\_INT nIndex);

**参数**： [in] hIntArray INT数组实例句柄

[in] nIndex 索引位置

**返回值**： -1表示执行失败，成功返回移除数据在数组的索引值。

**说明**： 无

#### 移除INT数组全部数据

**函数**： CHNSYS\_INT OS\_INTARRAY\_RemoveAll(OS\_INTARRAY\_HANDLE hIntArray);

**参数**： [in] hIntArray INT数组实例句柄

**返回值**： -1表示执行失败，成功返回0。

**说明**： 无

#### 获取INT数组数据数量

**函数**： CHNSYS\_INT OS\_INTARRAY\_Count(OS\_INTARRAY\_HANDLE hIntArray);

**参数**： [in] hIntArray INT数组实例句柄

**返回值**： -1表示hIntArray为NULL，其他表示成功返回数据个数。

**说明**： 无

#### 获取INT数组指定位置数据

**函数**： CHNSYS\_INT OS\_INTARRAY\_At(OS\_INTARRAY\_HANDLE hIntArray, CHNSYS\_INT nIndex, CHNSYS\_INT \*pValue);

**参数**： [in] hIntArray INT数组实例句柄

[in] nIndex 索引位置

[out] pValue 数据值

**返回值**： -1表示执行失败，成功返回0。

**说明**： 无

#### 创建指针数组实例

**函数**： OS\_VOIDARRAY\_HANDLE OS\_VOIDARRAY\_CreateInstance(CHNSYS\_UINT nInitSize);

**参数**： [in]nInitSize 数组大小

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_VOIDARRAY\_Add等函数的句柄参数。

**说明**： 无

#### 销毁指针数组实例

**函数**： VOID OS\_VOIDARRAY\_DestroyInstance(OS\_VOIDARRAY\_HANDLE hVOIDArray);

**参数**： [in] hVOIDArray 指针数组实例句柄

**返回值**： 无

**说明**： 无

#### 在指针数组末尾插入数据

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_Add(OS\_VOIDARRAY\_HANDLE hVOIDArray, VOID \*pValue);

**参数**： [in] hVOIDArray 指针数组实例句柄

[in] pValue 插入的数据

**返回值**： -1表示执行失败，成功返回插入数据在数组的索引值。

**说明**： 无

#### 设置指针数组指定位置数据

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_Set(OS\_VOIDARRAY\_HANDLE hVOIDArray, CHNSYS\_INT nIndex, VOID \*pValue);

**参数**： [in] hVOIDArray 指针数组实例句柄

[in] nIndex 索引位置

[in] pValue 数据值

**返回值**： -1表示执行失败，成功返回设定数据在数组的索引值。

**说明**： 无

#### 在指针数组指定位置插入数据

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_Insert(OS\_VOIDARRAY\_HANDLE hVOIDArray, CHNSYS\_INT nIndex, VOID \*pValue);

**参数**： [in] hVOIDArray 指针数组实例句柄

[in] nIndex 索引位置

[in] pValue 数据值

**返回值**： -1表示执行失败，成功返回插入数据在数组的索引值。

**说明**： 无

#### 移除指针数组指定位置数据

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_Remove(OS\_VOIDARRAY\_HANDLE hVOIDArray, CHNSYS\_INT nIndex);

**参数**： [in] hVOIDArray INT数组实例句柄

[in] nIndex 索引位置

**返回值**： -1表示执行失败，成功返回移除数据在数组的索引值。

**说明**： 无

#### 移除指针数组全部数据

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_RemoveAll(OS\_VOIDARRAY\_HANDLE hVOIDArray);

**参数**： [in] hVOIDArray 指针数组实例句柄

**返回值**： -1表示执行失败，成功返回0。

**说明**： 无

#### 获取指针数组数据数量

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_Count(OS\_VOIDARRAY\_HANDLE hVOIDArray);

**参数**： [in] hVOIDArray 指针数组实例句柄

**返回值**： -1表示hVOIDArray为NULL，其他表示成功返回数据个数。

**说明**： 无

#### 获取指针数组指定位置数据

**函数**： CHNSYS\_INT OS\_VOIDARRAY\_At(OS\_VOIDARRAY\_HANDLE hVOIDArray, CHNSYS\_INT nIndex, VOID \*\*ppValue);

**参数**： [in] hVOIDArray 指针数组实例句柄

[in] nIndex 索引位置

[out] ppValue 数据值

**返回值**： -1表示执行失败，成功返回0。

**说明**： 无

#### 创建堆栈元素实例

**函数**： OS\_HEAPELEMENT\_HANDLE OS\_HEAPELEMENT\_CreateInstance(CHNSYS\_UINT64 nValue, VOID \*pObject);

**参数**： [in] nValue 堆栈元素值

[in] pObject 对象指针

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_HEAPELEMENT\_GetValue等函数的句柄参数。

**说明**： 无

#### 销毁堆栈元素实例

**函数**： VOID OS\_HEAPELEMENT\_DestroyInstance(OS\_HEAPELEMENT\_HANDLE hHeapElement);

**参数**： [in] hHeapElement 堆栈元素实例句柄

**返回值**： 无

**说明**： 无

#### 获取堆栈元素值

**函数**： CHNSYS\_BOOL OS\_HEAPELEMENT\_GetValue(OS\_HEAPELEMENT\_HANDLE hHeapElement, CHNSYS\_UINT64 \*pValue);

**参数**： [in] hHeapElement 堆栈元素实例句柄

[out] pValue 堆栈元素值

**返回值**： 返回FALSE表示hHeapElement为NULL，返回TRUE表示获取成功。

**说明**： 无

#### 设置堆栈元素值

**函数**： CHNSYS\_BOOL OS\_HEAPELEMENT\_SetValue(OS\_HEAPELEMENT\_HANDLE hHeapElement, CHNSYS\_UINT64 nValue);

**参数**： [in] hHeapElement 堆栈元素实例句柄

[in] nValue 堆栈元素值

**返回值**： 返回FALSE表示hHeapElement为NULL，返回TRUE表示设置成功。

**说明**： 无

#### 获取堆栈元素对象

**函数**： CHNSYS\_BOOL OS\_HEAPELEMENT\_GetObject(OS\_HEAPELEMENT\_HANDLE hHeapElement, VOID \*\*ppObject);

**参数**： [in] hHeapElement 堆栈元素实例句柄

[out] ppObject 堆栈元素对象

**返回值**： 返回FALSE表示hHeapElement为NULL，返回TRUE表示获取成功。

**说明**： 无

#### 设置堆栈元素对象

**函数**： CHNSYS\_BOOL OS\_HEAPELEMENT\_SetObject(OS\_HEAPELEMENT\_HANDLE hHeapElement, VOID \*pObject);

**参数**： [in] hHeapElement 堆栈元素实例句柄

[in] pObject 堆栈元素对象

**返回值**： 返回FALSE表示hHeapElement为NULL，返回TRUE表示设置成功。

**说明**： 无

#### 获取堆栈元素所在堆栈

**函数**： CHNSYS\_BOOL OS\_HEAPELEMENT\_GetHeap(OS\_HEAPELEMENT\_HANDLE hHeapElement, OS\_HEAP\_HANDLE \*phHeap);

**参数**： [in] hHeapElement 堆栈元素实例句柄

[out] phHeap 堆栈实例句柄

**返回值**： 返回FALSE表示hHeapElement为NULL，返回TRUE表示获取成功。

**说明**： 无

#### 创建堆栈实例

**函数**： OS\_HEAP\_HANDLE OS\_HEAP\_CreateInstance(CHNSYS\_UINT nInitSize);

**参数**： [in] nInitSize 堆栈大小

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_HEAP\_Insert等函数的句柄参数。

**说明**： 无

#### 销毁堆栈实例

**函数**： VOID OS\_HEAP\_DestroyInstance(OS\_HEAP\_HANDLE hHeap);

**参数**： [in] hHeap 堆栈实例句柄

**返回值**： 无

**说明**： 无

#### 向堆栈插入元素

**函数**： CHNSYS\_BOOL OS\_HEAP\_Insert(OS\_HEAP\_HANDLE hHeap,OS\_HEAPELEMENT\_HANDLE hHeapElement);

**参数**： [in] hHeap 堆栈实例句柄

[in] hHeapElement 堆栈元素实例句柄

**返回值**： 返回FALSE表示hHeap或hHeapElement为NULL，返回TURE表示成功。

**说明**： 无

#### 获取堆栈第一个索引元素

**函数**： CHNSYS\_BOOL OS\_HEAP\_ExtractMin(OS\_HEAP\_HANDLE hHeap, OS\_HEAPELEMENT\_HANDLE \*phHeapElement);

**参数**： [in] hHeap 堆栈实例句柄

[in] hHeapElement 堆栈元素实例句柄

**返回值**： 返回FALSE表示hHeap为NULL，返回TURE表示成功。

**说明**： 执行OS\_HEAP\_ExtractMin获取元素后，会在堆栈中自动删除该元素。

#### 获取堆栈第一个索引元素

**函数**： CHNSYS\_BOOL OS\_HEAP\_PeekMin(OS\_HEAP\_HANDLE hHeap, OS\_HEAPELEMENT\_HANDLE \*phHeapElement);

**参数**： [in] hHeap 堆栈实例句柄

[in] hHeapElement 堆栈元素实例句柄

**返回值**： 返回FALSE表示hHeap为NULL，返回TURE表示成功。

**说明**： 与OS\_HEAP\_ExtractMin功能相同，但执行OS\_HEAP\_PeekMin后不会删除该元素。

#### 删除堆栈指定元素

**函数**： CHNSYS\_BOOL OS\_HEAP\_Remove(OS\_HEAP\_HANDLE hHeap, OS\_HEAPELEMENT\_HANDLE hHeapElement);

**参数**： [in] hHeap 堆栈实例句柄

[in] hHeapElement 堆栈元素实例句柄

**返回值**： 返回FALSE表示hHeap或hHeapElement为NULL，返回TURE表示成功。

**说明**：

#### 删除堆栈所有元素

**函数**： CHNSYS\_BOOL OS\_HEAP\_RemoveAll(OS\_HEAP\_HANDLE hHeap);

**参数**： [in] hHeap 堆栈实例句柄

**返回值**： 返回FALSE表示hHeap为NULL，返回TURE表示成功。

**说明**：

#### 获取堆栈元素数量

**函数**： CHNSYS\_INT OS\_HEAP\_Count(OS\_HEAP\_HANDLE hHeap);

**参数**： [in] hHeap 堆栈实例句柄

**返回值**： 返回-1表示hHeap为NULL，其他表示成功为元素的数量。

**说明**：

#### 创建队列元素实例

**函数**： OS\_QUEUEELEMENT\_HANDLE OS\_QUEUEELEMENT\_CreateInstance(VOID \*pObject);

**参数**： [in] pObject 对象指针

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_QUEUEELEMENT\_GetObject等函数的句柄参数。

**说明**： 无

#### 销毁队列元素实例

**函数**： VOID OS\_QUEUEELEMENT\_DestroyInstance(OS\_QUEUEELEMENT\_HANDLE hQueueElement);

**参数**： [in] hQueueElement 队列元素实例句柄

**返回值**： 无

**说明**： 无

#### 获取队列元素对象

**函数**： CHNSYS\_BOOL OS\_QUEUEELEMENT\_GetObject(OS\_QUEUEELEMENT\_HANDLE hQueueElement, VOID \*\*ppObject);

**参数**： [in] hQueueElement 队列元素实例句柄

[out] ppObject 元素对象

**返回值**： 返回FALSE表示hQueueElement为NULL，返回TRUE表示成功。

**说明**： 无

#### 设置队列元素对象

**函数**： CHNSYS\_BOOL OS\_QUEUEELEMENT\_SetObject(OS\_QUEUEELEMENT\_HANDLE hQueueElement, VOID \*pObject);

**参数**： [in] hQueueElement 队列元素实例句柄

[in] pObject 元素对象

**返回值**： 返回FALSE表示hQueueElement为NULL，返回TRUE表示成功。

**说明**： 无

#### 获取队列元素所在队列

**函数**： CHNSYS\_BOOL OS\_QUEUEELEMENT\_GetQueue(OS\_QUEUEELEMENT\_HANDLE hQueueElement, OS\_QUEUE\_HANDLE \*phQueue);

**参数**： [in] hQueueElement 队列元素实例句柄

[out] phQueue 队列实例句柄

**返回值**： 返回FALSE表示hQueueElement为NULL，返回TRUE表示成功。

**说明**： 无

#### 创建队列实例

**函数**： OS\_QUEUE\_HANDLE OS\_QUEUE\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_QUEUE\_Push等函数的句柄参数。

**说明**： 该队列是先进先出类型。

#### 销毁队列实例

**函数**： VOID OS\_QUEUE\_DestroyInstance(OS\_QUEUE\_HANDLE hQueue);

**参数**： [in] hQueue 队列实例句柄

**返回值**： 无

**说明**： 无

#### 向队列插入元素

**函数**： CHNSYS\_BOOL OS\_QUEUE\_Push(OS\_QUEUE\_HANDLE hQueue, OS\_QUEUEELEMENT\_HANDLE hQueueElement);

**参数**： [in] hQueue 队列实例句柄

[in] hQueueElement 队列元素实例句柄

**返回值**： 返回FALSE表示hQueue或hQueueElement为NULL，返回TRUE表示成功。

**说明**： OS\_QUEUE\_Push函数是在队列末尾插入元素

#### 获取队列元素

**函数**： CHNSYS\_BOOL OS\_QUEUE\_Pop(OS\_QUEUE\_HANDLE hQueue, OS\_QUEUEELEMENT\_HANDLE \*phQueueElement);

**参数**： [in] hQueue 队列实例句柄

[out] hQueueElement 队列元素实例句柄

**返回值**： 返回FALSE表示hQueue为NULL，返回TRUE表示成功。

**说明**： 获取队列的第一个元素，并且在队列中删除该元素。

#### 获取队列第一个元素

**函数**： CHNSYS\_BOOL OS\_QUEUE\_GetHead(OS\_QUEUE\_HANDLE hQueue, OS\_QUEUEELEMENT\_HANDLE \*phQueueElement);

**参数**： [in] hQueue 队列实例句柄

[out] hQueueElement 队列元素实例句柄

**返回值**： 返回FALSE表示hQueue为NULL，返回TRUE表示成功。

**说明**： 获取队列的第一个元素，与OS\_QUEUE\_Pop不同，在队列中不会删除该元素。

#### 获取队列最后一个元素

**函数**： CHNSYS\_BOOL OS\_QUEUE\_GetTail(OS\_QUEUE\_HANDLE hQueue, OS\_QUEUEELEMENT\_HANDLE \*phQueueElement);

**参数**： [in] hQueue 队列实例句柄

[out] hQueueElement 队列元素实例句柄

**返回值**： 返回FALSE表示hQueue为NULL，返回TRUE表示成功。

**说明**：

#### 获取队列元素数量

**函数**： CHNSYS\_INT OS\_QUEUE\_Count(OS\_QUEUE\_HANDLE hQueue);

**参数**： [in] hQueue 队列实例句柄

**返回值**： 返回-1表示表示hQueue为空，其他表示成功为元素数量。

**说明**：

### 文件部分

#### 初始化日志资源

**函数**： CHNSYS\_INT OS\_LOG\_Init();

**参数**： 无

**返回值**： 返回-1表示失败，返回0表示成功。

**说明**： 使用日志时必须先调用该函数进行初始化。

#### 释放日志资源

**函数**： CHNSYS\_INT OS\_LOG\_UnInit();

**参数**： 无

**返回值**： 返回-1表示失败，返回0表示成功。

**说明**： 使用日志完毕以后需要调用该函数释放资源否则会有内存泄漏。

#### 设置日志路径

**宏**： OS\_LOG\_SET\_PATH

**实例**：OS\_LOG\_SET\_PATH("D:\\Task\\log");

**说明**：

#### 设置日志级别

**宏**： OS\_LOG\_SET\_LEVEL

**实例**：OS\_LOG\_SET\_LEVEL(ZLog::LOG\_DEBUG);

**说明**：日志共有四个级别：LOG\_ERROR、LOG\_WARNING、LOG\_INFORMATION、LOG\_DEBUG

当设置等级为DEBUG时，日志文件中将打印所有等级的日志

当设置等级为INFO时，日志文件中将打印INFO，WARNING，ERROR等级的日志

当设置等级为WARNING时，日志文件中将打印WARNING，ERROR等级的日志

当设置等级为ERROR时，日志文件中将只打印ERROR等级的日志

#### 写日志

**宏**： LOG\_ERROR、LOG\_WARNING、LOG\_INFORMATION、LOG\_DEBUG四个级别

**实例**：LOG\_ERROR(("[main] Level = LOG\_WARNING, info = LOG\_ERROR\r\n"));

LOG\_WARNING(("[main] Level = LOG\_WARNING, info = LOG\_WARNING\r\n"));

LOG\_INFO(("[main] Level = LOG\_WARNING, info = LOG\_INFO\r\n"));

LOG\_DEBUG(("[main] Level = LOG\_WARNING, info = LOG\_DEBUG\r\n"));

**说明**：

#### 创建文件实例

**函数**： OS\_FILE\_HANDLE OS\_FILE\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_FILE\_IsOpen等函数的句柄参数。

**说明**： 无

#### 销毁文件实例

**函数**： VOID OS\_FILE\_DestroyInstance(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 无

**说明**： 无

#### 判断文件是否打开

**函数**： CHNSYS\_BOOL OS\_FILE\_IsOpen(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回FALSE表示hHandle为NULL或者文件没有被打开，返回TRUE表示已经打开

**说明**： 无

#### 判断文件路径是否正确

**函数**： CHNSYS\_BOOL OS\_FILE\_IsDir(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回FALSE表示hHandle为NULL或者路径错误，返回TRUE表示路径正确

**说明**： 无

#### 判断文件是否只读

**函数**： CHNSYS\_BOOL OS\_FILE\_IsReadOnly(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回FALSE表示hHandle为NULL或者文件非只读，返回TRUE表示文件为只读

**说明**： 无

#### 判断文件是否可读

**函数**： CHNSYS\_BOOL OS\_FILE\_IsReadable(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回FALSE表示hHandle为NULL或者文件不可读，返回TRUE表示文件为可读

**说明**： 无

#### 判断文件是否可写

**函数**： CHNSYS\_BOOL OS\_FILE\_IsWriteable(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回FALSE表示hHandle为NULL或者文件不可写，返回TRUE表示文件为可写

**说明**： 无

#### 打开文件

**函数**： CHNSYS\_BOOL OS\_FILE\_Open(OS\_FILE\_HANDLE hHandle, CONST CHNSYS\_CHAR\* sName, CHNSYS\_BOOL bReadOnly = TRUE, CHNSYS\_BOOL bAppend = FALSE);

**参数**： [in] hHandle 文件实例句柄

[in] sName 文件路径

[in] bReadOnly 是否以只读方式打开,默认为只读

[in] bAppend 每次写操作都写入文件的末尾，默认为否

**返回值**： 返回FALSE表示hHandle为NULL或者文件打开失败，返回TRUE表示打开成功

**说明**： 无

#### 创建文件

**函数**： CHNSYS\_BOOL OS\_FILE\_Create(OS\_FILE\_HANDLE hHandle, CONST CHNSYS\_CHAR\* sName, CHNSYS\_BOOL bCreateNew = TRUE);

**参数**： [in] hHandle 文件实例句柄

[in] sName 文件路径

[in] bCreateNew 是否指定文件不存在，则创建这个文件，默认为是

**返回值**： 返回FALSE表示hHandle为NULL或者文件创建失败，返回TRUE表示创建成功

**说明**： 无

#### 关闭文件

**函数**： CHNSYS\_BOOL OS\_FILE\_Close(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回FALSE表示hHandle为NULL或者文件关闭失败，返回TRUE表示关闭成功

**说明**： 无

#### 读文件

**函数**： CHNSYS\_INT OS\_FILE\_Read(OS\_FILE\_HANDLE hHandle, CHNSYS\_CHAR\* sdata, CHNSYS\_INT ndata);

**参数**： [in] hHandle 文件实例句柄

[out] sdata 数据内容

[in] ndata 数据大小

**返回值**： 返回-1表示hHandle为NULL，其他大于等于0值表示读取的数据大小。

**说明**：

#### 写文件

**函数**： CHNSYS\_INT OS\_FILE\_Write(OS\_FILE\_HANDLE hHandle, CONST CHNSYS\_CHAR\* sdata, CHNSYS\_INT ndata);

**参数**： [in] hHandle 文件实例句柄

[in] sdata 数据内容

[in] ndata 数据大小

**返回值**： 返回-1表示hHandle为NULL，其他大于等于0值表示写入的数据大小。

**说明**：

#### 定位文件

**函数**： CHNSYS\_INT OS\_FILE\_Seek(OS\_FILE\_HANDLE hHandle, CHNSYS\_UINT64 nPos);

**参数**： [in] hHandle 文件实例句柄

[in] nPos 位置

**返回值**： 返回-1表示hHandle为NULL或定位失败，返回0表示定位成功。

**说明**：

#### 获取文件句柄

**函数**： CHNSYS\_INT OS\_FILE\_GetHandle(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回-1表示hHandle为NULL或获取失败，其他返回当前的文件句柄。

**说明**：

#### 获取文件路径

**函数**： CHNSYS\_CHAR\* OS\_FILE\_GetPath(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回NULL表示hHandle为NULL或获取失败，其他返回当前的文件句柄路径。

**说明**：

#### 获取当前光标位置

**函数**： CHNSYS\_UINT64 OS\_FILE\_GetPosition(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

**返回值**： 返回-1表示hHandle为NULL，其他返回表示当前的光标位置。

**说明**：

#### 设置当前光标位置

**函数**： CHNSYS\_UINT64 OS\_FILE\_SetPosition(OS\_FILE\_HANDLE hHandle, CHNSYS\_UINT64 nPos);

**参数**： [in] hHandle 文件实例句柄

[in] nPos 位置

**返回值**： 返回-1表示hHandle为NULL，其他返回表示当前的光标位置。

**说明**：

#### 获取文件大小

**函数**： CHNSYS\_UINT64 OS\_FILE\_GetSize(OS\_FILE\_HANDLE hHandle);

**参数**： [in] hHandle 文件实例句柄

[in] nPos 位置

**返回值**： 返回-1表示hHandle为NULL，其他返回表示当前的文件大小。

**说明**：

### 网络部分

在此将Socket分成了两大类：UDPSocket，TCPSocket。

* 使用UDPSocket，可以方便创建一个UDP服务器端或UDP客户端程序。
* 使用TCPSocket，可以方便创建一个TCP服务器端或TCP客户端程序。

函数名中以OS\_SOCKET\_开头的函数，可以被UDPSocket及TCPSocket使用。

#### 网络初始化

**函数**： VOID OS\_SOCKET\_Init();

**参数**： 无

**返回值**： 无

**说明**： 使用。

#### 创建UDPSocket实例

**函数**： OS\_UDPSOCKET\_HANDLE OS\_UDPSOCKET\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_UDPSOCKET\_Create等函数的句柄参数。

**说明**： UDPSocket相关的接口用于完成UDP客户端、服务器端的程序。

#### 销毁UDPSocket实例

**函数**： VOID OS\_UDPSOCKET\_DestroyInstance(OS\_UDPSOCKET\_HANDLE hOSUdpSocket);

**参数**： [in] hOSUdpSocket UDPSocket实例句柄

**返回值**： 无

**说明**： 无

#### UDPSocket初始创建操作

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_Create(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_INT nAddrFamily = 2);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[in]nAddrFamily address family specification。

1-AF\_UNIX，2-AF\_INET。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无

#### UDPSocket关闭操作

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_Close(OS\_UDPSOCKET\_HANDLE hOSUdpSocket);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无

#### UDPSocket发送数据

**函数**： CHNSYS\_INT OS\_UDPSOCKET\_SendTo(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CONST CHNSYS\_CHAR\* sData, CONST CHNSYS\_INT nData, CHNSYS\_UINT nRemoteAddr, CHNSYS\_UINT nRemotePort);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[in]sData 发送的数据

[in]nData 发送的数据大小

[in]nRemoteAddr 发送到的IP地址。

[in]nRemotePort 发送到的端口

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### UDPSocket接收数据

**函数**： CHNSYS\_INT OS\_UDPSOCKET\_RecvFrom(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_CHAR\* sData, CHNSYS\_INT nData, CHNSYS\_UINT \*pRemoteAddr, CHNSYS\_UINT \*pRemotePort);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[out]sData 接收到的数据存放处

[in]nData 期望接收的数据大小

[out]pRemoteAddr 数据发送端的IP地址

[out]pRemotePort 数据发送端的端口

**返回值**： 返回实际接收到的数据大小。

返回-1，则有可能是UDPSocket被设置成非阻塞，此时没有数据可接收。或者是UDPSocket接收出错。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### UDPSocket获得远端地址

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_GetRemoteAddr(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_UINT \*pRemoteAddr, CHNSYS\_UINT \*pRemotePort);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[out]pRemoteAddr 数据发送端的IP地址

[out]pRemotePort 数据发送端的端口

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 在OS\_UDPSOCKET\_RecvFrom执行成功后调用此接口有效。通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### UDPSocket加入组播组

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_JoinMulticast(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_UINT nRemoteAddr);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[in]nRemoteAddr 要加入的组播地址

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### UDPSocket离开组播组

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_LeaveMulticast(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_UINT nRemoteAddr);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[in]nRemoteAddr 要离开的组播地址

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### UDPSocket设置组播关联接口

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_SetMulticastInterface(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_UINT nLocalAddr);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[in]nLocalAddr 组播关联的本地地址。设置为0表示所有接口。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### UDPSocket设置组播TTL属性

**函数**： CHNSYS\_BOOL OS\_UDPSOCKET\_SetMulticastTTL(OS\_UDPSOCKET\_HANDLE hOSUdpSocket, CHNSYS\_UINT nTTL);

**参数**： [in]hOSUdpSocket UDPSocket实例句柄

[in]nTTL TTL值。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无

#### Socket绑定端口

**函数**： CHNSYS\_BOOL OS\_SOCKET\_Bind(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_UINT nAddr, CHNSYS\_UINT nPort);

**参数**： [in]hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in]nAddr 绑定的接口地址。0表示本机的所有接口。

[in]nPort 绑定的端口号

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无

#### Socket设置阻塞模式

**函数**： CHNSYS\_BOOL OS\_SOCKET\_SetNonBlocking(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_BOOL bNonBlocking);

**参数**： [in]hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in]bNonBlocking 是否阻塞表示。TRUE表示非阻塞，FALSE表示阻塞

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： Socket默认为阻塞。

#### Socket设置地址重用属性

**函数**： CHNSYS\_BOOL OS\_SOCKET\_SetReuseAddr(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_BOOL bReuseAddr);

**参数**： [in]hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in]bReuseAddr 设置是否可重用。TRUE表示可重用，FALSE表示不可重用。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： Socket默认为不可重用。

#### Socket设置发送缓存大小

**函数**： CHNSYS\_BOOL OS\_SOCKET\_SetSendBufferSize(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_INT nBufferSize);

**参数**： [in]hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in]nBufferSize 发送缓存大小。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无。

#### Socket设置接收缓存大小

**函数**： CHNSYS\_BOOL OS\_SOCKET\_SetRecvBufferSize(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_INT nBufferSize);

**参数**： [in]hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in]nBufferSize 接收缓存大小。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无。

#### Socket设置发送接收超时

**函数**： CHNSYS\_BOOL OS\_SOCKET\_SetTimeout(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_INT nMilliSecSend, CHNSYS\_INT nMilliSecRevc);

**参数**： [in]hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in]nMilliSecSend 发送超时时间。以毫秒为单位。

[in]nMilliSecRecv 接收超时时间。以毫秒为单位。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无。

#### Socket地址转换-从字符串到整形

**函数**： CHNSYS\_UINT OS\_SOCKET\_ConvertAddrToUINT(CONST CHNSYS\_CHAR \*sAddr);

**参数**： [in]sAddr IP字符串格式。如：10.14.1.176

**返回值**： 返回整形格式的IP地址。

**说明**： 无。

#### Socket地址转换-从整形到字符串

**函数**： CHNSYS\_CHAR\* OS\_SOCKET\_ConvertAddrToSTRING(CHNSYS\_UINT nAddr);

**参数**： [in]nAddr IP整形格式。

**返回值**： 返回字符串格式的IP地址。

**说明**： 无。

#### Socket 获得底层socket句柄

**函数**： CHNSYS\_INT OS\_SOCKET\_GetLowSocket(OS\_SOCKET\_HANDLE hOSSocket);

**参数**： [in]hOSSocket UDP实例句柄，TCP实例句柄。

**返回值**： 返回底层的socket句柄。

**说明**： 通过返回底层的socket句柄，可做select等其他应用。

#### TCPSocket设置NoDelay属性

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_SetNoDelay(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_BOOL bNoDelay);

**参数**： [in]nOSSocket TCP实例句柄。

[in]bNoDelay TRUE表示NoDelay，FALSE表示Delay。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： Socket默认为NoDelay。

#### TCPSocket设置KeepAlive属性

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_SetKeepAlive(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_BOOL bKeepAlive);

**参数**： [in]nOSSocket TCP实例句柄。

[in]bKeepAlive TRUE表示KeepAlive，FALSE表示非KeepAlive。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： Socket默认为KeepAlive。

#### 创建TCPSocket实例

**函数**： OS\_TCPSOCKET\_HANDLE OS\_TCPSOCKET\_CreateInstance();

**参数**： 无

**返回值**： NULL表示执行失败，其他非0值表示执行成功，返回值作为OS\_TCPSOCKET\_Create等函数的句柄参数。

**说明**： TCPSocket相关的接口用于完成TCP客户端、服务器端的程序。

#### 销毁TCPSocket实例

**函数**： VOID OS\_TCPSOCKET\_DestroyInstance(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： 无。

**说明**： 无。

#### TCPSocket初始创建操作

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Create(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_INT nAddrFamily = 2);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]nAddrFamily address family specification。

1-AF\_UNIX，2-AF\_INET。

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无。

#### TCPSocket关闭操作

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Close(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无。

#### TCPSocket监听操作

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Listen(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_UINT nListenCount);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]nListenCount 最大支持挂起的连接数

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 无。

#### TCPSocket accept操作

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Accept(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_INT \*phSocket, CHNSYS\_UINT \*pRemoteAddr, CHNSYS\_UINT \*pRemotePort);

**参数**： [in]hOSTcpSocket TCP实例句柄

[out]phSocket 与TCP客户端连接交互的socket句柄

[out]pRemoteAddr TCP客户端的IP地址

[out]pRemotePort TCP客户端的端口

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。此函数执行后调用OS\_TCPSOCKET\_Create 及OS\_TCPSOCKET\_Attach函数来产生新的TCPSocket。

#### TCPSocket Connect操作

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Connect(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_UINT nAddr, CHNSYS\_UINT nPort);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]nAddr TCP服务器端的IP地址

[in]nPort TCP服务器端的端口

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### TCPSocket 是否可连接判断

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_ConnectAble(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_UINT nMilliSecond);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]nMilliSecond 等待是否可连接的时间。单位为毫秒

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： TCP客户端连接TCP服务器端，有时可能连接不上。为了限制连接等待的时间，先设置为no-block，然后执行OS\_TCPSOCKET\_Connect函数，再然后执行OS\_TCPSOCKET\_ConnectAble来等待指定时间。如果在指定的时间内返回连接成功，则立即返回TRUE。如果指定的时间到，仍然未连接成功，则返回FALSE。

#### TCPSocket Attach一个底层的socket句柄

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Attach(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_INT hSocket, CHNSYS\_UINT nRemoteAddr, CHNSYS\_UINT nRemotePort);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]hSocket 底层的socket句柄

[in]nRemoteAddr TCP客户端IP地址

[in]nRemotePort TCP客户端端口

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。此函数与OS\_TCPSOCKET\_Accept配合使用。

#### TCPSocket Detach一个底层的socket句柄

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_Detach(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 与OS\_TCPSOCKET\_Attach配套。

#### TCPSocket 发送数据

**函数**： CHNSYS\_INT OS\_TCPSOCKET\_Send(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CONST CHNSYS\_CHAR \*sData, CONST CHNSYS\_INT nData);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]sData 要发送的数据

[in]nData 要发送的数据大小

**返回值**： 实际发送的数据大小。

返回0表示连接断开；

返回-1表示此次发送失败，但可再尝试发送。可能原因是发送太快。

返回-2表示发送出错；

返回-100表示hOSTcpSocket为NULL

**说明**： 无。

#### TCPSocket 接收数据

**函数**： CHNSYS\_INT OS\_TCPSOCKET\_Recv(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_CHAR \*sData, CONST CHNSYS\_INT nData);

**参数**： [in]hOSTcpSocket TCP实例句柄

[out]sData 接收到的数据存放处

[in]nData 期望接收的数据大小

**返回值**： 实际接收到的数据大小。

返回0表示连接断开；

返回-1表示无数据可接收。

返回-2表示接收出错；

返回-100表示hOSTcpSocket为NULL

**说明**： 无。

#### TCPSocket 获得远端地址

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_GetRemoteAddr(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_UINT \*pRemoteAddr, CHNSYS\_UINT \*pRemotePort);

**参数**： [in]hOSTcpSocket TCP实例句柄

[out]pRemoteAddr 与此TCP连接的对端的IP地址

[out]pRemotePort 与此TCP连接的对端的端口

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 通过OS\_SOCKET\_ConvertAddrToUINT函数及OS\_SOCKET\_ConvertAddrToSTRING 函数完成地址在INT与字符串之间的转换。

#### TCPSocket 设置流标志

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_StreamCreate(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 设置为流式后，OS\_TCPSOCKET\_StreamRead函数在TCPSocket为阻塞的情况下，会读取到指定的数据大小才返回。

#### TCPSocket取消流标志

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_StreamClose(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示执行成功，返回FALSE表示执行失败。

**说明**： 与OS\_TCPSOCKET\_StreamCreate对应。

#### TCPSocket特殊流式读数据

**函数**： CHNSYS\_INT OS\_TCPSOCKET\_StreamRead(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CHNSYS\_CHAR \*sData, CHNSYS\_INT nData);

**参数**： [in]hOSTcpSocket TCP实例句柄

[out]sData 接收到的数据存放处

[in]nData 期望接收的数据大小

**返回值**： 实际接收到的数据大小。

返回0表示连接断开；

返回-1表示无数据可接收；

返回-2表示接收出错；

返回-100表示hOSTcpSocket为NULL；

**说明**： 如果执行过OS\_TCPSOCKET\_StreamCreate函数且socket为阻塞模式，那么调用此函数时，会循环接收数据，直到收到nData数据大小或接收出错为止。如果未执行过OS\_TCPSOCKET\_StreamCreate函数，则此函数与OS\_TCPSOCKET\_Recv功能相同。

#### TCPSocket特殊流式写数据

**函数**： CHNSYS\_INT OS\_TCPSOCKET\_StreamWrite(OS\_TCPSOCKET\_HANDLE hOSTcpSocket, CONST CHNSYS\_CHAR \*sData, CHNSYS\_INT nData);

**参数**： [in]hOSTcpSocket TCP实例句柄

[in]sData 要发送的数据

[in]nData 要发送的数据大小

**返回值**： 实际发送的数据大小。

返回0表示连接断开；

返回-1表示此次发送失败，但可再尝试发送。可能原因是发送太快。

返回-2表示发送出错；

返回-100表示hOSTcpSocket为NULL；

**说明**： 如果执行过OS\_TCPSOCKET\_StreamCreate函数且socket为阻塞模式，那么调用此函数时，会循环发送数据，直到发送了nData数据大小或发送出错为止。如果未执行过OS\_TCPSOCKET\_StreamCreate函数，则此函数与OS\_TCPSOCKET\_Send功能相同。

#### TCPSocket判断是否可读

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_StreamReadable(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示可读，返回FALSE表示不可读。

**说明**： 此函数内部使用select判断是否可读，判断时间为1毫秒。

#### TCPSocket判断是否可写

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_StreamWriteable(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示可写，返回FALSE表示不可写。

**说明**： 此函数内部使用select判断是否可读，判断时间为1毫秒。

#### TCPSocket判断是否有错误

**函数**： CHNSYS\_BOOL OS\_TCPSOCKET\_StreamError(OS\_TCPSOCKET\_HANDLE hOSTcpSocket);

**参数**： [in]hOSTcpSocket TCP实例句柄

**返回值**： TRUE表示有错误，返回FALSE表示无错误。

**说明**： 此函数内部使用select判断是否有错误，判断时间为10毫秒

### Task部分

#### 初始化线程池资源

**函数**： VOID TASK\_POOL\_Init(CHNSYS\_INT nTotalCount, CHNSYS\_INT nExclusiveCount = 1);

**参数**： [in] nTotalCount 线程池数量

[in] nExclusiveCount 特有线程数量

**返回值**： 无

**说明**： 线程池分为特用线程与公用线程2部分，公用线程是所有任务都可以用的线程，特用线程只有指定的线程可以使用，目前0号线程已经被使用，其他特用线程需要从1开始，因此nExclusiveCount默认为1。

比如：

使用者有任务A，任务B，任务C，一大堆其他任务。其中任务A，任务B，任务C希望能独占线程。

如下：

0 1 2 3 4 5 6 7 8 9 10 11

| A | B| C| 共用 |

调用方式为：

TASK\_POOL\_Init(12, 5);

// Task内部占用0号线程

// TaskA独占1,2号线程

hTaskA = TASK\_TASK\_CreateInstance(1, 3);

// TaskB独占3号线程

hTaskB = TASK\_TASK\_CreateInstance(3, 4);

// TaskC独占4号线程

hTaskC = TASK\_TASK\_CreateInstance(4, 5);

hOtherTask = TASK\_TASK\_CreateInstance(-1, -1);

#### 释放线程池资源

**函数**： VOID TASK\_POOL\_Uninit();

**参数**： 无

**返回值**： 无

**说明**： 与TASK\_POOL\_Init对应。

#### 创建任务实例

**函数**： TASK\_TASK\_HANDLE TASK\_TASK\_CreateInstance(CHNSYS\_INT nThreadBeginNo = -1, CHNSYS\_INT nThreadEndNo = -1);

**参数**： [in] nThreadBeginNo 开始线程号

[in] nThreadEndNo 结束线程号

**返回值**： 返回Task实例句柄，作为TASK\_TASK\_Create等函数的句柄

**说明**： 由于0号线程已经被占用，所以创建任务实例时指定特用线程只能从1开始，如指定用1号线程则nThreadBeginNo为1，nThreadEndNo为2.

#### 销毁任务实例

**函数**： VOID TASK\_TASK\_DestroyInstance(TASK\_TASK\_HANDLE hTask);

**参数**： [in] hTask Task实例句柄

**返回值**： 无

**说明**：

#### 开始任务

**函数**： CHNSYS\_BOOL TASK\_TASK\_Create(TASK\_TASK\_HANDLE hTask);

**参数**： [in] hTask Task实例句柄

**返回值**： 返回FALSE表示hTask为NULL或创建任务失败，返回TRUE表示创建成功

**说明**：

#### 结束任务

**函数**： CHNSYS\_BOOL TASK\_TASK\_Close(TASK\_TASK\_HANDLE hTask);

**参数**： [in] hTask Task实例句柄

**返回值**： 返回FALSE表示hTask为NULL或关闭任务失败，返回TRUE表示关闭成功

**说明**：

#### 任务回调函数

**函数**： typedef CHNSYS\_INT (\*TASK\_TASK\_CallBackFunc)(CHNSYS\_INT nEventType, VOID\* pContext);

**参数**： [out] nEventType 事件类型

[out] pContext 上下文参数

**返回值**： 当返回值大于0时，该任务触发TASK\_EVENT\_UPDATE事件，触发UPDATE事件的时间为返回值的毫秒数。如返回900，则900毫秒后此回调还会收到UPDATE事件。返回<=0的值，无特殊作用。

**说明**： 事件类型包括：

TASK\_EVENT\_READ：

如果Task和Socket绑定，且执行OS\_SOCKET\_RequestEvent，那么当Socket可读时，将触发此回调函数，对应的nEventType=TASK\_EVENT\_READ。使用者也可以通过调用TASK\_TASK\_AddEvent来给Task设置TASK\_EVENT\_READ事件。

TASK\_EVENT\_WRITE：

暂未使用。

TASK\_EVENT\_START：

TASK\_EVENT\_STOP：

TASK\_EVENT\_TIMEOUT：

如果调用TASK\_TASK\_SetTimeout函数给Task设置了超时，那么在超时时间到后，将触发此回调函数，对应的nEventType= TASK\_EVENT\_TIMEOUT。

TASK\_EVENT\_KILL：

TASK\_EVENT\_UPDATE：

此事件通过TASK\_TASK\_AddEvent触发，也可以通过回调返回>0值来自触发。

TASK\_EVENT\_IDLE：

如果调用TASK\_TASK\_AddIdleTask函数给Task设置了Idle时间，那么在Idle时间到后，将触发此回调函数，对应的nEventType= TASK\_EVENT\_IDLE。

#### 设置任务回调函数

**函数**： VOID TASK\_TASK\_SetCallBack(TASK\_TASK\_HANDLE hTask, TASK\_TASK\_CallBackFunc cbf, VOID\* pContext);

**参数**： [in] hTask Task实例句柄

[in] cbf 任务回调函数

[in] pContext 上下文参数

**返回值**： 无

**说明**： 无

#### 添加事件

**函数**： CHNSYS\_BOOL TASK\_TASK\_AddEvent(TASK\_TASK\_HANDLE hTask, CHNSYS\_INT nTaskEvent);

**参数**： [in] hTask Task实例句柄

[in] nTaskEvent 事件

**返回值**： 返回FALSE表示hTask为NULL或添加事件失败，返回TRUE表示添加事件成功

**说明**： 添加完事件后，Task的回调函数会被触发。

#### 添加空闲任务

**函数**： CHNSYS\_BOOL TASK\_TASK\_AddIdleTask(TASK\_TASK\_HANDLE hTask, CHNSYS\_INT nMilliSec);

**参数**： [in] hTask Task实例句柄

[in] nMilliSec 时间

**返回值**： 返回FALSE表示hTask为NULL或添加失败，返回TRUE表示添加成功。

**说明**： 设置的nMilliSec表示在每过nMilliSec时间hTask会收到一个TASK\_IDLE\_EVENT事件

#### 移除空闲任务

**函数**： CHNSYS\_BOOL TASK\_TASK\_RemoveIdleTask(TASK\_TASK\_HANDLE hTask);

**参数**： [in] hTask Task实例句柄

**返回值**： 返回FALSE表示hTask为NULL或移除失败，返回TRUE表示移除成功。

**说明**：

#### 设置任务超时时间

**函数**： CHNSYS\_BOOL TASK\_TASK\_SetTimeout(TASK\_TASK\_HANDLE hTask, CHNSYS\_INT nMilliSec);

**参数**： [in] hTask Task实例句柄

[in] nMilliSec 时间

**返回值**： 返回FALSE表示hTask为NULL或设置失败，返回TRUE表示设置成功。

**说明**： 对应回调函数的TASK\_EVENT\_TIMEOUT事件

#### 重置任务超时时间

**函数**： CHNSYS\_BOOL TASK\_TASK\_RefreshTimeout(TASK\_TASK\_HANDLE hTask);

**参数**： [in] hTask Task实例句柄

**返回值**： 返回FALSE表示hTask为NULL或重置失败，返回TRUE表示重置成功。

**说明**： 该函数是将时间从0开始重新计算超时，如设置5秒超时，经过3秒后调用该函数则再次经过5秒才算超时。

#### SOCKET绑定任务

**函数**： CHNSYS\_BOOL OS\_SOCKET\_SetTask(OS\_SOCKET\_HANDLE hOSSocket, TASK\_TASK\_HANDLE hTask);

**参数**： [in] hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in] hTask Task实例句柄

**返回值**： 返回FALSE表示hTask或hOSSocket为NULL或绑定失败，返回TRUE表示绑定成功。

**说明**： SOCKET绑定任务后，SOCKET收到的读写事件都会由Task的回调函数进行处理。

注:不能多个SOCKET绑定同一个任务。

#### 向SOCKET添加事件

**函数**： CHNSYS\_BOOL OS\_SOCKET\_RequestEvent(OS\_SOCKET\_HANDLE hOSSocket, CHNSYS\_INT nEventType);

**参数**： [in] hOSSocket Socket实例句柄，包括UDP实例句柄，TCP实例句柄

[in] nEventType 事件

**返回值**： 返回FALSE表示hTask为NULL或请求失败，返回TRUE表示请求成功。

**说明**： SOCKET收到某一事件并处理以后，必须再次添加该事件，SOCKET才会继续监听该事件，否则不会再对该事件进行监听处理。

### 常用工具函数

#### 获取当前时间(毫秒)

**函数**： CHNSYS\_UINT64 OS\_UTIL\_Milliseconds();

**参数**： 无

**返回值**： 返回当前时间。

**说明**：

#### 获取当前时间(微秒)

**函数**： CHNSYS\_UINT64 OS\_UTIL\_Microseconds();

**参数**： 无

**返回值**： 返回当前时间。

**说明**：

#### 睡眠指定时间(微秒)

**函数**： VOID OS\_UTIL\_SleepMicroseconds(CHNSYS\_INT nMicroSecond);

**参数**： 无

**返回值**： 返回值为。

**说明**：

#### 字母转小写

**函数**： CHNSYS\_CHAR \*OS\_UTIL\_Lower(CHNSYS\_CHAR \*sValue);

**参数**： [in] sValue 要转换的内容

**返回值**： 返回转换后的内容。

**说明**：

#### 字母转大写

**函数**： CHNSYS\_CHAR \*OS\_UTIL\_Upper(CHNSYS\_CHAR \*sValue);

**参数**： [in] sValue 要转换的内容

**返回值**： 返回转换后的内容。

**说明**：

#### 检查目录

**函数**： CHNSYS\_INT OS\_UTIL\_CheckDir(CONST CHNSYS\_CHAR \*sDir);

**参数**： [in] sDir 路径

**返回值**： 返回-1表示sDir为NULL，返回0表示路径已经存在，返回1表示文件已经存在。

**说明**：

#### 创建目录

**函数**： CHNSYS\_INT OS\_UTIL\_RecursiveMakeDir(CONST CHNSYS\_CHAR \*sDir);

**参数**： [in] sDir 路径

**返回值**： 返回-1表示创建失败，返回0表示创建成功。

**说明**： 该函数可以创建多级目录

#### 解析URL地址

**函数**： CHNSYS\_BOOL OS\_UTIL\_DecodeURL(CHNSYS\_CHAR \*sURL, CHNSYS\_CHAR \*\*sProtocol,

CHNSYS\_CHAR \*\*sHost = NULL, CHNSYS\_CHAR \*\*sPort = NULL,

CHNSYS\_CHAR \*\*sUser = NULL, CHNSYS\_CHAR \*\*sPass = NULL,

CHNSYS\_CHAR \*\*sPath = NULL);

**参数**： [in] sURL 路径

[out] sProtocol 协议名

[out] sHost 主机

[out] sPort 端口

[out] sUser 用户名

[out] sPass 密码

[out] sPath 路径

**返回值**： 返回FALSE表示解析失败，返回TRUE表示解析成功。

**说明**： 该函数针对固定格式URL进行解析,格式如下所示：protocol:username:password@host:port/path