**TCBSD下C++ ADS sum-cmd批量读取变量**

|  |  |
| --- | --- |
|  | 作者：范小军职务：风电 技术工程师邮箱：x.fan@beckhoff.com.cn日期：2023-04-12 |
| **摘 要：**在TCBSD下使用ADS sum-cmd通过变量的名称读取变量数值。ADS提供强大而快速的通信来交换任何类型的信息,每个ADS-API调用都可以读取单个变量或完整的数组和结构。但是ADS通讯每次通讯请求都会占用很长的时间，一次请求读写一个变量和读写多个变量的时间相近，所以在使用sum cmd命令读写变量的时候优势很明显，可以一次请求完成多个变量的读写操作。  |
| **附 件：**

|  |  |  |
| --- | --- | --- |
| 序 号 | 文件名 | 备注 |
| 1 | TCBSD Sum-Cmd\_PLC.zip | PLC |
| 2. | sum-cmd.cpp | C++ |
|  |  |  |
|  |  |  |

 |
| **历史版本：**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

 |
| **免责声明：**我们已对本文档描述的内容做测试。但是差错在所难免，无法保证绝对正确并完全满足您的使用需求。本文档的内容可能随时更新，如有改动，恕不事先通知，也欢迎您提出改进建议。 |
| **参考信息：** |

目 录

[1. Sum-cmd介绍 3](#_Toc132222102)

[1.1. ADS通讯 3](#_Toc132222103)

[1.2. Sum Cmd ADS Index Group说明 3](#_Toc132222104)

[2. TCBSD系统中批量读程序的解析（以变量名读取为例） 4](#_Toc132222105)

[2.1. 过程简介： 4](#_Toc132222106)

[2.2. TCBSD系统中数据类型的介绍 4](#_Toc132222107)

[2.3. 通过变量名获取变量的ADS symbol information信息 5](#_Toc132222108)

[2.3.1. ADS symbol information 5](#_Toc132222109)

[2.3.2. AdsSyncReadWriteReq函数的说明 6](#_Toc132222110)

[2.3.3. 用AdsSyncReadWriteReq函数通过变量名方式获取ads symbol的信息 7](#_Toc132222111)

[2.4. 设置AdsSyncReadWriteReq函数的参数及pWriteData数据的打包 8](#_Toc132222112)

[2.4.1. pWriteData参数的设定（数据的打包） 8](#_Toc132222113)

[2.4.2. cbWriteLength参数的设定 10](#_Toc132222114)

[2.4.3. cbReadLength参数设定 10](#_Toc132222115)

[2.5. 用AdsSyncReadWriteReq批量读取 11](#_Toc132222116)

[2.6. 检查执行结果中的错误信息 11](#_Toc132222117)

[3. TCBSD系统中C++文件的编译、连接动态连接库、运行生成的可执行文件 12](#_Toc132222118)

[3.1. TcAdsAPI.h 、TcAdsDef.h和libTcAdsDll.so的文件位置 12](#_Toc132222119)

[3.2. 激活PLC程序 13](#_Toc132222120)

[3.3. 生成执行文件的过程 14](#_Toc132222121)

[4. 附件： 16](#_Toc132222122)

[4.1. 常见的报错信息 16](#_Toc132222123)

[4.2. Plc和高级语言变量对应 17](#_Toc132222124)

[4.3. WinSCP软件的设置 17](#_Toc132222125)

# Sum-cmd介绍

## ADS通讯

 TCBSD系统和Windows系统一样支持ADS通讯，Ads通讯不是一个实时的通讯协议，其通讯过程受到系统和网络状况的多重影响，无法保证一个稳定的通讯时间，ADS 通讯每次请求会占用较多时间，但是一次请求读取或写入一个变量和一次请求读取或写入多个变量所花费的时间相近，此时可以采用Sum Cmd将所有需要读取或写入的变量写入字节流，可以一次请求读写多个变量，以便于多个数据周期性的读写操作，且所需的延迟会更少。

## Sum Cmd ADS Index Group说明

 0xF080：ADSIGRP\_SUMUP\_READ 批量读取

 0xF081：ADSIGRP\_SUMUP\_WRITE 批量写入

 0xF082：ADSIGRP\_SUMUP\_READWRITE 批量读写；

 常见的ADS Index Group：





Infosys:

https://infosys.beckhoff.com/english.php?content=../content/1033/tcadscommon/12495372427.html&id=

# TCBSD系统中批量读程序的解析（以变量名读取为例）

## 过程简介：

第一步：获取单个变量名的ADS symbol information信息；

第二步：数据的打包及AdsSyncReadWriteReq函数的参数设定

第三步：批量读取。

第四步：检查报错信息。

## TCBSD系统中数据类型的介绍

在TcAdsDef.h中如果使用POSIX，那么C++代码中的数据结构也需要做一定的调整

#ifdef POSIX

 #include <stdint.h>

 typedef int64\_t ads\_i64;

 typedef int ads\_i32;

 typedef unsigned int ads\_ui32;

#else

 typedef \_\_int64 ads\_i64;

 typedef long ads\_i32;

 typedef unsigned long ads\_ui32;

#endif

例如：之前在windows下定义的unsigned long在TCBSD下就用unsigned int，long 就需要用int。

其他类型的数据如下：

typedef char ads\_i8;

typedef unsigned char ads\_ui8;

typedef unsigned short ads\_ui16;

typedef int ads\_bool;

## 通过变量名获取变量的ADS symbol information信息

### ADS symbol information

ADS symbol information是PLC变量的说明，其中包含了entryLength、iGroup、iOffs、size等信息，后面将会把获取的这些信息进行打包处理。

详细解释如下：

 例如：PLC程序中定义了一下的两个变量，



 var01:INT :=100;

 var02:INT :=200;

那么这个变量在ADS系统中有ADS信息，也就是ADS symbol information，在TCBSD体统中提供的ADS symbol信息如下（在TcAdsDef.h头文件中定义）：

typedef struct

{

 ads\_ui32 entryLength; // length of complete symbol entry

 ads\_ui32 iGroup; // indexGroup of symbol: input, output etc.

 ads\_ui32 iOffs; // indexOffset of symbol

 ads\_ui32 size; // size of symbol ( in bytes, 0 = bit )

 ads\_ui32 dataType; // adsDataType of symbol

 ads\_ui32 flags; // see above

 ads\_ui16 nameLength; // length of symbol name (excl. \0)

 ads\_ui16 typeLength; // length of type name (excl. \0)

 ads\_ui16 commentLength; // length of comment (excl. \0)

 // ADS\_INT8 name[]; // name of symbol with terminating \0

 // ADS\_INT8 type[]; // type name of symbol with terminating \0

 // ADS\_INT8 comment[]; // comment of symbol with terminating \0

} AdsSymbolEntry, \*PAdsSymbolEntry, \*\*PPAdsSymbolEntry;

可以通过iGroup 、iOffset 、size等信息用ADS读写函数访问到对应的变量，在配置中我们可以看到对应变量的ADS info的信息。



### AdsSyncReadWriteReq函数的说明

在TCBSD下AdsSyncReadWriteReq函数如下，可以根据不同的Index Group我们可以获取到不同的信息。

TCADSDLL\_EXPORT

ads\_i32 TCADSDLL\_API AdsSyncReadWriteReq

(

AmsAddr\* pAddr, // Ams address of ADS server

ads\_ui32 indexGroup, // index group in ADS server interface

ads\_ui32 indexOffset, // index offset in ADS server interface

ads\_ui32 cbReadLength, // count of bytes to read

void\* pReadData, // pointer to the client buffer

ads\_ui32 cbWriteLength, // count of bytes to write

void\* pWriteData // pointer to the client buffer

);

参数如下：

pAddr [in] Structure with NetId and port number of the ADS server.

nIndexGroup [in] Index Group.

nIndexOffset [in] Index Offset.

nReadLength [in] Length of the data, in bytes, returned by the ADS device.

pReadData [out] Buffer with data returned by the ADS device.

nWriteLength [in] Length of the data, in bytes, written to the ADS device.

pWriteData [out] Buffer with data written to the ADS device.

Return value Returns the function's error status.

在此函数中Index Group的数值不同读取到的数据也不同。

例如：

**Index Groups：**ADSIGRP\_SYM\_INFOBYNAME,16#F009，此时是通过变量名获取到ads symbol information信息，其主要信息是给定变量的indexGroup，indexOffset，length等信息，在此示例中使用的是通过变量名的方式获取ads symbol information信息，并打包进行批量的读操作。

当然Index Group可以是ADSIGRP\_SYM\_HNDBYNAME ，16#F003，以变量名的方式获取变量的句柄信息。然后通过句柄批量读写变量。

 常见的ADS/System Services的index Group信息如下：



### 用AdsSyncReadWriteReq函数通过变量名方式获取ads symbol的信息

通过AdsSyncReadWriteReq可以获取到ads symbol information的信息，

如下操作：

1.定义变量存放PLC变量的名称

 char szVar7[] = { "MAIN.var01" }; ///PLC中声明的变量名称var01:INT :=100;

 char szVar8[] = { "MAIN.var02" }; ///PLC中声明的变量名称var02:real: = 200;

注意：在TC3中PLC的变量名称必须加Program的名称.变量名，例如：

PROGRAM MAIN

VAR

 var01:INT :=100;

 var02:INT :=200;

END\_VAR

 在MAIN中定义了变量，那么在使用的时候必须要加MAIN.var01。

2.定义结构体存放变量的主要Ads symbol information信息

 typedef struct dataPar

 {

 ads\_ui32 indexGroup; // index group in ADS server interface

 ads\_ui32 indexOffset; // index offset in ADS server interface

 ads\_ui32 l ength; // count of bytes to read

 } TDataPar;

3.声明通过AdsSyncReadWriteReq函数读取到变量的symbol信息的结构体（这个结构体能够读取到变量的所有ads symbol information信息）。

AdsSymbolEntry这个结构体在TcAdsDef.h已经定义，只需要定义AdsSymbolEntry类型的变量即可，并新建指针指向该变量所在的地址（因为AdsSyncReadWriteReq需要的就是指针类型的数据）。

如下：

AdsSymbolEntry InfoExFloat; //存放用AdsSyncReadWriteReq读取到的变量的信息

 AdsSymbolEntry\* pInfoExFloat = &InfoExFloat;//指针指向InfoExFloat的地址

4．通过AdsSyncReadWriteReq函数读取symbol信息。

 /\*调用AdsSyncReadWriteReq函数， 获取 szVar7也就是 "MAIN.var01"的变量的信息

 （indexGroup，indexOffset，length等信息）存放在pInfoExFloat指针的指向的地址中\*/

 nErr = AdsSyncReadWriteReq(pAddr,

 ADSIGRP\_SYM\_INFOBYNAMEEX,//16#F007 通过变量名获取symbol info

 0x0,

 sizeof(InfoExFloat),

 pInfoExFloat,

 sizeof(szVar7),

 szVar7);

通过以上就可以通过变量名获取到对应变量的symbol信息（主要是indexGroup，indexOffset，length）；对于其他的变量也是同样的操作，当然后面可以通过循环去获取。

获取到变量的 ads symbol information信息以后我们进行数据得打包，用于批量读写变量。

## 设置AdsSyncReadWriteReq函数的参数及pWriteData数据的打包

数据的打包其是就是AdsSyncReadWriteReq函数的pWriteData参数的设定

参数的设定主要是以下的参数：

pWriteData：打包好的数据的地址

cbReadLength：读取的长度

cbWriteLength：写入的长度

pReadData：读取到的批量数据

### pWriteData参数的设定（数据的打包）

pWriteData参数是一个指针类型的参数，写入的数据有相应的格式要求，因为是批量的读写，所以需要将单个变量的ads symbol information信息进行打包处理。

 1.单个数据的结构如下：





包括indexGroup，indexOffset，Length。也就是IG、IO、Len。

详细解释如下：

|  |  |  |
| --- | --- | --- |
| 内容  | 长度  | 描述 |
| 首地址  | 4 bytes  | 数据写入的位置，也表示需要执行的命令类型 |
| 偏移量  | 4 bytes  | 数据写入的位置，也表示需要执行的命令参数 |
| 长度  | 4 bytes  | 以字节为单位，表示需要被写入的字节长度 |

在C++中定义一个TDataPar结构体方式实现单个数据，如上3.3介绍用AdsSyncReadWriteReq函数通过变量名方式获取ads symbol的信息

2.pWriteData参数的打包

pWriteData参数根据变量的个数进行打包，如下是读取两个变量的打包后的格式。

打包步骤如下：

1）定义结构体用来存放获取的变量名的ads symbol information信息。

typedef struct dataPar

 {

 ads\_ui32 indexGroup; // index group in ADS server interface

 ads\_ui32g indexOffset; // index offset in ADS server interface

 ads\_ui32 length; // count of bytes to read

 } TDataPar;

2）定义结构体数组，根据使用的情况定义大小。

 TDataPar parReq[2]; //定义结构体数组，存放对应两个PLC变量的数据。

3）获取到的ads symbol信息放在结构体数组中

 /\* build first entry inside request buffer \*/

 parReq[0].indexGroup = pAdsSymbolEntry->iGroup;//给写入请求的结构体赋值，将信息打包

 parReq[0].indexOffset = pAdsSymbolEntry->iOffs;

 parReq[0].length = pAdsSymbolEntry->size;

 /\* build second entry inside request buffer \*/

 parReq[1].indexGroup = pAdsSymbolEntry->iGroup;//请求数据的打包

 parReq[1].indexOffset = pAdsSymbolEntry->iOffs;

 parReq[1].length = pAdsSymbolEntry->size;

### cbWriteLength参数的设定

 cbWriteLength是写入给ADS设备的数据大小。

写入长度由的计算：



用indexGroup为0xF080写入数据，也就是写入结构体的数据，如下是结构体：

typedef struct dataPar

 {

 ads\_ui32 indexGroup; // index group in ADS server interface

 ads\_ui32 indexOffset; // index offset in ADS server interface

 ads\_ui32 length; // count of bytes to read

 } TDataPar;

ads\_ui32 占用4个字节，有indexGroup 、indexOffset 、length 三个ads\_ui32的数据，所以大小为3\*4=12个字节，同时如果有n个变量需要读取，那么cbWriteLength的计算方式就是：12\*n；

其中n代表请求的变量个数

### cbReadLength参数设定

cbReadLength是从ADS设备返回的数据长度。



读取长度由如下两部分组成：执行结果（错误标识）占用的字节大小和ADS设备中读取到的数据占用的字节大小。

执行结果（错误标识）就是返回的数据（Result）其数据类型为ads\_ui32，占用4个字节，如果有n个读取的变量，则是4\*n

ADS设备中读取到的数据占用的字节大小，根据读取变量占用内存的大小来设置。

比如：读取的数据是int类型（PLC 中定义 var01:INT :=100;）：在C++中int对应是short类型占用2个字节。如果PLC中Real对应C++中的float数据类型占用4字节；在实例中采用的是两int的类型的变量，那么就是n\*2；

当然在使用AdsSyncReadWriteReq函数根据变量名方式获取到的ADS symbol信息中的length就是对应的变量的长度。所以可以使用int reqSize = parReq[0].length + parReq[1].length + parReq[2].length;这种方式获取变量占用内存的大小

计算得到的cbReadLength的长度为：4\*n+ reqSize；

其中n代表请求的变量个数

## 用AdsSyncReadWriteReq批量读取

批量读设置index Group为0xF080；将上面设置好的参数写入AdsSyncReadWriteReq函数的参数并执行函数。

 nErr = AdsSyncReadWriteReq

(

pAddr,

0xf080,//读取的指令 0xf080 读 0xf081 写 0xf082读写同时

reqNum, ///读取的变量个数

4 \* reqNum + reqSize, //// n\*4+ parReq[0].length + parReq[1].length cbReadLength

(void\*)(mAdsSumBufferRes), //存放读取回来的数据的数据的地址

12 \* reqNum, ///// n\*（4+4+4） cbWriteLength

&parReq //打包的所有变量的结构体的数据

);

执行完函数以后读取到的批量数据存放在mAdsSumBufferRes数组中。

ads\_ui8 mAdsSumBufferRes[200];

## 检查执行结果中的错误信息

检查读取过程中的执行结果来判断每一个变量是否有错误。





在完成读取以后需要将读取到的数据地位ADS-err的位置

 /\* 0x0 returned !!! \*/

pObjAdsRes是指向数据的地址

 ads\_ui8\* pObjAdsRes = (ads\_ui8\*)mAdsSumBufferRes + (reqNum \* 4); // point to ADS-data

pObjAdsErrRes指针就是指向错误的地址

ads\_ui8\* pObjAdsErrRes = (ads\_ui8\*)mAdsSumBufferRes; // point to ADS-err

 读取完成以后判断返回值pObjAdsErrRes有没有错误。

 if (nErr == 0)//

 {

 for (int idx = 0; idx < reqNum; idx++)

 {

 // was communication for ADS-sub command OK ??

 int nAdsErr = \*(int\*)pObjAdsErrRes;

 if (nAdsErr == 0)

 {

 // get data out of stream

 }

 pObjAdsErrRes = pObjAdsErrRes + 4; // point to next ADS-err object

 pObjAdsRes = pObjAdsRes + 4;

 }

 }

# TCBSD系统中C++文件的编译、连接动态连接库、运行生成可执行文件

## TcAdsAPI.h 、TcAdsDef.h和libTcAdsDll.so的文件位置

在TCBSD系统中TcAdsAPI.h 和TcAdsDef.h在usr/local/include路径下，libTcAdsDll.so

是动态连接库文件在/usr/local/lib/下。

TcAdsAPI.h 和TcAdsDef.h文件位置：



（通过WinSCP软件连接到TCBSD系统查看文件的结构）

libTcAdsDll.so位置：



（通过WinSCP软件连接到TCBSD系统查看文件的结构）

## 激活PLC程序

添加TCBSD的路由，需要在TCBSD下关闭防火墙，通过doas service pf stop命令关闭TCBSD的防火墙。



添加路由信息，并激活程序。

 

## TCBSD下安装程序开发包

为了能编译C/C++代码，需要在TCBSD下安装os-generic-userland-devtools 的开发包，使用命令 doas pkg install os-generic-userland-devtools。



附件：os-generic-userland-devtools.pkg开发包，需要放在TCBSD下安装。

##  生成执行文件的过程

1. 将编辑好的sum-cmd.cpp文件通过WinSCP拷贝到对应的路径下，例如/usr/local/ADSinterface文件夹下：



2.在TCBSD系统下通过cd命令导航到对应的文件夹下，并用ls查看文件夹下是否已经有对应文件。



1. 编译C++文件，通过doas cc -c -I /usr/local/include/ -D POSIX sum-cmd.cpp -o sum-cmd.o命令编译sum-cmd.cpp文件生成sum-cmd.o文件，用ls命令查看文件生成情况。



4.连接ADS动态连接库生成可执行文件。通过命令：cc -lpthread sum-cmd.o /usr/
local/lib/libTcAdsDll.so -o sum-cmd，用ls命令查看文件生成情况。

 

 可以看到已经生成了sum-cmd可执行文件。

5.运行可以执行文件，通过命令 ./sum-cmd



执行可执行文件后可以读取到对应的变量的数值 var01:INT :=100; var02:INT :=200;

# 附件：

## 常见的报错信息

|  |  |  |
| --- | --- | --- |
| **ADS错误代码** | **描述** | **建议** |
| 0x745(1861) | 通讯超时 | 常见故障，检测连线、路由设置、防火墙设置、服务端TwinCAT是否已经启动 |
| 0x710(1808) | 对象没有找到 | 检测变量名称是否正确，注意全局变量为：.变量名 |
| 0x705(1797) | 参数长度错误 | 常见于批处理，检查计算命令长度 |
| 0x707(1799) | 设备未就绪 | 可能为tsm配置不正确，重新激活配置 |

## Plc和高级语言变量对应



## WinSCP软件的设置

WinSCP软件下载：<https://winscp.net/>

安装完成，

通过ifconfig查看TCBSD的IP地址信息，



新建站点,输入IP地址、端口（端口号：22）、用户名和密码。



编辑->高级->在SFTP中服务器设置：doas /usr/libexec/sftp-server



**上海（ 中国区总部）**

中国上海市静安区汶水路 299 弄 9号（市北智汇园）

电话: 021-66312666

**北京分公司**

北京市西城区新街口北大街 3 号新街高和大厦 407 室

电话: 010-82200036 邮箱: beijing@beckhoff.com.cn

**广州分公司**

广州市天河区珠江新城珠江东路16号高德置地G2603 室

电话: 020-38010300/1/2 邮箱: guangzhou@beckhoff.com.cn

**成都分公司**

成都市锦江区东御街18号 百扬大厦2305 房

电话: 028-86202581 邮箱: chengdu@beckhoff.com.cn

|  |  |
| --- | --- |
| QR 代码  描述已自动生成请用微信扫描二维码通过公众号与技术支持交流 | 倍福官方网站：https://www.beckhoff.com.cn在线帮助系统：https://infosys.beckhoff.com/index\_en.htm |
| 倍福虚拟学院：https://tr.beckhoff.com.cn/ |
| 招贤纳士：job@beckhoff.com.cn技术支持：support@beckhoff.com.cn产品维修：service@beckhoff.com.cn方案咨询：sales@beckhoff.com.cn |
|  |