1. 宏\_CAG\_START\_TEXT未定义（一般CagOS中出现）

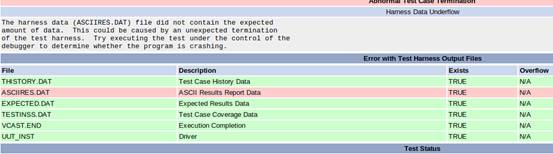
>> \_CAG\_START\_TEXT是编译工程时候make传递，windows底下单独编译文件，可以采用如下方式赋值，

       在模块文件内，

        #define \_CAG\_START\_TEXT address

UINT32 address[10\*1024\*1024];

1. 宏定义#define EPIC\_REG\_READ(offset)    \\*(volatile UINT32 \*)((UINT32)pEpicRegBase + (offset & REG\_ADRS\_MASK))，局部变量地址是固定的，报错（一般CagOS中出现）如下



>> 针对宏定义封装进行地址读写，可以替换宏定义为函数接口，对函数接口进行打桩。

#define EPIC\_REG\_READ epic\_reg\_read

UINT32 epic\_reg\_read(offset)  
{

UINT32 Reg;

Reg = \*(volatile UINT32 \*)((UINT32)pEpicRegBase + (offset & REG\_ADRS\_MASK)

return Reg;

}

1. 对于桩函数返回值是指针，且第一次返回非空第二次返回空的情况

先在Environment->user code->edit中自定义一个全局变量aa,然后写桩代码如下：

static ModbusTCPConfigInfo\_t\* gg;

ModbusTCPConfigInfo\_t const \*g\_pstMBTCPCfg;

aa++;

switch( aa )

{

case 1:

g\_pstMBTCPCfg=&gg;

break;

case 2:

g\_pstMBTCPCfg=NULL;

break;

case 3:

default:

g\_pstMBTCPCfg=NULL;

break;

}

<<Modbus.GetCurMBTCPCfg.return>> = ( g\_pstMBTCPCfg );

或者在user code里定义一个static的局部的aa，初始化为0也可以。

1. 将一个uint8\_t的指针强制转化为结构体指针，然后对结构体成员赋值

首先通过《alloc X》给指针分配多个空间，然后找要赋值变量对应空间，找对应地址方法如问题五。

1. 分配地址时需考虑字节对齐问题

查看结构体变量NetMsgHead\_t地址分配的情况。

     typedef struct NetMsgHeadTag

      {

          uint32\_t uiIPAddr;

          uint16\_t usPort;

          NetworkPortType\_t emNetwork;

        uint8\_t ucReserve[3];

        uint16\_t usLen;

        }NetMsgHead\_t;

其中成员变量NetworkPortType\_t定义如下：

  typedef enum emNetworkPortTypeTypeTag

     { NET1=0x00,

      NET2,

      NUM\_OF\_NETWORK\_PORT,

     INVALID\_NETWORK\_PORT

      }NetworkPortType\_t

所以对于NetMsgHead\_t地址空间分配如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 变量 | | 类型 | 分配空间 |
| uiIPAddr | | uint32\_t | 4字节 |
| usport | | uint16\_t | 2字节 |
| emNetwork | NET1=0X00 | uint8\_t | 与编译器和芯片相关:  VS:4字节  CCS ARM:2字节 |
| NET2 |  |
| NUM\_OF\_NETWORK\_PORT |  |
| INVALID\_NETWORK\_PORT |  |
| ucReserver[3] | | uint8\_t | 3字节 |
| usLen | | uint16\_t | 2字节 |

实际变量分配地址时还需考虑字节对齐的问题，以4个字节为单位，

（1）若变量分配时，4个字节未被其它变量占用，则直接分配地址

（2）若变量分配时，4个字节已被部分占用，则需比较剩余的空间大小与变量的长度，若剩余的空间大小>变量的长度，则直接分配。若剩余的空间大小<变量的长度，则将变量分配到下一个4个字节。

基于此原则，结构体NetMsgHead\_t的地址分配如下：

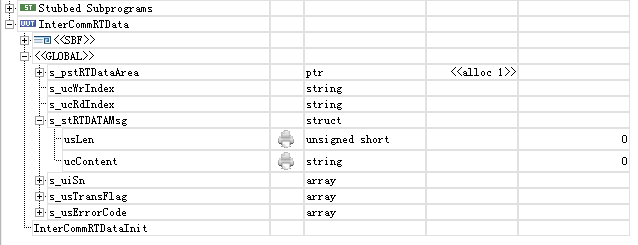
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | uiIPAddr（0-3） |  |
| **4** | **5** | **6** | **7** | usport（4-5） |  |
| **8** | **9** | **10** | **11** | emNetwork（8-11） | |
| **12** | **13** | **14** | **15** | ucReserver[3]（12-14） | |
| **16** | **17** | **18** | **19** | usLen（16-17） | |

注：红色标识的6,7,15为字节补齐的地址，不被变量使用。

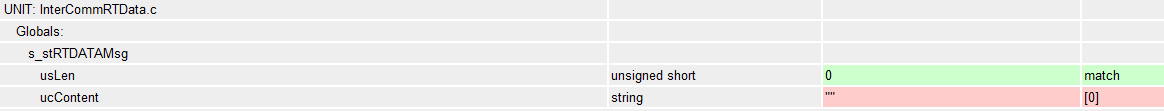
1. string类型读取的是其ACSII码，最好先通过转化为String Display Mode转化为Array

或Pointer类型，否则出错

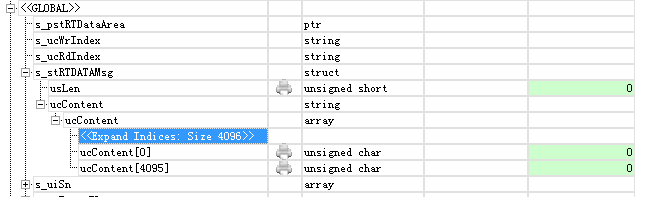
例如：memset((puint8\_t)&s\_stRTDATAMsg, 0, sizeof(SysRTDataMsg\_t));一个简单初始化语句，VCAST里若设置如下:



输出结果fail:



若将string转化为Array类型输出结果正确:



1. 对于uesr类型结构体，里面的成员函数是指针类型，在uercode里设置函数不认，可以这样设置

比如 \_\_shell\_cmds是一个数组指针结构体，

static const struct shell\_cmd \* \_\_shell\_cmds[MAX\_SHELL\_CMD];

成员如下：struct shell\_cmd {

const char \* name;

const char \* help;

const char \* usage;

int (\*handler)(int argc, char \* argv[]);

};

被测代码如下for (i = 0; i < MAX\_SHELL\_CMD; i++)

{

const struct shell\_cmd \*c = \_\_shell\_cmds[i];

if (c && !strcmp(argv[0], c->name))

{

ret = c->handler(argc, argv);

if (ret)

要设置c->handler在\_\_shell\_cmds的uercode里写桩代码不认，可以自己定义一个结构体t1和其成员函数test如下：

struct shell\_cmd t1;

int test(int a, char \*b)

{

return 0;

}

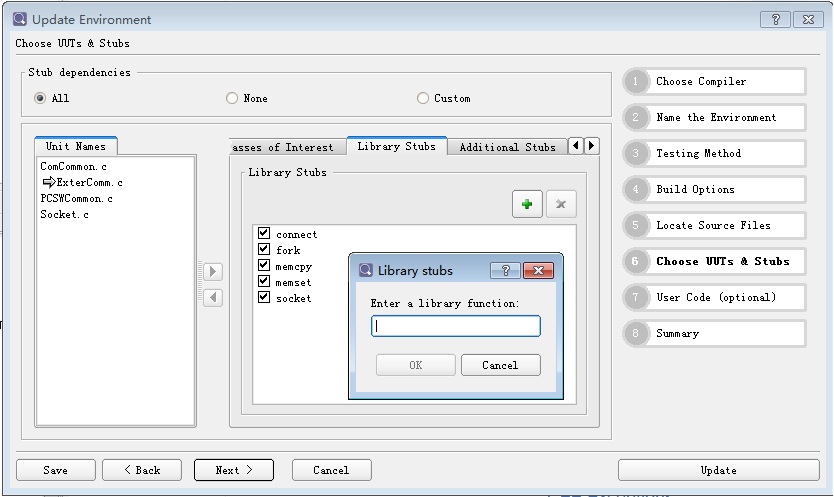
然后在VCAST的SBF里设置test返回值，\_\_shell\_cmds的uercode指向t1<<shell.<<GLOBAL>>.\_\_shell\_cmds[0]>> = ( &t1 );，t1里的handler指向test<<shell.<<GLOBAL>>.t1>>.handler = ( test );。

注：若\_\_shell\_cmds不是指针结构体，直接自定义一个函数，让handler指向即可。

1. VectorCAST构建环境时会遇到一些问题，此时需要进行文件替换和测试软件设置

1、对于未定义系统函数，需替换王东修改的sys文件，见附件；

2、对于系统函数无法打桩问题，在VectorCAST构建测试环境时，Choose UUTs & Stubs步骤时，选择需测试的.c文件后，在Library Stubs界面添加需打桩的系统函数，如截图所示



VCAST面板上标签代表意义：

|  |  |  |
| --- | --- | --- |
| 图标 | 代表意义 | |
| Test Settings | 定位问题发生在那一阶段（预处理、编译还是解析） | |
| Testing Method  （2、3不需要源码，但无法插桩，看不到覆盖率） | Traditional Unit Testing传统单元测试 | |
| Object File Testing针对目标文件（Obj文件）测试 | |
| Library Interface Testing针对库接口测试 | |
| Test-Driven Development测试驱动开发 | |
|  | 添加当前目录以及所包含子目录，添加多个路径 | |
|  | 以函数为单位进行打桩的动作（stubs by fuction） | |
| UUT | 所有被测单元以外的依赖均被打桩（Units Under Test） | |
| Libaray stubs | 对malloc、printf等系统函数、库函数打桩 | |
| Additional Stubs | 添加额外的桩（不在所测环境的桩、所添加路径里不包含的桩） | |
| UUT\_INST\_EXE | 有插桩，有覆盖率信息 | 可执行文件，在Message中的linking可生成 |
| UUT\_INTE\_EXE | 无插桩，无覆盖率信息 |
| Compound Tests | 集成测试用例位置，集成测试时先建立用例，然后把相应函数单元用例拖进去 | |
| Initialization Tests | 调试用例，建立最原始测例，定位函数存在问题 | |
|  | S表示搜索search，一般都选择S | |
|  | I搜索头文件 include | |
|  | T用户自己定义类型，只有头文件type handle | |
|  | 勾选whitebox可看到调用到的静态、保护、私有对象 | |

有些编译器没有嵌套在VCAST工具里，需要写启动脚本启动VCAST，以CCS编译器举例，在目录Workspaces/CCS4/vcast.bat下编写启动脚本如下：

set VCAST\_CCS\_INSTALL\_DIR=C:\progra~1\TexasI~1\ccsv4（安装目录路径）

set path=% VCAST\_CCS\_INSTALL\_DIR%\scripting\examples\loadti;%path%

set path=% VCAST\_CCS\_INSTALL\_DIR%\tools\compiler\tms470\bin;%windir%\system32;%path%（2、3行告诉工具要找的程序、编译器、库、头文件位置，%path%必须有否则许多程序读不到，%windir%\system32;可省略）

set DEBUGSERVER\_ROOT= C:\progra~1\TexasI~1\ccsv4\Debugserver（指定Debugserver路径）

start %VECTORCAST\_DIR%\vcastqt –lc（启动工具且设为C语言）

cmd

查看短路径命令：dir /x