自定义封装函数的问题

主要问题：

- 函数参数定义问题
- 函数实现问题

以下举例说明

参数顺序不一致

对于memcpy:

- SIP 模块使用第3个参数作为实际使用的复制大小：

```
WORD32 P_SIPCA_memcpy(OUT VOID *pvDst,IN const VOID *pvSrc,
     IN WORD32 dwNeedCopyLen
    ,IN WORD32 dwDstMaxSize);
```

- MGCP/H248 模块使用第4个参数作为实际使用的复制大小:

```
LPSTR PMGCP_MEMCPY(LPSTR pbDstBuffer,LPSTR pbSrcBuffer,
    WORD32 dwDestLen, WORD32 dwSrcLen
    );
LPSTR PH248_MEMCPY(LPSTR pbDstBuffer,LPSTR pbSrcBuffer,
    WORD32 dwDestLen, WORD32 dwSrcLen
    );
```

不一致的定义容易引发混淆

参数定义与标准函数不一致

对于memcpy:

- 使用WORD32作为返回值,而不是缺省的目的指针地址:

```
WORD32   P_SIPCA_memcpy(OUT VOID *pvDst,IN const
    VOID *pvSrc,IN WORD32 dwNeedCopyLen,IN WORD32
    dwDstMaxSize);
```

- 标准函数定义:

```
void*   memcpy (void* dstpp,const void* srcpp,size_t
     len);
```

如果使用返回值,容易出现错误

参数定义与标准函数不一致

对于memcpy:

- 使用LPSTR作为参数类型,而不是缺省的 void*:

```
LPSTR   PMGCP_MEMCPY( LPSTR   pbDstBuffer, LPSTR
    pbSrcBuffer,WORD32 dwDestLen,WORD32 dwSrcLen});
```

- 标准函数定义:

```
void*   memcpy ( void*   dstpp,const  void*
    srcpp,size_t len);
```

这样会导致 *PCLINT* 对指针大小的判断失效

参数定义与标准函数不一致

对于strcpy:

- 封装函数:

```
VOID     PH248_STRCPY(LPSTR strDes, LPSTR strS);
WORD16   P_SIPCA_strcpy(OUT BYTE *pbDst, IN const BYTE *pbSrc, IN  WORD16
      wMaxDstLen);
LPSTR    PMGCP_STRCPY(LPSTR pbDstBuffer, LPSTR pbSrcBuffer,  WORD32
      dwDestLen);
WORD16   P_SIPCA_strncpy(OUT BYTE *pbDst, IN const BYTE *pbSrc, IN  WORD16
      wNeedCopyLen, IN  WORD16   wMaxDstLen);
```

- 标准函数定义:

```
char *strncpy (char *dest, const char*src,size_t n);
char *strcpy (char *dest, const char*src);
```

应该返回什么？到底用 *WORD16* 还是用 *WORD32*。

封装不彻底

对于strcpy:

- 封装函数:

```
VOID  PH248_STRCPY(LPSTR strDes, LPSTR strS)
{
    if(strDes != NULL)
    {
        strcpy(strDes,strS);
    }
    return;
}
```

仅仅判断第一个指针？另：此函数的注释是 *memcpy* 的注释

内部实现欠妥

对于memcpy:

- 封装函数:

```
void* PH248_MEMCPY(void* pbDstBuffer, void* pbSrcBuffer,
    WORD32 dwDestLen, WORD32 dwSrcLen)
{
    ......
    if(dwDestLen < dwSrcLen)
    {
        /** chengliang added for agcf */
        memcpy(pbDstBuffer, pbSrcBuffer, dwDestLen);
        return NULL;
    }
    memcpy(pbDstBuffer, pbSrcBuffer, dwSrcLen);
    return pbDstBuffer;
}
```

返回 *null*, 表示函数执行失败, 此时仍然执行复制了。*PH248_MEMSET*一样。

内部实现欠妥

对于strncpy:

- **封装函数:**

```
LPSTR PMGCP_STRCPY(LPSTR pbDstBuffer, LPSTR pbSrcBuffer, WORD32 dwDestLen)
{
    WORD32 dwSrcLen = 0;
    if(NULL == pbDstBuffer || NULL == pbSrcBuffer||0 == dwDestLen)
    {
        return NULL;
    }
     dwSrcLen = strlen(pbSrcBuffer);
    if(dwDestLen <= dwSrcLen)
    {
        strncpy(pbDstBuffer,pbSrcBuffer,dwDestLen);
        *(pbDstBuffer+dwDestLen -1) = '\0';
    }
    else
    {
         strncpy(pbDstBuffer, pbSrcBuffer,dwSrcLen);
         *(pbDstBuffer+dwSrcLen ) = '\0';
    }
    return pbDstBuffer;
}
```

且不说 *strlen* 额外进行了一次遍历, 如果 *pbSrcBuffer* 没有结束符, *strlen* 不安全

无意义封装

对于strlen:

- 封装函数:

```
WORD16 P_SIPCA_strlen(IN const BYTE *pbString)
{
    WORD16 wRet;
    wRet = strlen((char*)pbString);
    return wRet;
}
```

仅仅是多一次调用

后续建议：

- 参数和返回值定义请按照 C 标准库定义, 可以增加参数, 但是不要改变原有的参数定义
- 函数实现请参考标准库的源代码, 或者是 gnu libc 的源代码。这些基本函数会被多次调用, 有必要仔细推敲
- 不需要每个模块都定义一套实现, 应该部门内部共享共用
- 函数名称改变需要修改 pclint 检查配置, 否则会使 pclint 检查失效