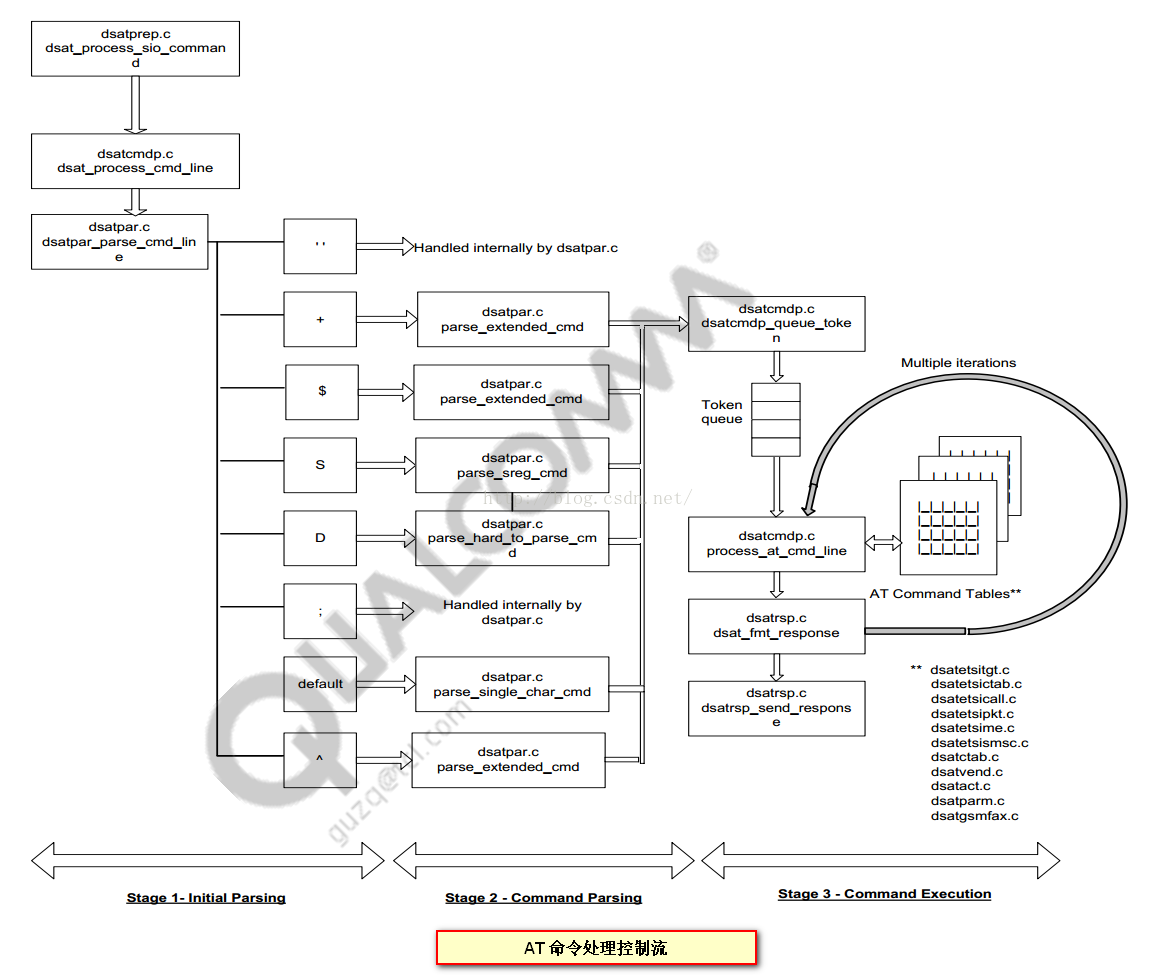
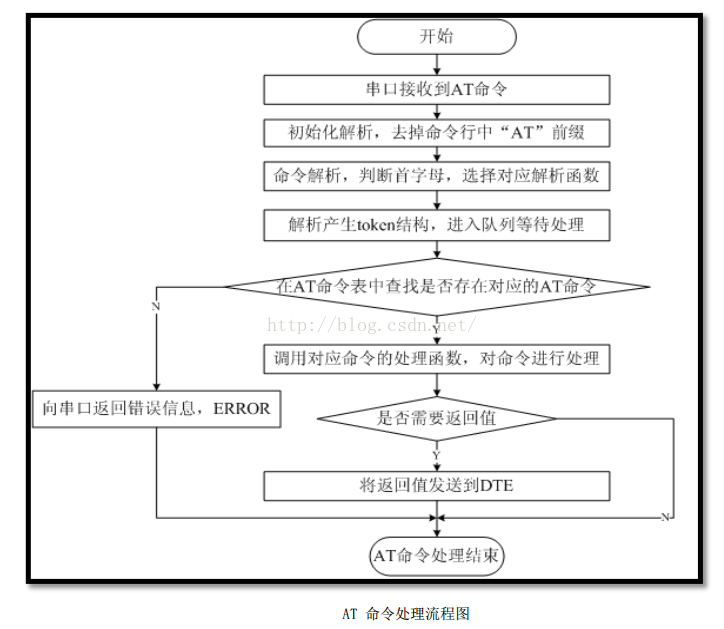
## ATCoP处理流程:



## 关键代码



**/amss\_2017\_la3.0/MPSS.AT.3.1/modem\_proc/datamodem/interface/atcop/src/**

1、dsatprep.c

dsatprep\_process\_sio\_command：处理不同类型的命令（AT command line、abort command、SMS message）

process\_at\_cmd\_mode：通过状态机转换（如下4种状态），去掉命令行中的"AT"前缀  
       DSAT\_CMD\_PREP\_STATE\_HUNT,     /\*  Initial:  looking for a or A                      \*/  
       DSAT\_CMD\_PREP\_STATE\_FOUND\_A,  /\*  Found A, looking for t, T or /                    \*/  
      DSAT\_CMD\_PREP\_STATE\_FOUND\_AT, /\*  Found AT: filling buffer and processing backspace \*/  
      DSAT\_CMD\_PREP\_STATE\_ERROR     /\*  Error!  loop until end of line                    \*/

2、dsatcmdp.c

dsatcmdp\_process\_cmd\_line：调用**dsatpar\_parse\_cmd\_lin**e对命令进行解析，调用**process\_at\_cmd\_line**对命令进行处理

1. dsatpar\_parse\_cmd\_line（dsatpar.c）  
   完成对命令的解析，检查每个AT命令的首字符然后根据AT命令的类型调用相应的解析函数。每次调用解析函数，都会产生一个tokens\_struct\_type类型的token，将解析的信息填充到token，并调用函数dsatcmdp\_queue\_token将token加入到队列token\_q
2. process\_at\_cmd\_line

处理队列token\_q，从队列token\_q中取出并删除一个命令token，cmdp\_validate\_cmd\_forwarding在命令表中搜索该命令。

如果在表中(**at\_cmd\_table**\**at\_cmd\_table\_ex**)找到改命令.调用表中对应处理函数执行该命令（table\_ptr\_ex->proc\_func）。命令执行后，如果有返回数据时，返回的数据在中格式化dsat\_fmt\_response

**注**：对于异步AT命令处理流程与正常AT命令略有不同，在命令预处理、命令解析过程都是一样的，在命令处理过程中（process\_at\_cmd\_line），如果命令处理函数返回DSAT\_OK，说明命令处理完成调用dsat\_fmt\_response函数格式化响应数据并发送，正常的命令处理流程；如果命令处理函数返回DSAT\_ASYNC\_CMD说明当 前 命 令 是 异 步 命 令 ， 此 时 函 数process\_at\_cmd\_line 设置 变 量dsatcmdp\_processing\_async\_cmd= TRUE，表示当前正在处理异步命令，然后返回，不再进行后续处理，直到该命令处理完成，函数返回DSAT\_OK（未必一定是DSAT\_OK，当返回不是DSAT\_ASYNC\_CMD和DSAT\_ASYNC\_EVENT时，说明异步命令/事件处理完成）。当DS收到异步事件经任务分发器，再次调用dsat\_process\_async\_cmd函数，在该函数中通过查找异步事件表async\_event\_table，调用相应的事件处理函数继续处理，如果事件处理函数返回值不是DSAT\_ASYNC\_CMD或DSAT\_ASYNC\_EVENT，说明异步事件处理完成，调用process\_at\_cmd\_line继续处理命令行的命令。

表结构：

**如下举例（父表到子表）**

**at\_cmd\_table\_ex -> vendor\_table\_entries\_ex -> dsat\_vendor\_table\_ex**

**dsat\_vendor\_table\_ex 表中含AT命令和处理函数的映射**

**如下举例（父表到子表）**

**at\_cmd\_table\_ex -> vendor\_table\_entries -> dsat\_vendor\_table**

**dsat\_vendor\_table  表中定义AT命令，确认了它的命令表项，name、属性、参数情况、处理函数指针。。。。**

const dsati\_at\_cmd\_table\_ex\_type at\_cmd\_table\_ex =  
  { basic\_table\_entries\_ex,               //基本命令表  
    sreg\_table\_entries\_ex,        //寄存器命令表  
    extended\_table\_entries\_ex,        //扩展命令表  
    vendor\_table\_entries\_ex        //厂商命令表  
  };

还不太清楚下面表的分类

LOCAL dsati\_at\_cmd\_table\_entry\_type vendor\_table\_entries[ ] =  
{  
 { dsat\_vendor\_table,  &dsat\_vendor\_table\_size },  
#ifdef FEATURE\_DATA\_IS707  
 { dsat707\_vend\_para\_table, &dsat707\_vend\_para\_table\_size },  
 { dsat707\_vend\_action\_table, &dsat707\_vend\_action\_table\_size },  
 { dsat707\_hdr\_table,  &dsat707\_hdr\_table\_size },  
#ifdef FEATURE\_DS\_MOBILE\_IP  
 { dsat707\_mip\_table,  &dsat707\_mip\_table\_size },  
#endif /\* FEATURE\_DS\_MOBILE\_IP \*/  
#ifdef FEATURE\_CDMA\_SMS  
 { dsat707\_sms\_table, &dsat707\_sms\_table\_size },  
#endif /\* FEATURE\_CDMA\_SMS \*/  
#endif /\* FEATURE\_DATA\_IS707 \*/  
    { NULL, 0 }  
};

## 如何扩展一个BP侧AT Command

AT命令分有以下几种类型

·基本 AT 命令（basic\_table）   
·寄存器 AT 命令（sreg\_table）   
·扩展 AT 命令（extended\_table）   
·厂商 AT 命令（vendor\_table）

**1、在dsati.h下的dsatetsi\_ext\_action\_index\_enum\_type枚举数组中添加一个指针变量如下：**

DSATETSI\_EXT\_ACT\_CLAY\_ETSI\_IDX = 14084

1. **建立AT命令和处理函数的映射，在dsatetsictab.c下的dsatetsi\_ext\_action\_table\_ex []数据下添加映射：**

{DSATETSI\_EXT\_ACT\_CLAY\_ETSI\_IDX, dsatetsime\_exec\_clay\_cmd }

**3、在dsatetsictab\_ex.c下的dsatetsi\_ext\_action\_table []数组中添加（name、属性、参数情况、处理函数指针等……）：**

{ "+CLAY", READ\_ONLY | COMMON\_CMD,

SPECIAL\_NONE, 0,DSATETSI\_EXT\_ACT\_CLAY\_ETSI\_IDX

}

**4、声明处理函数**

在定义之前，我们先要声明一下，在dsatetsime.h下添加定义：

dsat\_result\_enum\_type dsatetsime\_exec\_clay\_cmd(

dsat\_mode\_enum\_typemode, /\*AT command mode: \*/

constdsati\_cmd\_type \*parse\_table, /\*Ptr to cmd in parse table \*/

consttokens\_struct\_type \*tok\_ptr, /\*Command tokens from parser \*/

dsm\_item\_type\*res\_buff\_ptr /\* Place to put response \*/

);

1. **定义处理函数dsatetsime.c：**

dsat\_result\_enum\_type dsatetsime\_exec\_clay\_cmd(

dsat\_mode\_enum\_typemode, /\*AT command mode: \*/

constdsati\_cmd\_type \*parse\_table, /\*Ptr to cmd in parse table \*/

consttokens\_struct\_type \*tok\_ptr, /\*Command tokens from parser \*/

dsm\_item\_type\*res\_buff\_ptr /\* Place to put response \*/

){

dsat\_result\_enum\_type result= DSAT\_OK;

if(tok\_ptr->op == NA){

res\_buff\_ptr->used =(word) snprintf ((char\*)res\_buff\_ptr->data\_ptr,

res\_buff\_ptr->size,

"%s: %s,%s",

"+CLAY",

"hello",

"world");

}

else if(tok\_ptr->op ==(NA|EQ|QU)){} //针对其他的语法格式进行处理

else if(tok\_ptr->op ==(NA|QU)){} //针对其他的语法格式进行处理

else if(tok\_ptr->op ==(NA|EQ|AR)){} //针对其他的语法格式进行处理

else{ result= DSAT\_ERROR;} //针对错误的语法格式进行处理

return result;

}