**注册端点的函数afRegister( endPointDesc\_t \*epDesc )**

afRegister( endPointDesc\_t \*epDesc )这个函数用来注册一个新的端点到task，这样当空中有这个端点的数据到来会直接发送到对应的task。

传入参数：

typedef struct  
{  
  byte endPoint; //端点号  
  byte \*task\_id;   //端点对应的任务号  
  SimpleDescriptionFormat\_t \*simpleDesc; //简单描述符  
  afNetworkLatencyReq\_t latencyReq; //必须用noLatencyReqs来填充  
} endPointDesc\_t;

afStatus\_t afRegister( endPointDesc\_t \*epDesc )  
{  
  epList\_t \*ep;  
    
  // Look for duplicate endpoint  
  if ( afFindEndPointDescList( epDesc->endPoint ) )   
    return ( afStatus\_INVALID\_PARAMETER );  
  //在端点的链表中搜索新的端点描述符，看看能不能搜到。如果能搜到，则返回错误。

  //如果不能搜到，则说明这个端点在此之前没有被注册，则调用afRegisterExtended( epDesc, NULL )函数注册这个端点。  
  ep = afRegisterExtended( epDesc, NULL );  
  
  
  return ((ep == NULL) ? afStatus\_MEM\_FAIL : afStatus\_SUCCESS);  
}

//注册端点描述符

epList\_t \*afRegisterExtended( endPointDesc\_t \*epDesc, pDescCB descFn )  
{  
  epList\_t \*ep;  
  epList\_t \*epSearch;  
  
  
  //注册意味着在端点链表中增加一个新的元素，先通过osal\_mem\_alloc( sizeof ( epList\_t ) )申请需要的内存单元  
  ep = osal\_mem\_alloc( sizeof ( epList\_t ) );  
  if ( ep ) //如果申请成功  
  {  
    // Fill in the new list entry  
    ep->epDesc = epDesc;  
  
  
    // Default to allow Match Descriptor.  
    ep->flags = eEP\_AllowMatch;  
    ep->pfnDescCB = descFn;  
    ep->nextDesc = NULL;  
  
  
    // Does a list exist?  
    if ( epList == NULL )  //epList为全局epList\_t 指针变量，当链表为空时它指向NULL，如果这是创建的第一个链表元素，则 epList = ep  
      epList = ep;  // Make this the first entry  
    else //如果这不是第一个链表元素，则首先找到链表的首，在首端加入这个新的元素。  
    {  
      // Look for the end of the list  
      epSearch = epList;  
      while( epSearch->nextDesc != NULL )  
        epSearch = epSearch->nextDesc;  
  
  
      // Add new entry to end of list  
      epSearch->nextDesc = ep;  
    }  
  }  
  
  
  return ep;  
}

typedef struct  
{  
  endPointDesc\_t \*epDesc;  
  eEP\_Flags flags;  
  pDescCB  pfnDescCB;     // Don't use if this function pointer is NULL.  
  void \*nextDesc;  
} epList\_t;

在实际工作中，afIncomingData( aps\_FrameFormat\_t \*aff, zAddrType\_t \*SrcAddress, uint16 SrcPanId,NLDE\_Signal\_t \*sig, byte SecurityUse, uint32 timestamp )这个函数从APS层收到数据，在这个函数中判断收到数据对应的端点号在端点链表中能不能找到，如果不能找到则丢掉这包数据，如果能找到则把数据发送的端点对应的task.

通过  afBuildMSGIncoming( aff, epDesc, SrcAddress, SrcPanId, sig, SecurityUse, timestamp )这个函数建立OSAL的数据包到对应的任务，并且置位对应任务的SYS\_EVENT\_MSG。