void MailboxApp_Init(void)

```
void MailboxApp_Init(void)
{
    Mailbox_Config  cfg;
    int32_t         errCode;
    Semaphore_Params    semParams;

   /* Create a binary semaphore which is used to handle mailbox interrupt. */
   Semaphore_Params_init(&semParams);
   semParams.mode             = Semaphore_Mode_BINARY;
   st_gMrrMSSMCB.mboxSemHandle = Semaphore_create(0, &semParams, NULL);

    /* Create a binary semaphore which is used to handle object interrupt. */
    Semaphore_Params_init(&semParams);
    semParams.mode             = Semaphore_Mode_BINARY;
    Object_SemHandle = Semaphore_create(0, &semParams, NULL);

    /* Initialize the Mailbox */
    Mailbox_init(MAILBOX_TYPE_MSS);

    /*/ receive mailbox init*/
    if(Mailbox_Config_init(&cfg) < 0)
    {
        Mss_Errors.MailBox_Error= 1u;
    }
    /****** ch 1 ******************************************/
    cfg.chType       = MAILBOX_CHTYPE_MULTI;
    cfg.chId         = MAILBOX_CH_ID_0;
    cfg.readMode     = MAILBOX_MODE_CALLBACK;
    cfg.readCallback = &MailboxAppCallback;
    cfg.writeMode    = MAILBOX_MODE_POLLING;

    MailboxAppHandle[MAILBOX_ID_RECEIVE] = Mailbox_open(MAILBOX_TYPE_DSS, &cfg,
&errCode);
    if (MailboxAppHandle[MAILBOX_ID_RECEIVE] == NULL)
    {
        Mss_Errors.MailBox_Error = 2u;
    }
    if (errCode != 0)
    {
        Mss_Errors.MailBox_Error = 3u;
    }

    /*/ send mailbox init*/
    if(Mailbox_Config_init(&cfg) < 0)
    {
        Mss_Errors.MailBox_Error = 4u;
    }
    /****** ch 1 ******************************************/
```

```
    cfg.chType       = MAILBOX_CHTYPE_MULTI;
    cfg.chId         = MAILBOX_CH_ID_1;
    cfg.readMode     = MAILBOX_MODE_CALLBACK;
    cfg.readCallback = &MailboxAppCallback;
    cfg.writeMode    = MAILBOX_MODE_POLLING;

    MailboxAppHandle[MAILBOX_ID_SEND] = Mailbox_open(MAILBOX_TYPE_DSS, &cfg, &errCode);
    if (MailboxAppHandle[MAILBOX_ID_SEND] == NULL)
    {
        Mss_Errors.MailBox_Error = 5u;
    }
    if (errCode != 0)
    {
        Mss_Errors.MailBox_Error = 6u;
    }
}
```

DSS和MSS的交互靠的是邮箱（ Mailbox）。

信号量 : 这是DSP sys_bios操作系统里面的东西，作用就是用于处理邮箱中断，本质上是一个二进制信号量。

一直查看定义， semParams.mode =0x1；

后面这个参数创建了一个信号量。

邮箱初始化

然后继续获取邮箱初始化的参数，放在Cfg变量中，

如果Mailbox_Config_init(&cfg) < 0

联合体Mss_Errors.MailBox_Error= 1u;// receive mailbox init模式

然后对他进行修改，包含单邮箱通道和多邮箱通道类型选择、通道ID号（一共有7个）、读和写模式、读回调。

> 读回调函数在Mailbox_Config_init上面
>
> ```
> static void MailboxAppCallback(Mbox_Handle handle, Mailbox_Type remoteEndpoint)
> {
>     if(handle==MailboxAppHandle[MAILBOX_ID_RECEIVE])
>     {
>         Semaphore_post (st_gMrrMSSMCB.mboxSemHandle);
>     }
> }
> ```

这里面的Semaphore_post (st_gMrrMssMCB.mboxSemHandle)函数，是用来判断是否已从每个endpoint端点接收到消息，然后唤醒mmWave线程进行处理。

继续回到之前，接着是打开虚拟邮箱通道和检验，注意邮箱类型是MAILBOX_TYPE_DSS。

```
extern Mbox_Handle Mailbox_open(Mailbox_Type remoteEndpoint, const Mailbox_Config *cfg,
int32_t* errCode);
```

这个函数告诉我们，他的参数是远程端，也就是remoteEndpoint，也就是DSS端。

```
  /*/ send mailbox init*/
    if(Mailbox_Config_init(&cfg) < 0)
    {
        Mss_Errors.MailBox_Error = 4u;
    }
```