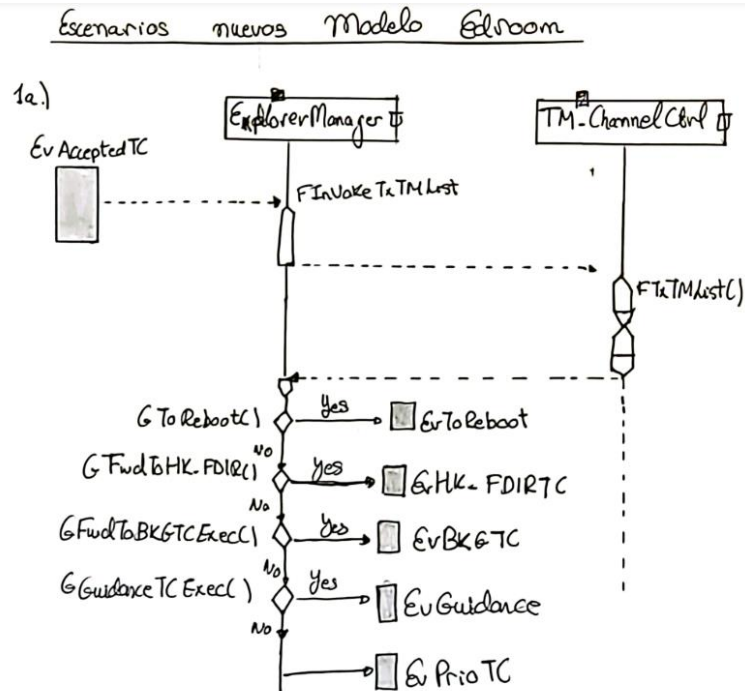
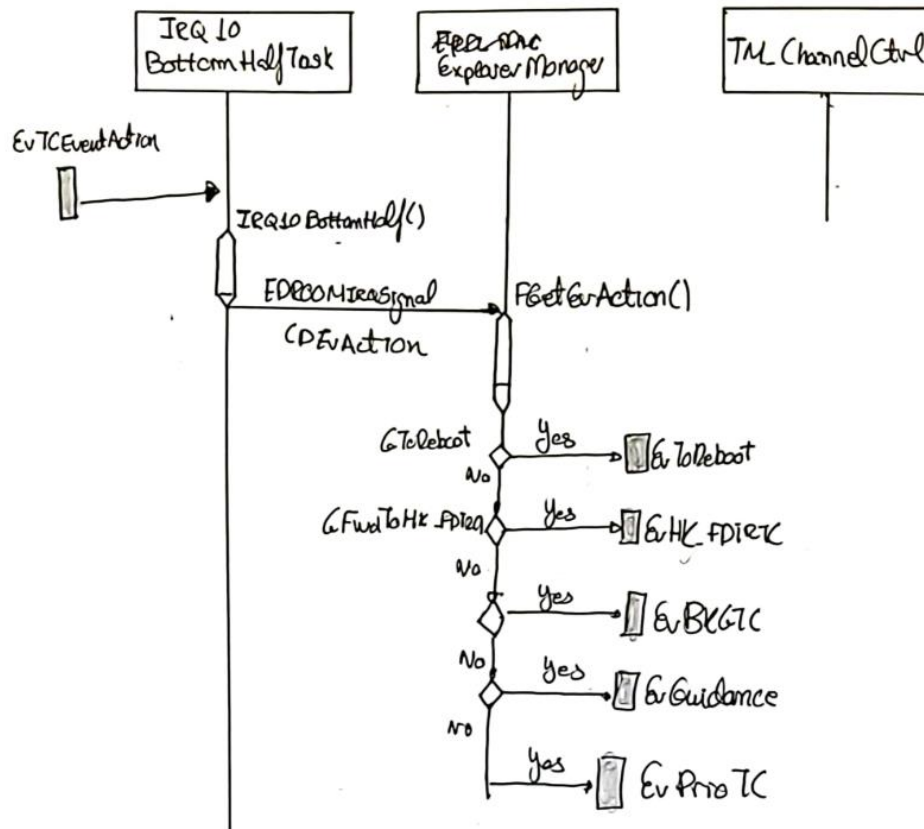


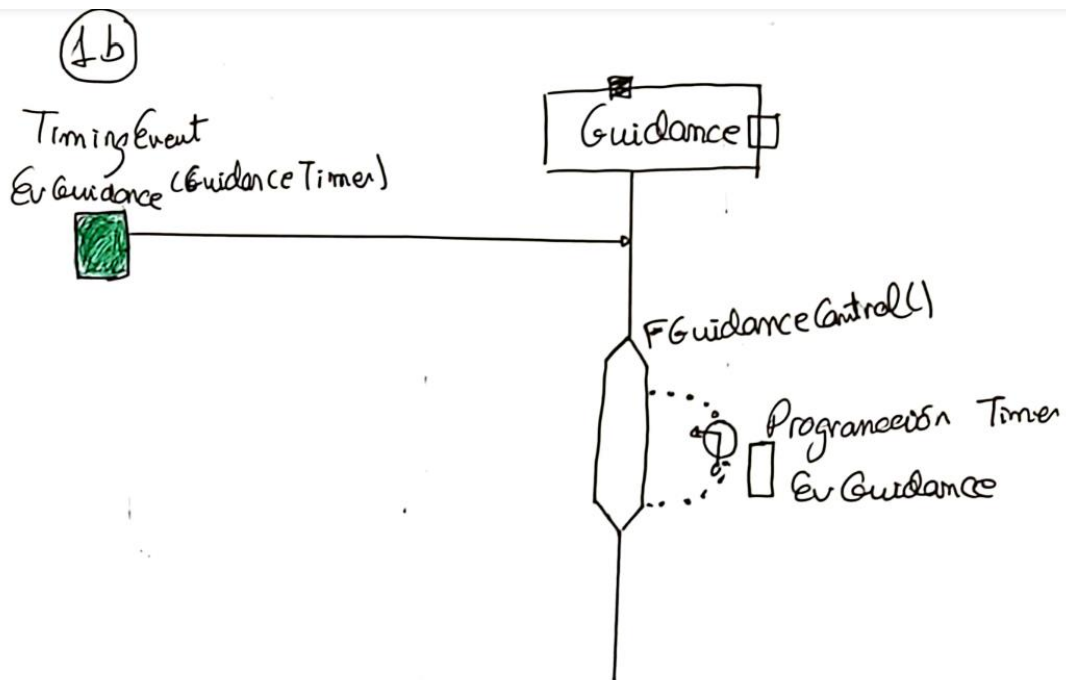
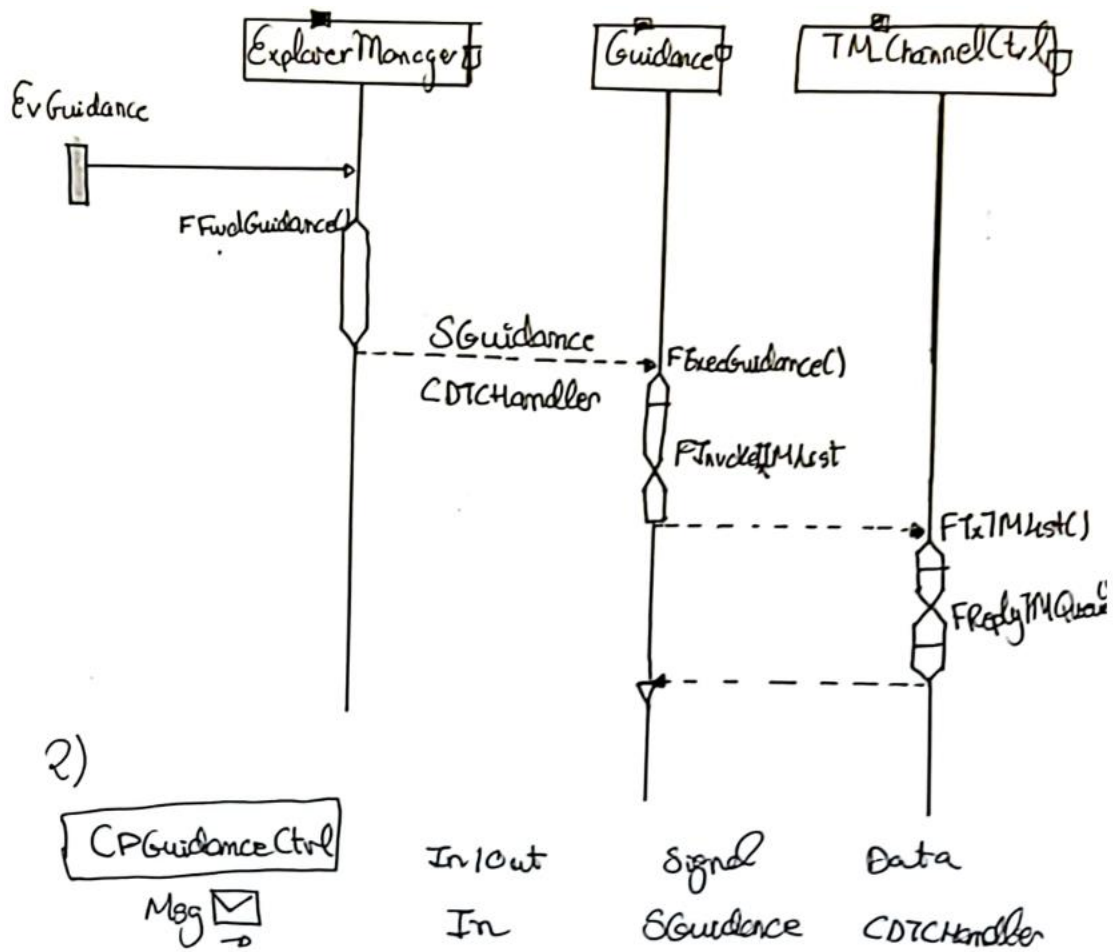
ENTREGABLES 1,2,3,4 – OBDH- DIEGO LÓPEZ SUERO

1.)




(Faltarían las guardas asociadas a los eventos de Background y Guidance, respectivamente, son las mismas que en el diagrama anterior)





2.) Definición de la clase Protocolo a añadir al Modelo EDROOM, aportando toda la información de cada mensaje.

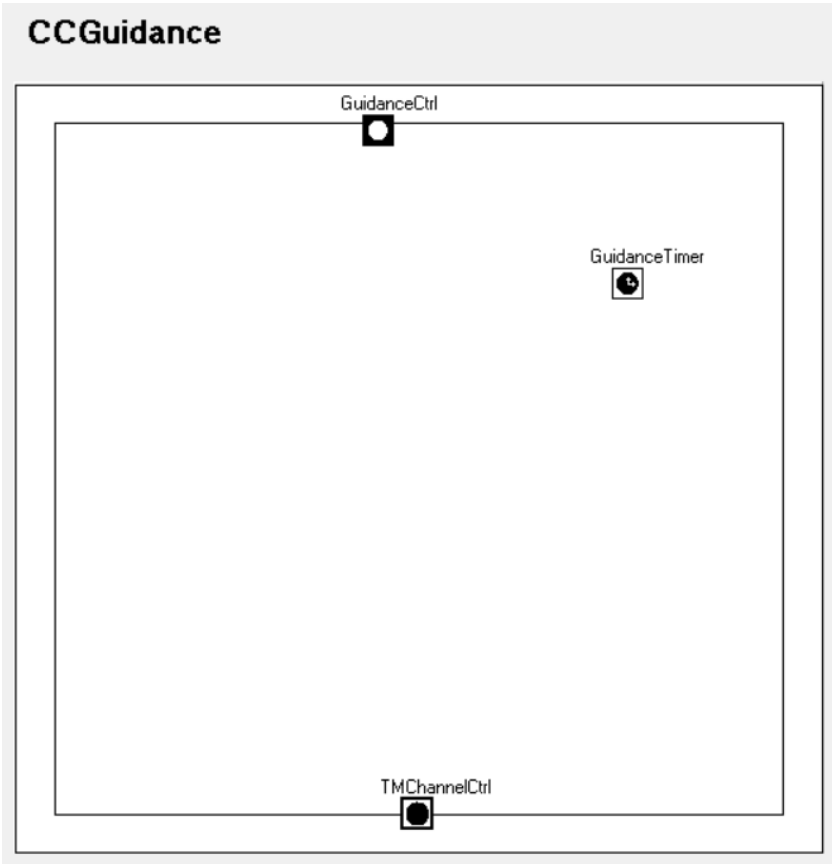
CPGuidanceCtrl

| | In/Out | Signal | Data |
|---|--------|-----------|-------------|
| Msg  | IN | SGuidance | CDTCHandler |

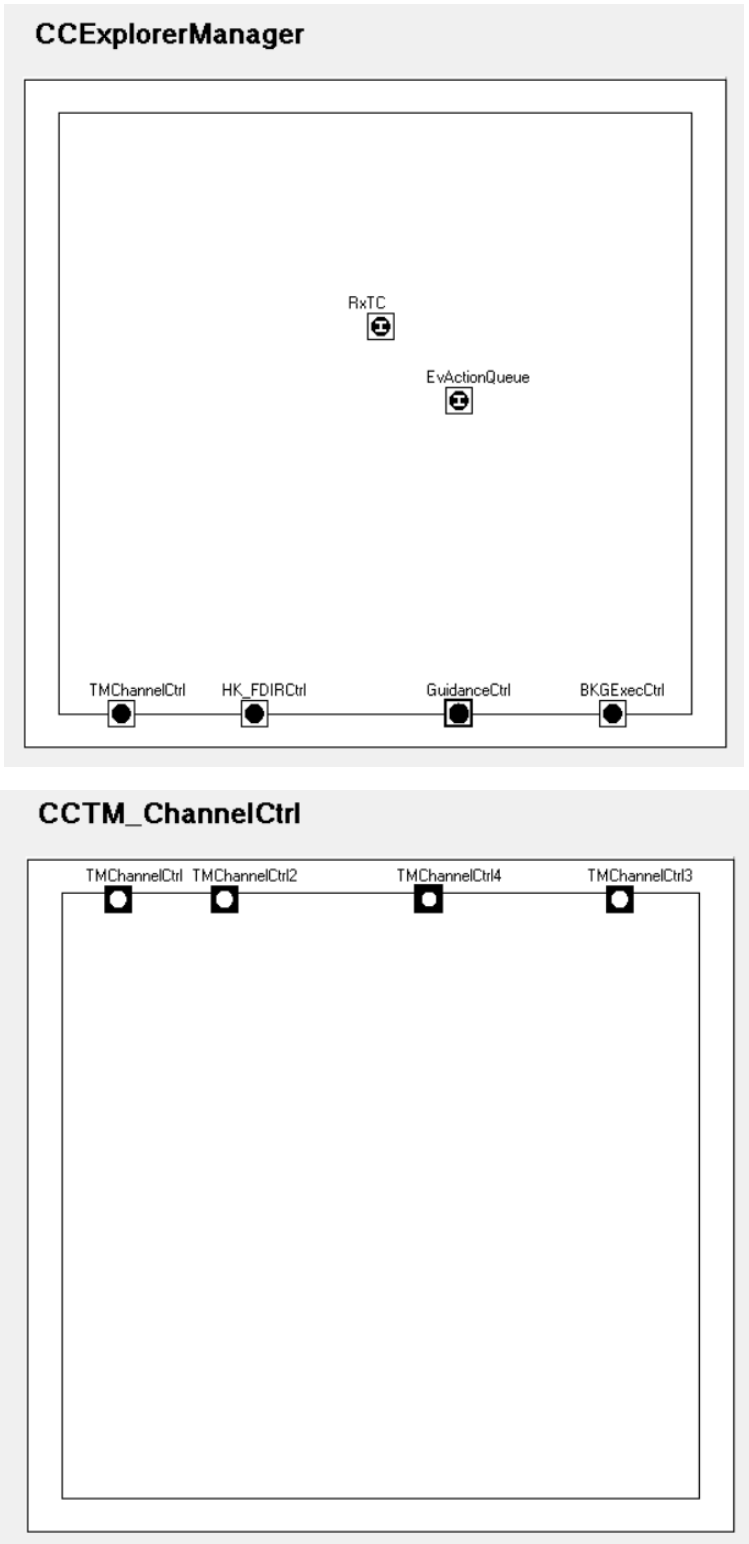
3.) Diseño de la interfaz de la clase componente CCGuidance empleando la misma notación gráfica que se ha proporcionado durante las prácticas, y que debe definir:

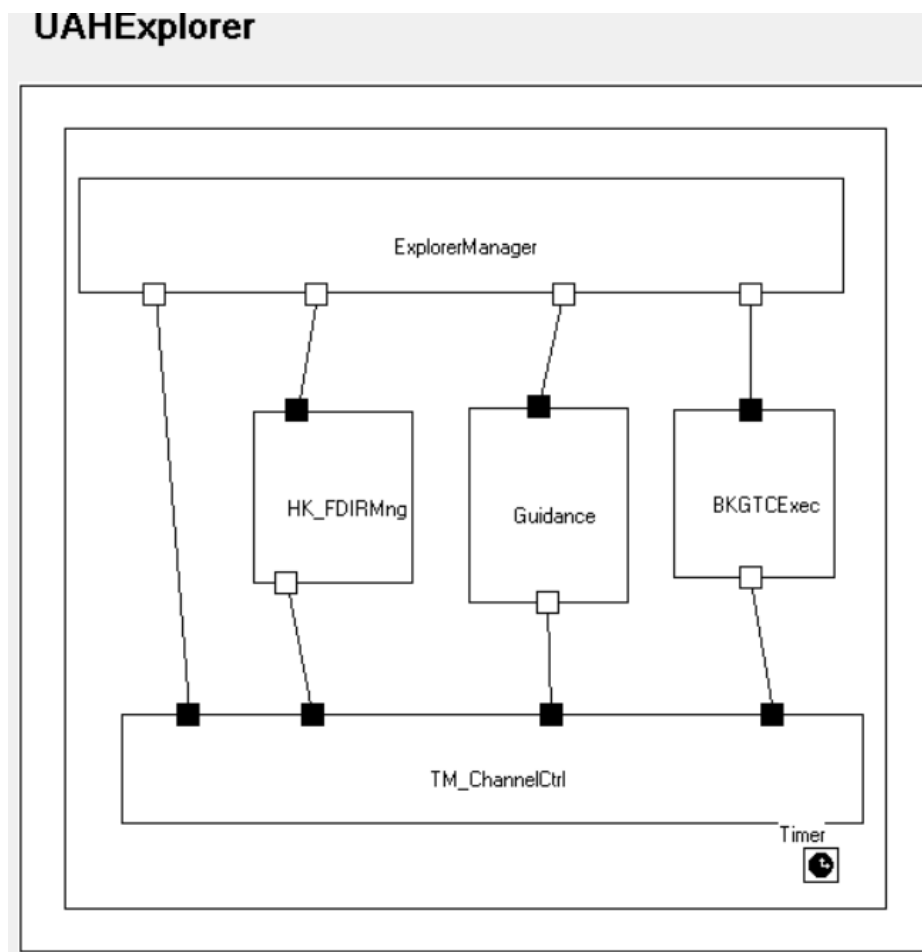
- a. Los puertos de la clase componente, indicando con la notación gráfica correspondiente si el puerto permite solicitar servicios de temporización, o es un puerto de comunicaciones o es un puerto asociado a una Interrupción.
- b. Para los puertos de comunicaciones, indicad la clase protocolo de cada puerto y el tipo de asociación (conjugada o nominal)

En el nuevo componente CCGuidance, tendremos un puerto de timing *GuidanceTimer* para la realización del control de la nave, un puerto (conjugado) *TMChannelCtrl* asociado a *CPTMChannelCtrl* y un puerto nominal asociado al protocolo *CPGuidanceCtrl*.



Por otra parte, se adjuntan las diferentes capturas asociadas a la comunicación del nuevo componente con los componentes CCTM_ChannelCtrl y CCExplorerManager, así como las conexiones globales del sistema.

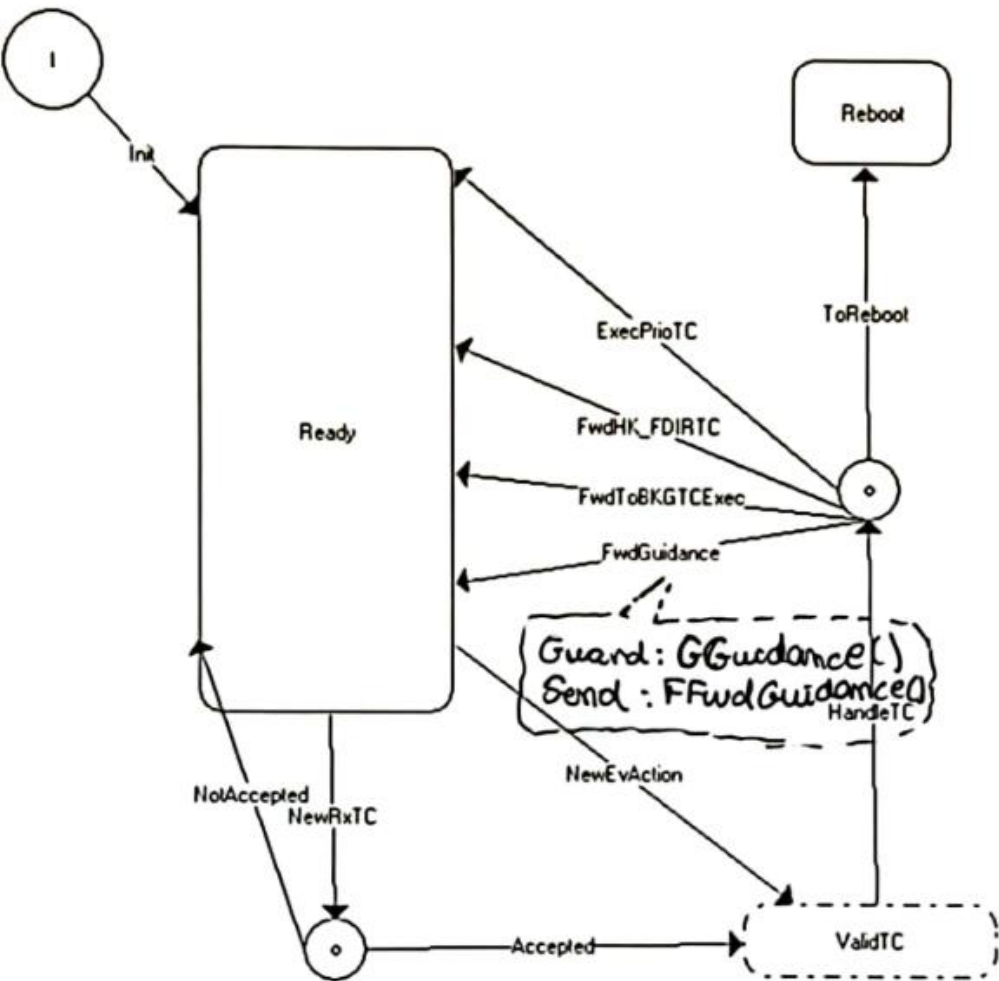




4. Diseño del comportamiento de la clase componente CCGuidance empleando la misma notación gráfica que se ha proporcionado durante las prácticas, y que debe definir:

- a. La máquina de estados de la clase componente.
- b. La declaración de las Variables de la clase componente.
- c. Definición del trigger de cada transición
- d. Actions a ejecutar, indicando:
 - Si está asociada a una transición, o a la entrada o salida de un estado.
 - El tipo de Action (MsgDataHandler, Action, InformIn, InformAt, Send, Invoke o Reply)
 - Su código, marcando en color azul aquella parte del código se genera automáticamente debido a la invocación del servicio EDROOM correspondiente.

CCExplorerManager



Function Edition

Declaration: `GGuidanceTCExec()`

Brief

```
return VCurrentTC.IsGuidanceTC();
```

Function Edition

Declaration:

Brief

Standard Library Includes

```

{
    CDTCHandler * pSGuidance_Data = EDROOMPoolCDTCHandler.AllocData();

    // Complete Data
    *pSGuidance_Data=VCurrentTC;

    GuidanceCtrl.send(SGuidance,pSGuidance_Data,&EDROOMPoolCDTCHandler);
}

```

EDROOM Service

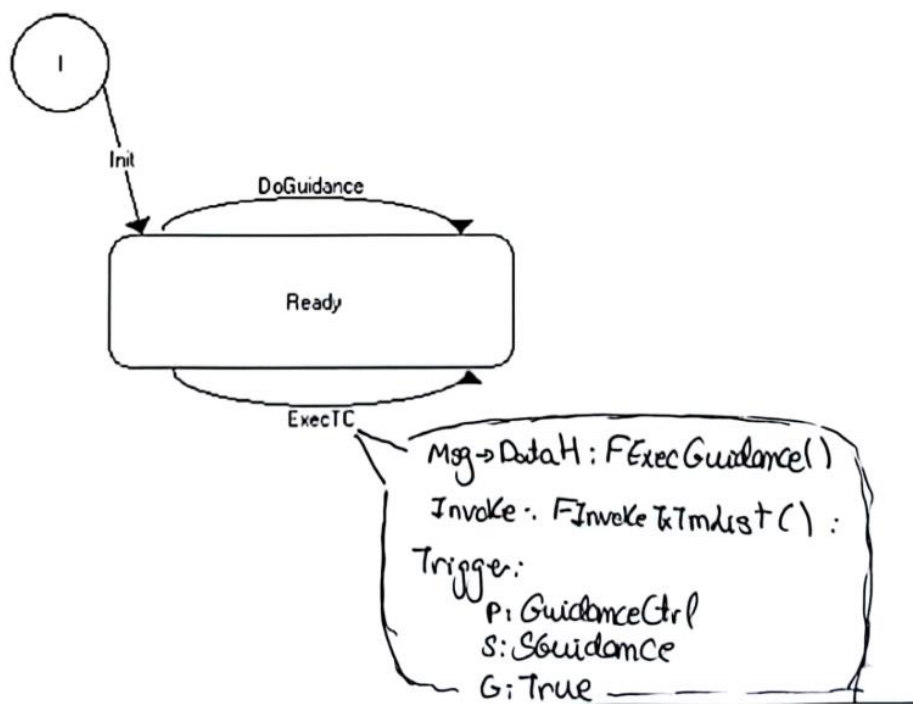
send

Port

Signal

Data Class

CCGuidance



Declaration:
FExecGuidance ()

Brief

Standard Library Includes

```

{
    CDTCHandler & varSGuidance = *(CDTCHandler *)Msg->data;

    // Data access
    CDEventList TCExecEventList;
    PUS_GuidanceTCExecutor::ExecTC (varSGuidance,VCurrentTMList,TCExecEventList);
}

```

EDROOM Service

Msg->data

Port

GuidanceCtrl

Signal

SGuidance

Data Class

CDTCHandler

Service Request

Declaration:
FInvokeTxTMList ()

Brief

Standard Library Includes

```

{
    CDTMList * pSTxTM_Data = EDROOMPoolCDTMList.AllocData();

    // Complete Data
    *pSTxTM_Data=VCurrentTMList;
    VCurrentTMList.Clear();

    MsgBack=TMChannelCtrl1.invoke(STxTM,pSTxTM_Data,&EDROOMPoolCDTMList);
}

```

EDROOM Service

invoke

Port

TMChannelCtrl

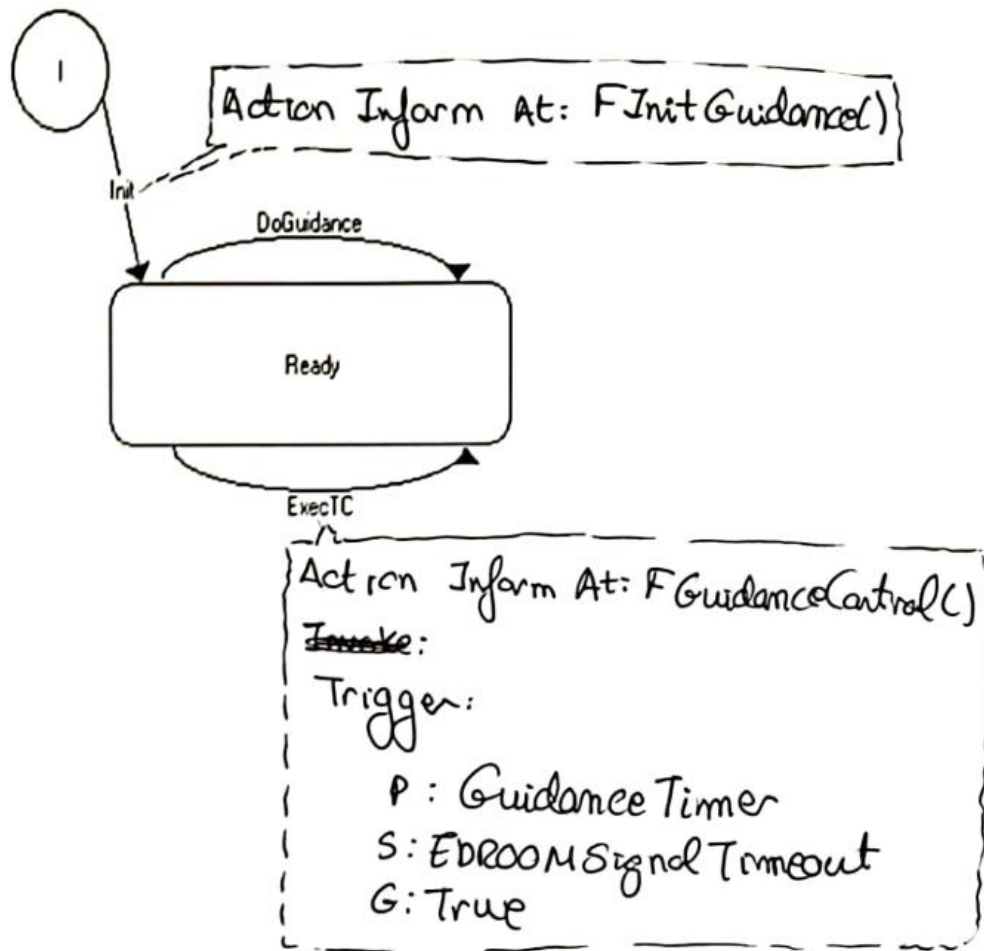
Signal

STxTM

Data Class

CDTMList

Service Request



VARIABLES:

Pr_Time VNextTimeout;

CDTMList VCurrentTMList;

```

{
    Pr_Time time;

    // Programamos el timer que realiza el control
    time.GetTime(); // Get current monotonic time
    time+=Pr_Time(0,100000); // Add X sec + Y microsec
    VNextTimeout=time;

    GuidanceTimer.InformAt( time );
}
        
```

EDROOM Service

InformAt

Port

Signal

Data Class

| | |
|--------------|--------------------|
| Declaration: | FGuidanceControl() |
|--------------|--------------------|

Brief

Standard Library Includes

```
{
    Pr_Time time;
```

```
VNextTimeout+= Pr_Time(0,100000); // Add X sec + Y microsec
time=VNextTimeout;
PUSService129::GuidanceControl(); //Realizamos el control cada 100ms, sin deriv
```

```
GuidanceTimer.InformAt( time );
}
```

EDROOM Service

InformAt

Port

GuidanceTimer

Signal

EDROOMSignalTimeout

Data Class

CDTMList

InformAt

Port

GuidanceTimer

Signal

EDROOMSignalTimeout ▼

Data Class

CDTMList

Service Request