

Proyecto Final Sistema de Gestión de Datos a Bordo

Raquel San Andrés Navarro

Junio 2024

1. Entregable 1

En este primer apartado se mostrarán los escenarios seguidos para la ejecución de los TC[129,1] y TC[129,2] y la aplicación periódica del algoritmo de control con un periodo de 100 ms. Para ello, me he basado en la estructura del escenario del servicio 3 debido a su carácter periódico.

La aplicación periódica del algoritmo de control (PUSService129::GuidanceControl()) se hace mediante el siguiente escenario, donde se realizará con un periodo de 100 ms:

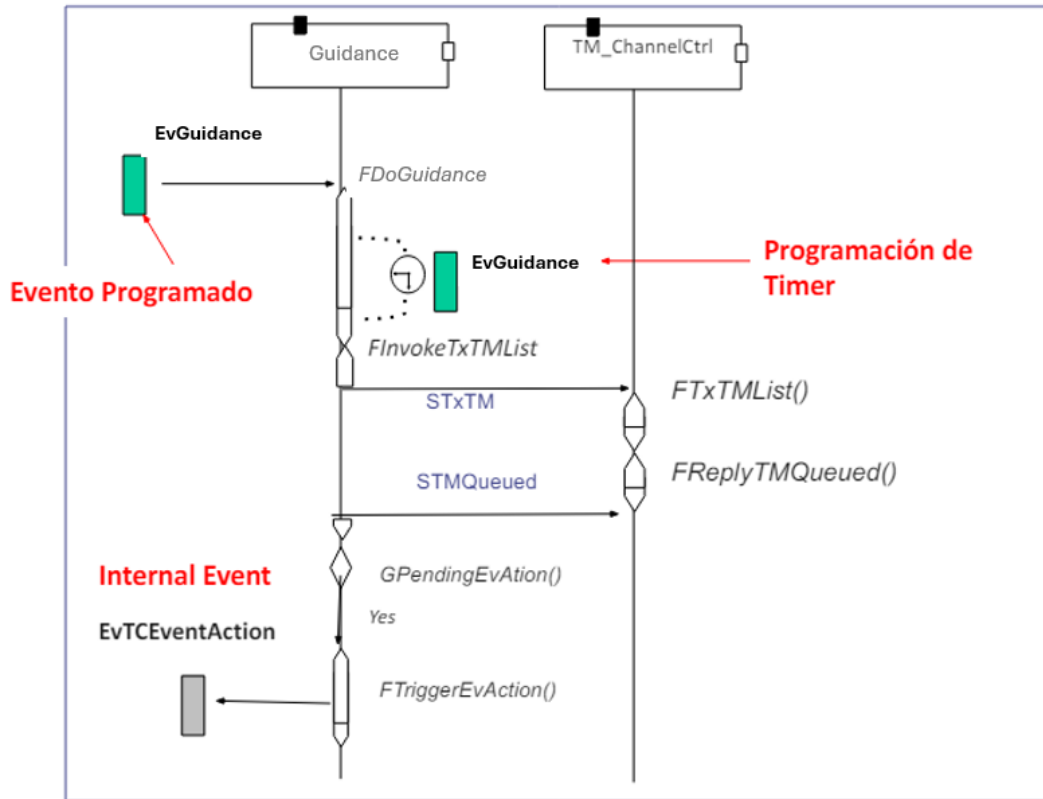


Figura 1: Escenario EvGuidance periódico

Esto se generará en EDROOM con el timer y la función **FDoGuidance()**.

Para la ejecución de los TC[129,1] y TC[129,2], necesitamos primero modificar el escenario de *EvAcceptedTC*, añadiendo un nuevo evento, *EvGuidanceTC*, en el que, una vez se acepten los TC, se ejecuten estos dos nuevos.

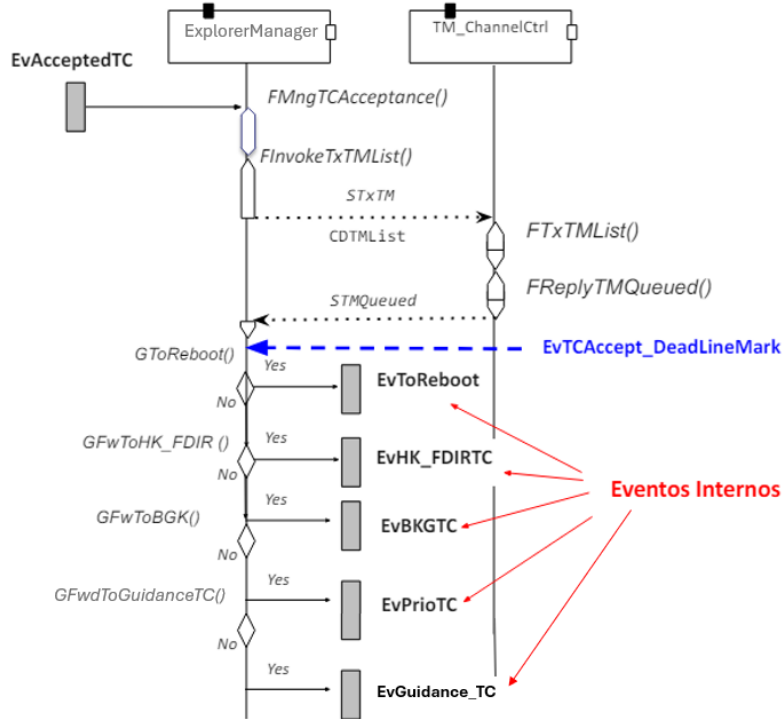


Figura 2: Escenario *EvAcceptedTC*

Y por último, se crea el escenario del *EvGuidanceTC*, en el que se ejecutarán los TC del servicio 129.

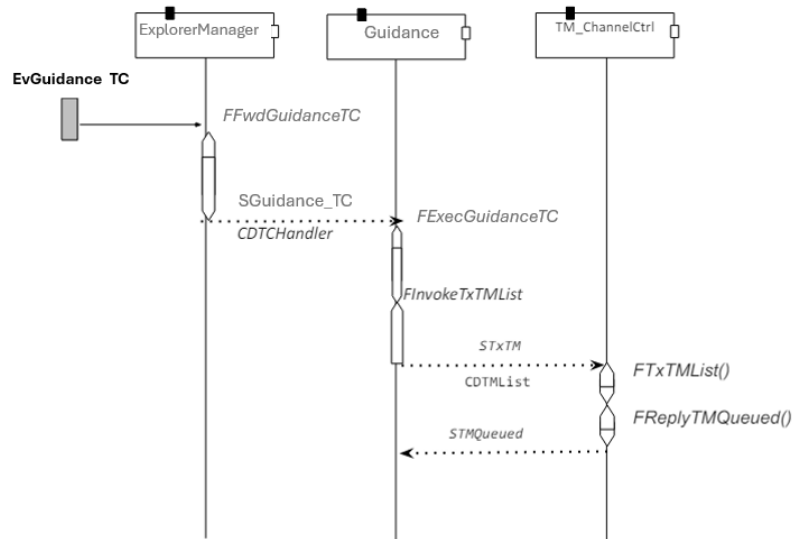


Figura 3: Escenario *EvGuidanceTC*

2. Entregable 2

En este entregable se define la clase protocolo que se ha tenido que definir para unir el componente Guidance con ExplorerManager. Este protocolo se ha definido como se muestra a continuación:

CPGuidanceCtrl

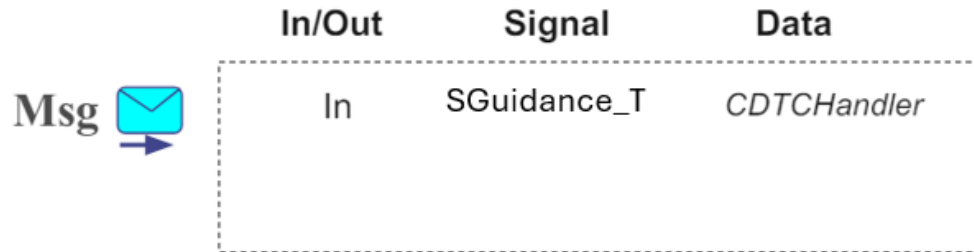


Figura 4: Definición protocolo CPGuidanceCtrl

Que en EDROOM se ha definido de la siguiente manera:

The screenshot shows the EDROOM software interface for defining a protocol. At the top, the 'Name' field is set to 'CPGuidanceCtrl'. Below this are two tabs: 'Design' (selected) and 'Analysis'. The 'Design' tab is divided into three main sections: 'Input Messages:', 'Output Messages:', and 'Protocol Brief'. The 'Input Messages:' section contains a list with 'SGuidance_TC' selected. The 'Output Messages:' section is empty. The 'Protocol Brief' section is also empty. Below these sections are three buttons: 'New Input Message', 'New Output Message', and 'Delete Message'. To the right of the 'Protocol Brief' section is a 'Message Edition Box' containing the following fields: 'Signal Name:' with the value 'SGuidance_TC', 'Data Class:' with a dropdown menu showing 'CDTCHandler', and two radio buttons for 'Synchronous Invoke' (unselected) and 'Asynchronous' (selected). There is also a 'Synchronous Reply To ---->' dropdown menu. An 'Edit Message' button is located at the bottom of the 'Message Edition Box'.

Figura 5: Declaración protocolo CPGuidanceCtrl en EDROOM

3. Entregable 3

En este entregable se hace el diseño de la interfaz de la clase componente CCGuidance. La estructura final de *UAHExplorer* queda de la siguiente manera:

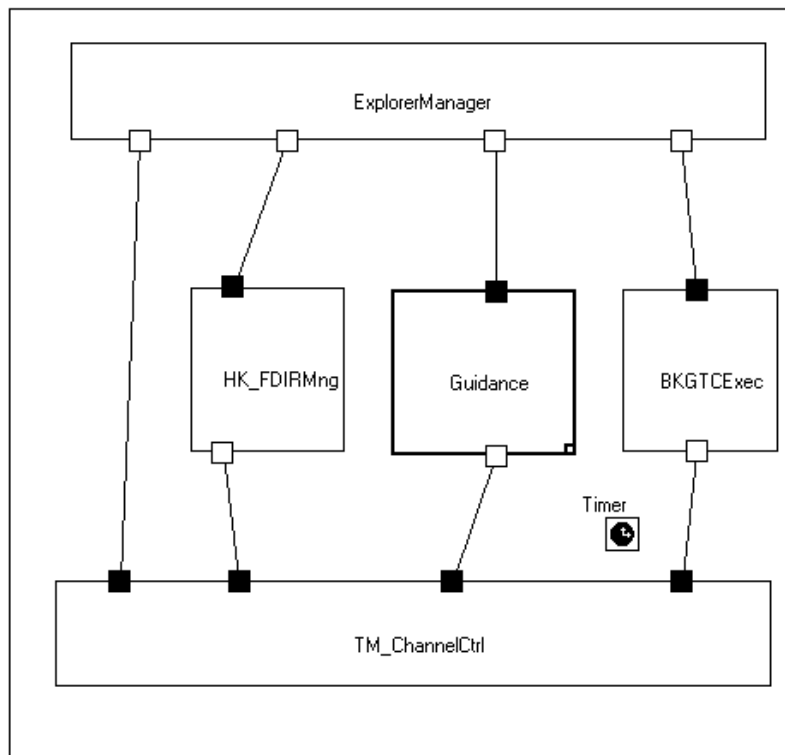


Figura 6: Estructura UAHExplorer

En el componente Guidance (CCGuidance) he definido los siguientes puertos:

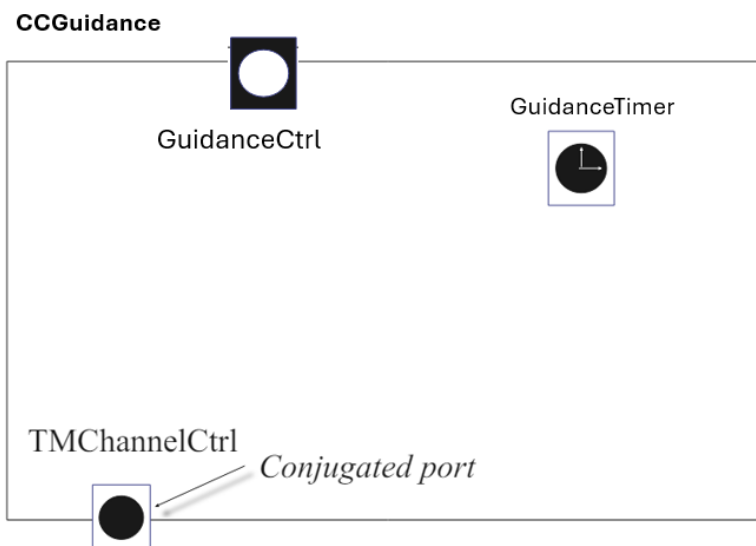


Figura 7: Puertos CCGuidance

En el componente CCExplorerManager, a adido un puerto de comunicaciones con el nuevo protocolo:

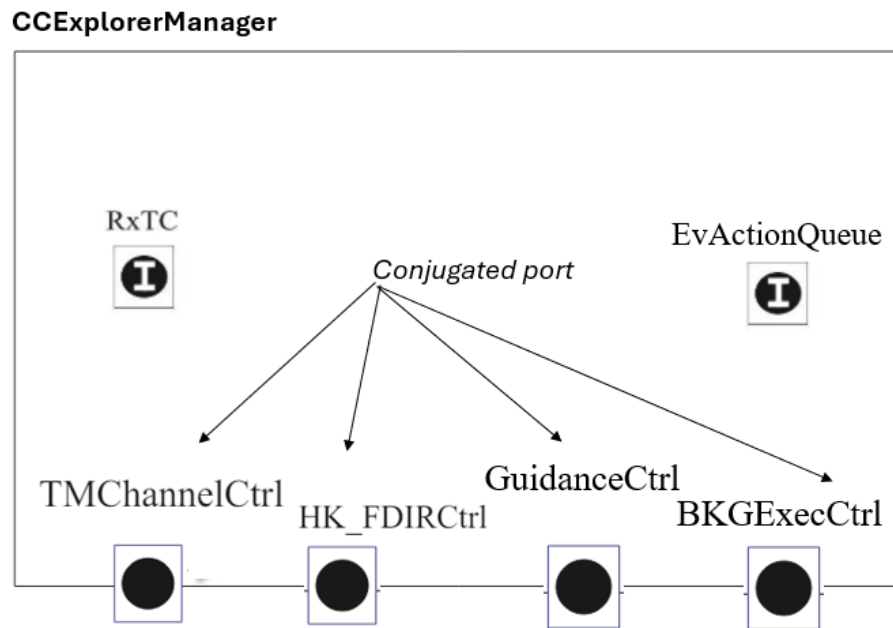


Figura 8: Puertos CCExplorerManager

En el componente *CCTMChannelCtrl*, a adimos un nuevo puerto para comunicar el componente nuevo de Guidance con  el mediante el protocolo CPTMChannelCtrl:

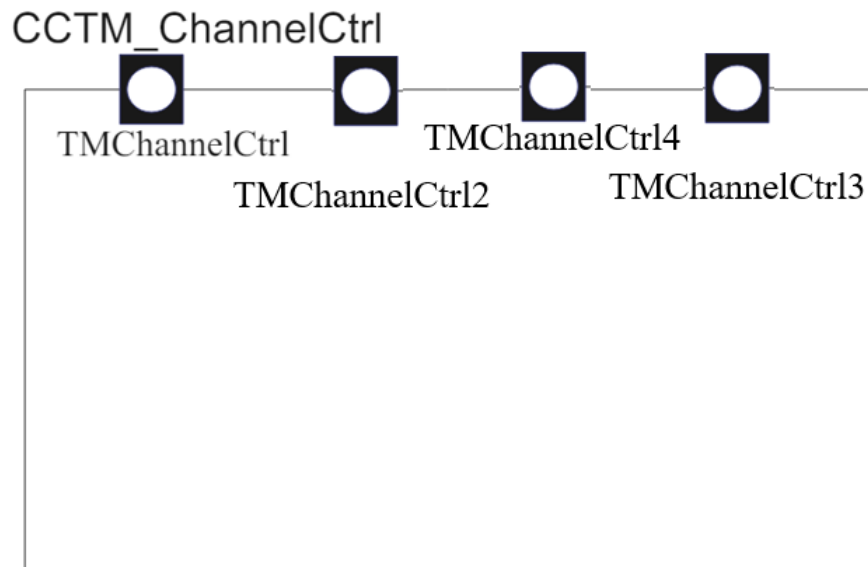


Figura 9: Puertos CCTMChannelCtrl

4. Entregable 4

En este entregable se mostrará el diseño del comportamiento de la clase componente CCGuidance. Empezamos con la máquina de estados del componente Guidance:

CCGuidance

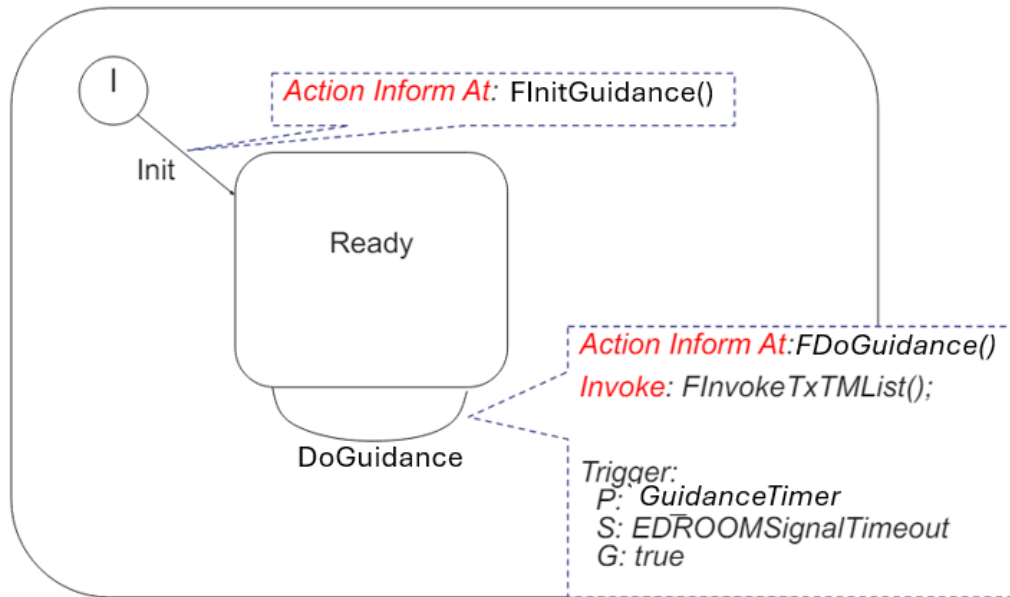


Figura 10: Máquina de estados CCGuidance

CCGuidance

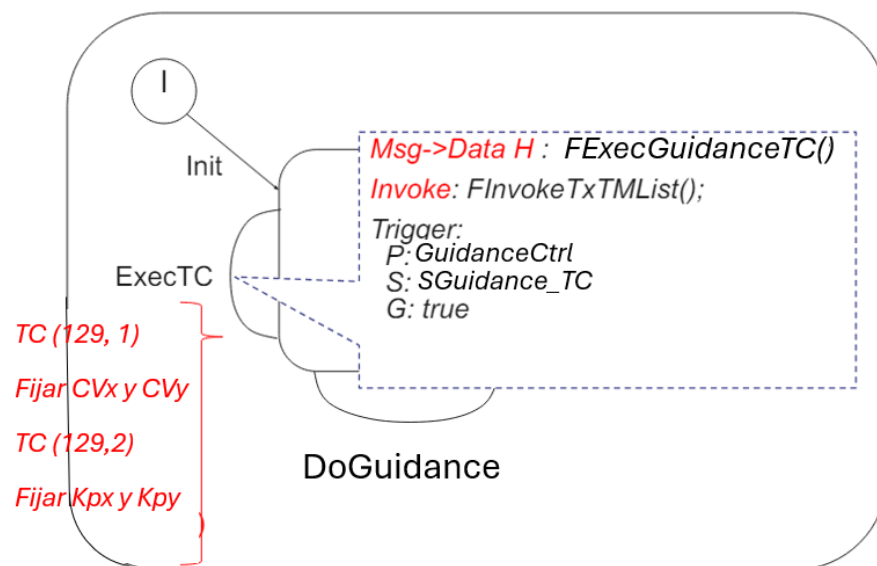


Figura 11: Máquina de estados CCGuidance

Las variables definidas en este componente han sido:

Vars:

Pr_Time VNextTimeout;

CDTMList VCurrentTMList;

Figura 12: Variable definidas en CCGuidance

Por último, los códigos de las funciones creadas serán:

```
void FInitGuidance(){
    Pr_Time time;
    time.GetTime(); // Get current monotonic time
    time+=Pr_Time(0,100000);
    VNextTimeout=time;
    PUSService129::Init (); //InitPUService 129
    GuidanceTimer.InformAt(time);
}

void FDoGuidance(){
    Pr_Time time;
    VNextTimeout+= Pr_Time(0,100000);
    time=VNextTimeout;
    PUSService129::GuidanceControl();
    GuidanceTimer.InformAt(time);
}

void FInvokeTxTMList(){
    CDTMList * pSTxTM_Data = EDROOMPoolCDTMList.AllocData();
    *pSTxTM_Data=VCurrentTMList;
    VCurrentTMList.Clear();
    MsgBack=TMChannelCtrl.invoke(STxTM,pSTxTM_Data,
        &EDROOMPoolCDTMList);
}
```

Absolute Timer

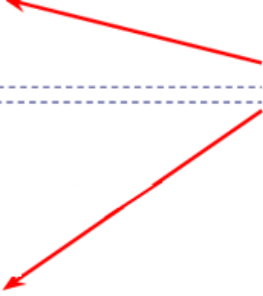


Figura 13: Funciones rama Init y DoGuidance

```

void FExecGuidanceTC() {
    CDTCHandler & varSGuidance_TC = *(CDTCHandler *) Msg->data;
    CDEventList TCExecEventList;
    PUS_GuidanceTCExecutor ::ExecTC(varSGuidance_TC,VCurrentTMList,TCExecEventList);
}

```

Figura 14: Función rama ExecTC

Ahora hay que modificar el CCExplorerManager, añadiendo una nueva rama 'FwdToGuidanceTC':

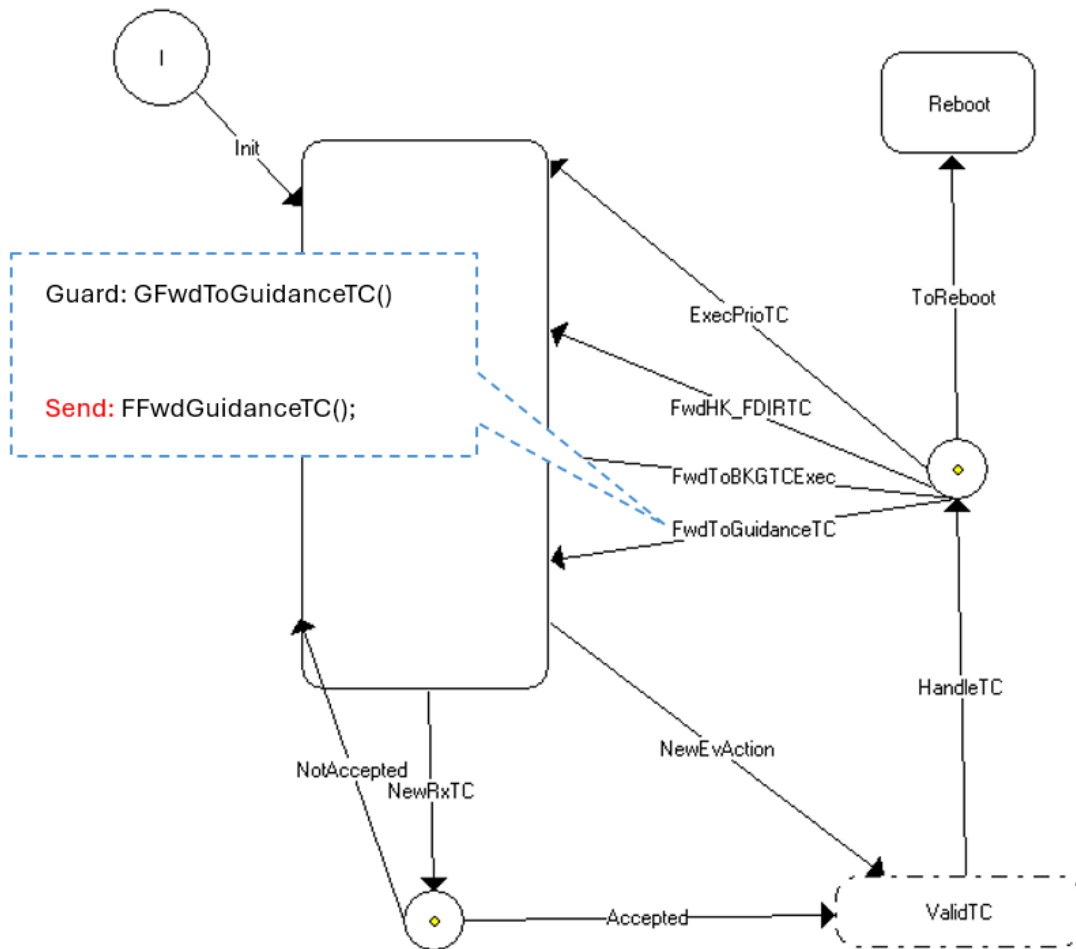


Figura 15: Máquina de estados de CCExplorerManager

Cuyas funciones son:


```
void FwdGuidanceTC (){
    CDTCHandler * pSGuidance_TC_Data = EDROOMPoolCDTCHandler.AllocData();
    *pSGuidance_TC_Data=VCurrentTC;
    GuidanceCtrl.send(SGuidance_TC, pSGuidance_TC_Data,
                     &EDROOMPoolCDTCHandler);
}
```

```
bool GFwdToGuidanceTC (){

    return VCurrentTC.IsGuidanceTC();
}
```

Figura 16: Funciones FwdToGuidanceTC

Por último creamos la pool y el módulo:

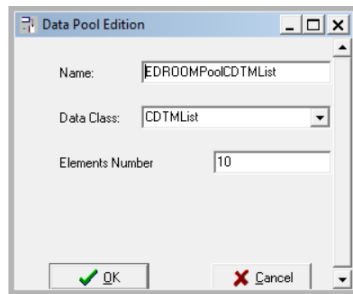


Figura 17: Pool

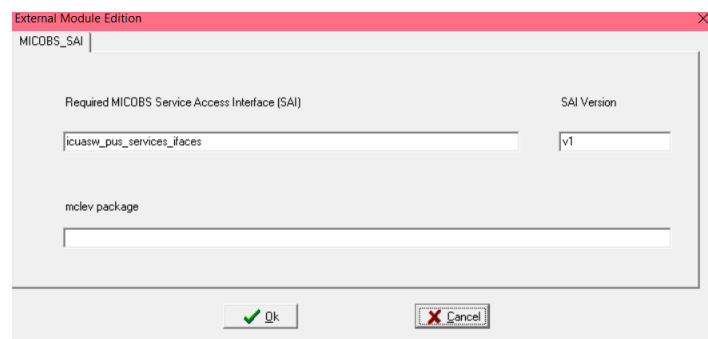


Figura 18: Módulo