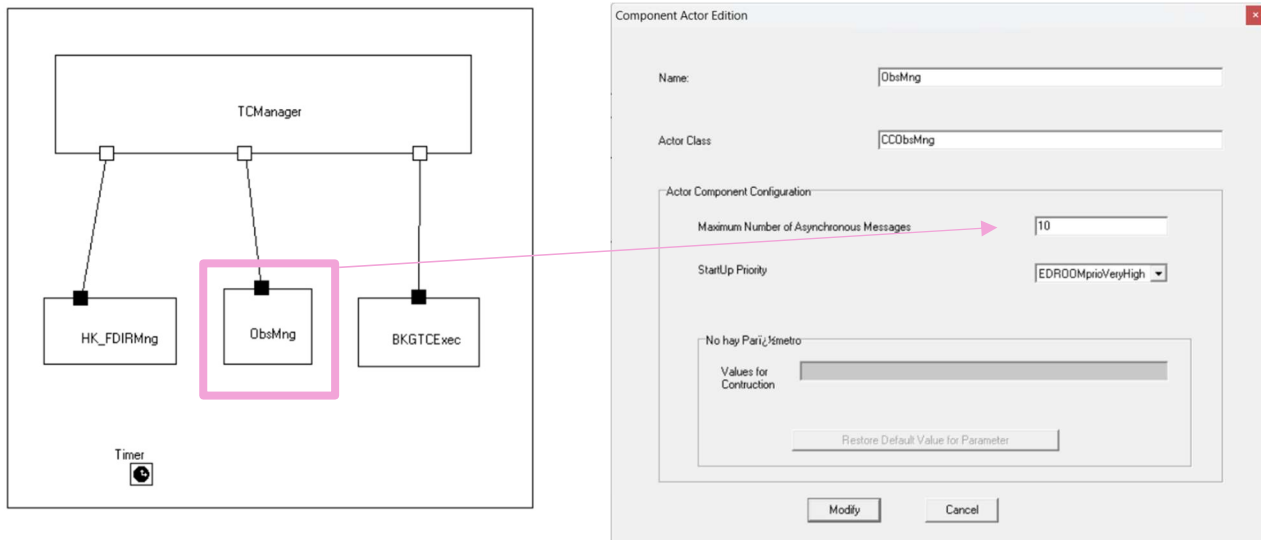
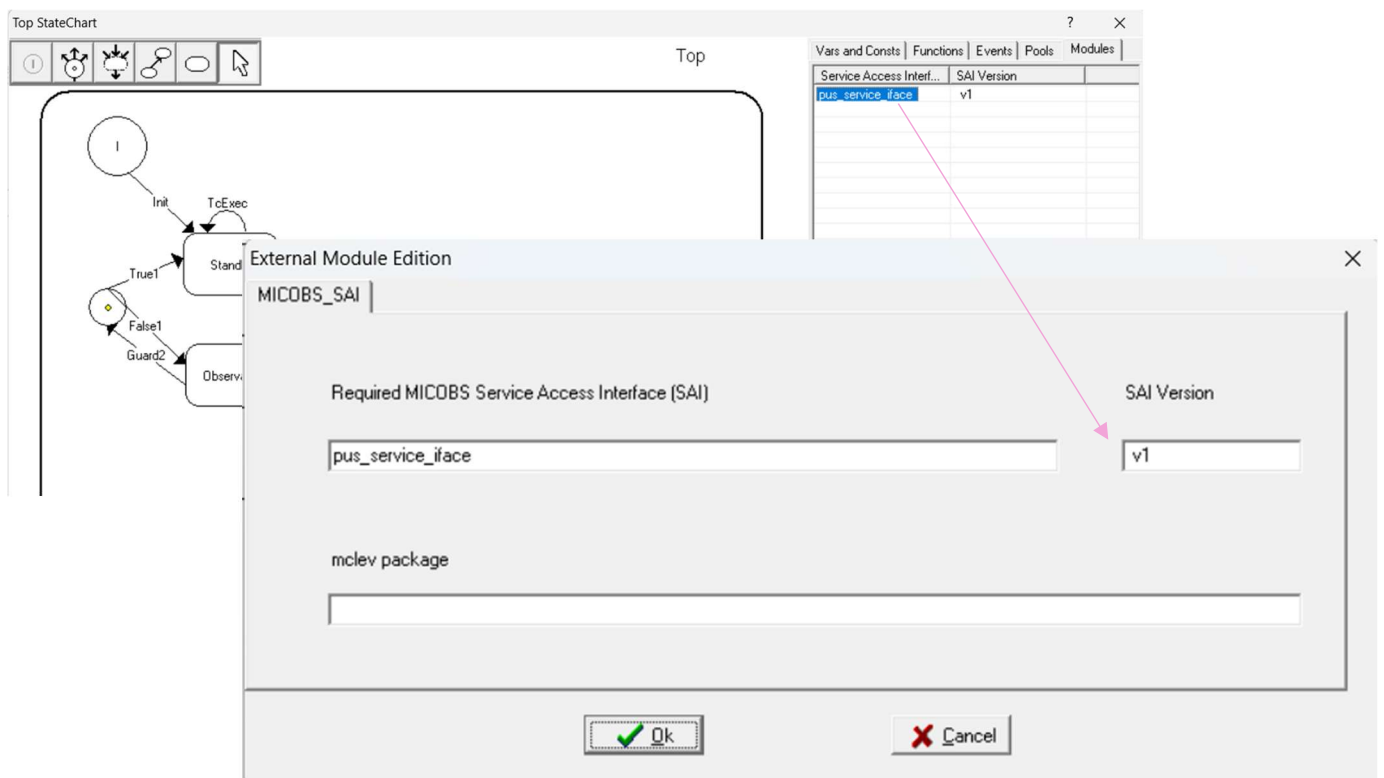


# PROYECTO FINAL

## PRIORIDAD DE LA CLASE COMPONENTE CREADA Y ACCESOS

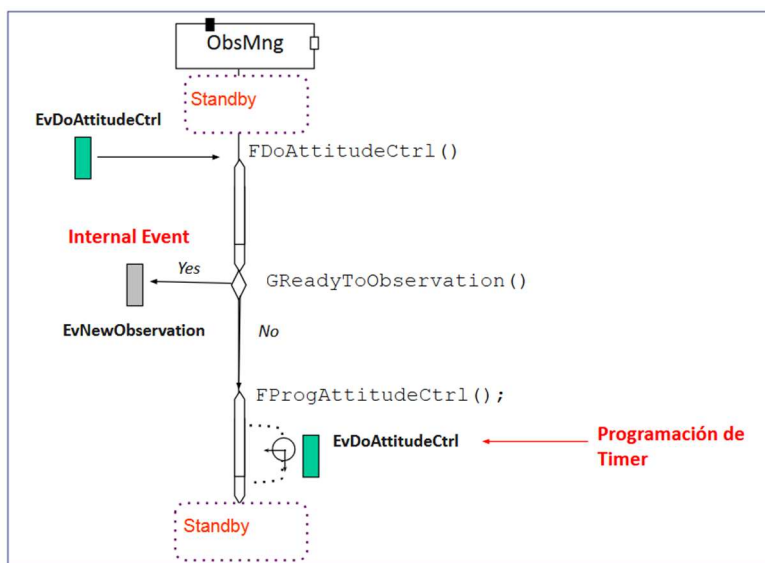
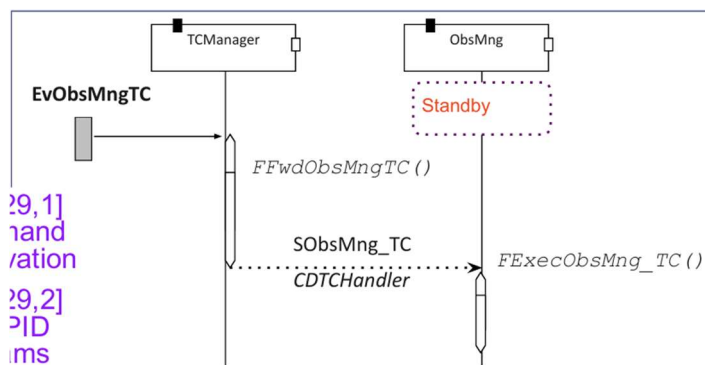
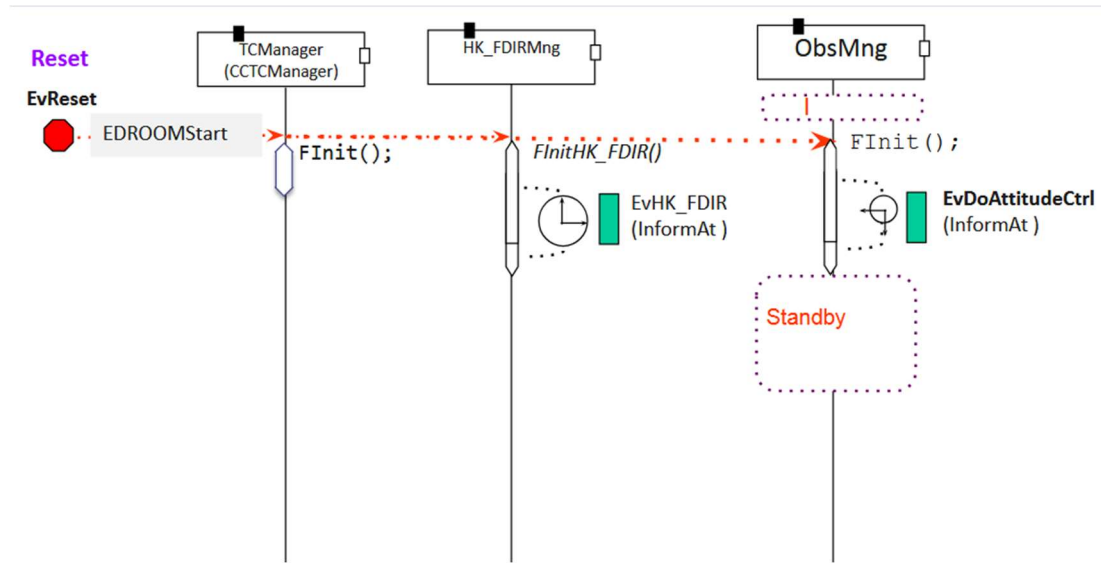


Hemos creado esta clase componente y le hemos puesto la prioridad más alta



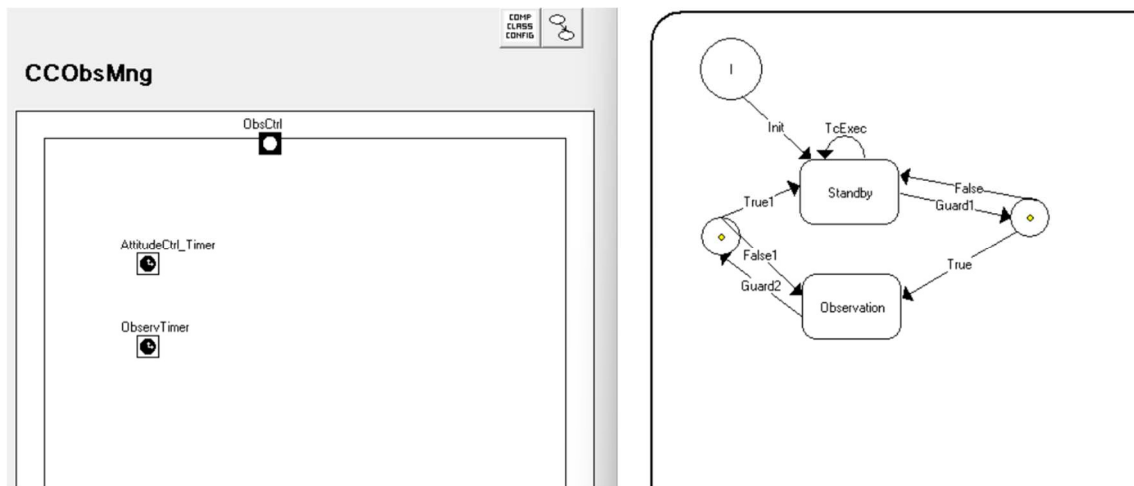
Ponemos a que modulo tiene acceso este componente

## CREAMOS LO NECESARIO PARA SEGUIR EL ESQUEMA DEL OBSMNG



Para ello vamos a crear los puertos necesarios por los que el ObsMng puede comunicarse y su diagrama en base a las funciones que este va a ir realizando

Programación de Timer



Para llegar a este resultado los pasos han sido los siguientes:

- 1- Crear el puerto externo

The 'Port Configuration' dialog box shows the following settings:

- Name: ObsCtrl
- Conjugated: ☐
- Type: External
- Protocol Class: CPObsCtrl

Buttons: Modify, Cancel

- 2- Crear los dos timers que se van a necesitar:

The 'Port Configuration' dialog box shows the following settings:

- Name: AttitudeCtrl\_Timer
- Conjugated: ☐
- Type: Internal
- Protocol Class: EDROOMTimingSAP

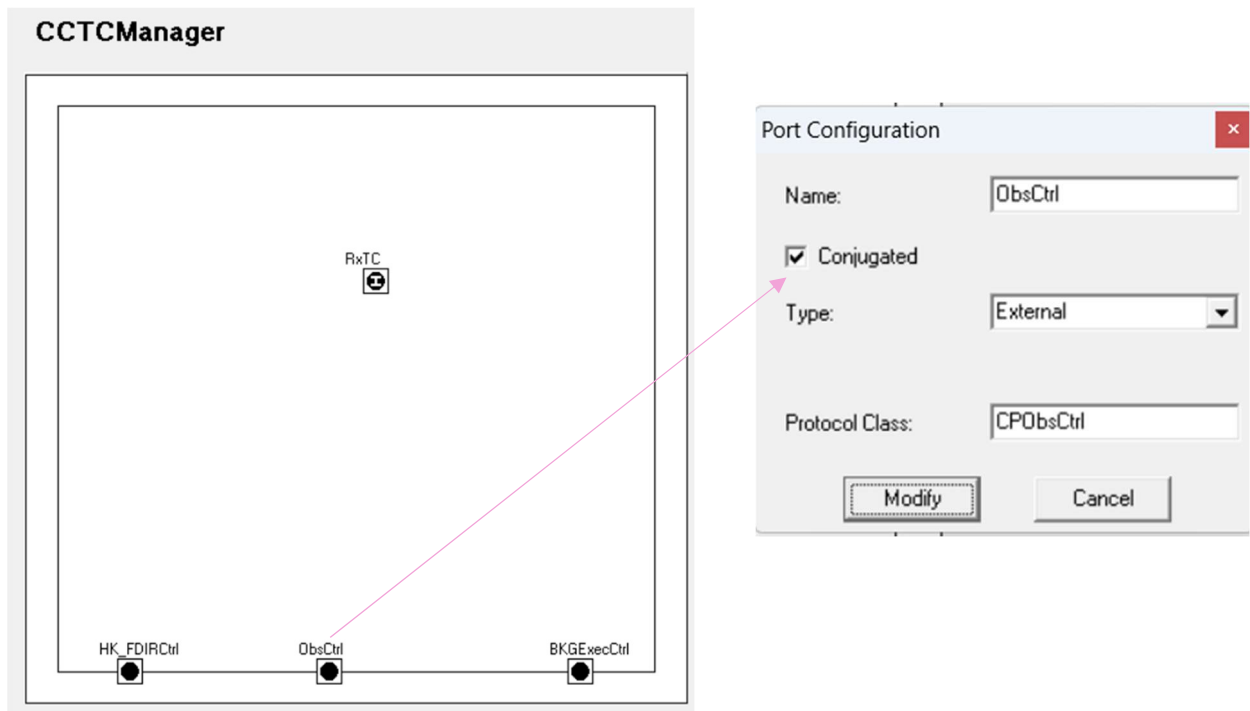
Buttons: Modify, Cancel

The 'Port Configuration' dialog box shows the following settings:

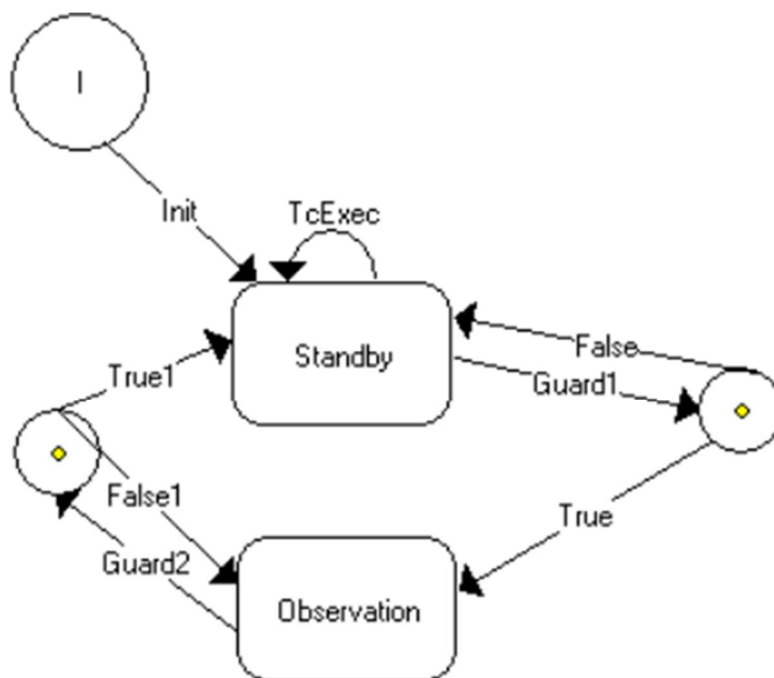
- Name: ObservTimer
- Conjugated: ☐
- Type: Internal
- Protocol Class: EDROOMTimingSAP

Buttons: Modify, Cancel

- 3- Conjugar el puerto del TcManager



4- Vamos a explicar todas las acciones que van el diagrama de flujo con capturas



**TENER EN CUENTA ESTE ESQUEMA DE FORMA GLOBAL PARA PODER ENTENDER LA EXPLIACIÓN DE CADA CACHITO IMPLEMENTADO**

Transition Edition

Design | Analysis

Name:

Trigger

port:  Signal:

Guard:

Transition Handler

Msg->Data Handler:

Action:

Send:

Invoke:

MsgBack->Data Handler:

Function Edition

Declaration:

Brief

```

{
  Pr_Time time;

  //Timing Service useful methods
  time.GetTime(); // Get current monotonic time
  time.Add(0,100000); // Add X sec + Y microsec
  VNextTimeout=time;
}

```

Standard Library Includes

EDROOM Service

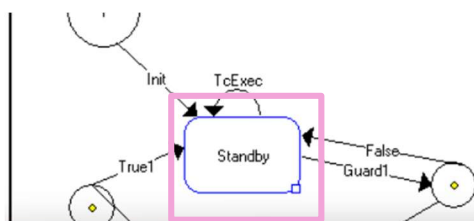
InformAt:

Port:

Signal:

Data Class:

Definimos la función  
FInit\_Obs():



State Edition

Name:

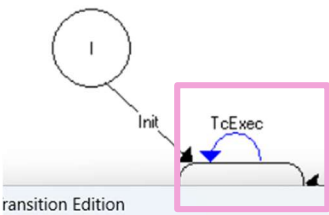
Entry Action:

Send at Entry:

Exit Action:

Send at Exit:

Creamos un nuevo estado  
que se lleva Standby que es  
al que se llega siguiendo la  
transición de Init



ransition Edition

Design | Analysis

Name:

Trigger  
 port:  Signal:   
 Guard:

Transition Handler  
 Msg->Data Handler:

Action:

Send:

Invoke:

MsgBack->Data Handler:

Function Edition

Declaration:

Brief

```

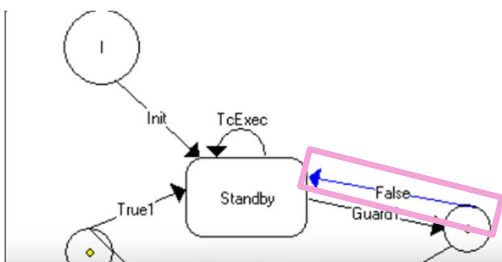
{
    CDTCHandler & varSObsMng_TC = *(CDTCHandler *)Msg->data;

    varSObsMng_TC.ExecTC();
}
  
```

Standard Library Includes

EDROOM Service  
 Msg->data:   
 Port:   
 Signal:   
 Data Class:

Creamos la transición  
TcExec: donde vamos a crear  
la función FObsCtrl\_Exec()



Branch Edition

Design | Analysis

Name:

Guard:

Branch Handler  
 Trans MsgBack->Data Handler:

Function Edition

Declaration:

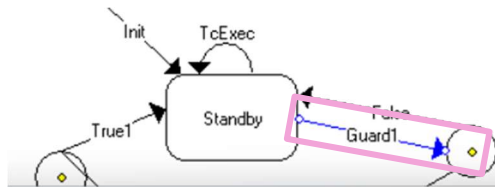
Brief

```

return pur_s>rvicel29_is_observation_ready();
  
```

Standard Library Includes

Creamos la transición de  
False y aplicamos la guarda  
correspondiente:  
GReadyToObservation()



Transition Edition

Design | Analysis

Name:

Trigger

port:  Signal:

Guard:

Transition Handler

Msg->Data Handler:

Action:

Send:

Declaration:

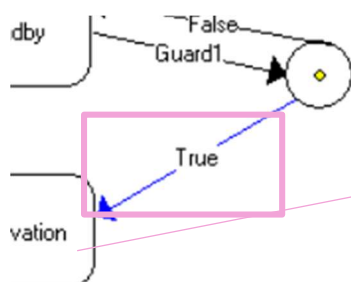
Brief

Standard Library Includes

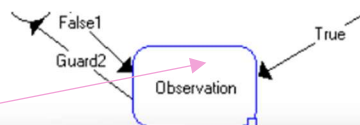
```

pus_servicel29_do_attitude_ctrl();
  
```

Seguimos con la trasiicion  
con la guarda 1 q si es  
true va a llamar a la  
funci3n  
FDoAttitudeCtrl\_Timer y  
creamos dicha funci3n



Del choiseipoint con la trasiicion  
true nos vamos al otro estado que  
a va ser Observation :



State Edition

Name:

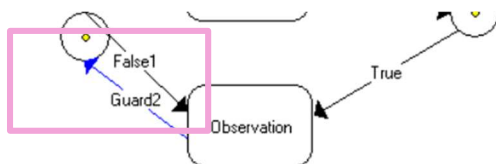
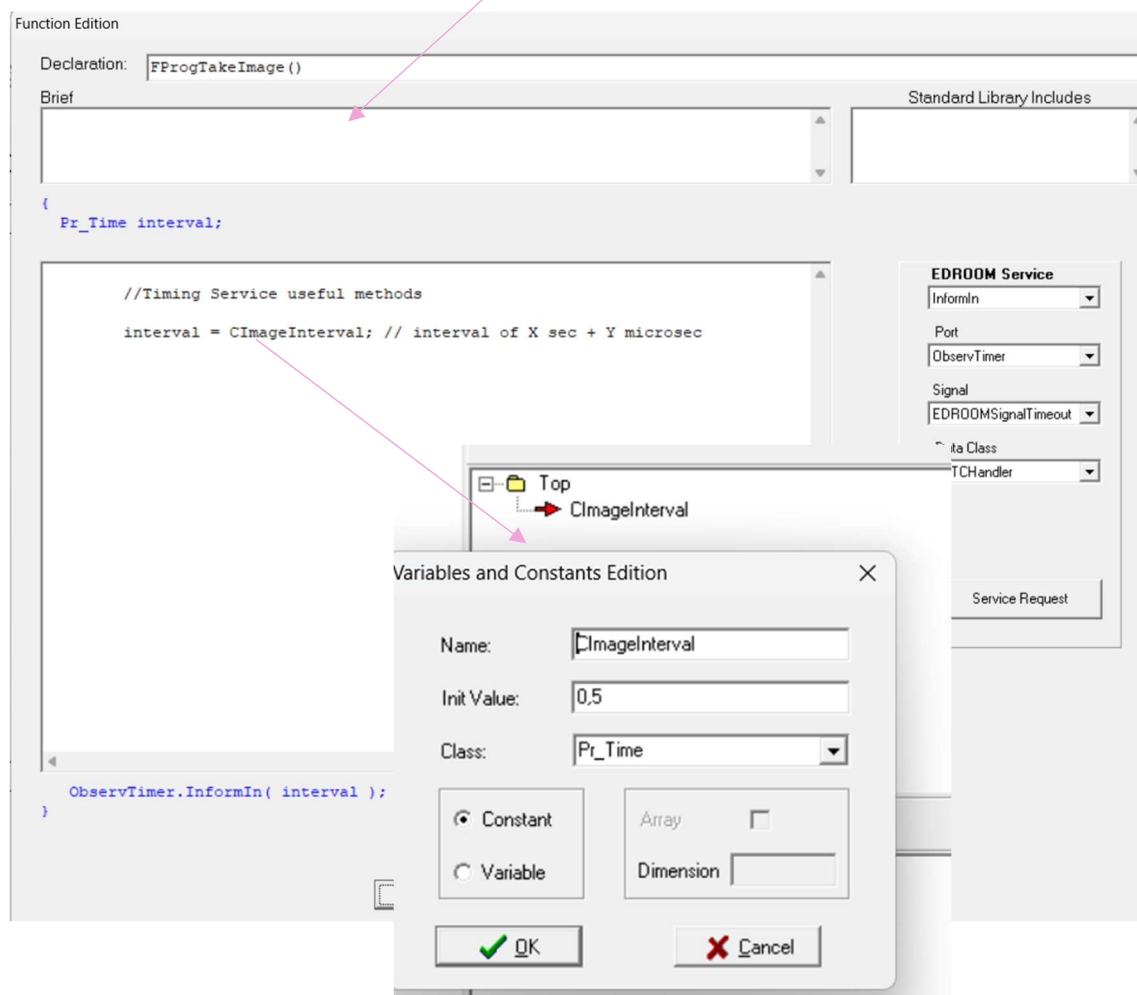
Entry Action:

Send at Entry:

Exit Action:

Send at Exit:

En este nuevo estado definimos la acción FProgTakeImage() no olvidar crear la constante



Hacemos esas dos transiciones respecto del choisepoint y el estado Observation , la de false no hace falta comentarla es con un return como habíamos visto antes y vamos a ver la de la guarda 2 donde debemos crear la acción FTakeImage();

Transition Edition

Design Analysis

Name: `Guard2`

Trigger

port: `ObservTimer` Signal: `EDROOMSignalTimeout`

Guard: `true`

New Guard

Transition Handler

Msg->Data Handler:

Action:

`FTakeImage();`

Function Edition

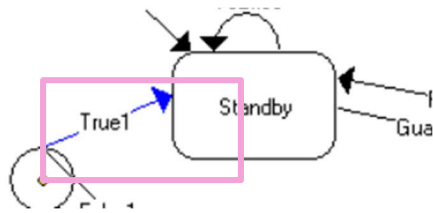
Declaration: `FTakeImage()`

Brief

Standard Library Includes

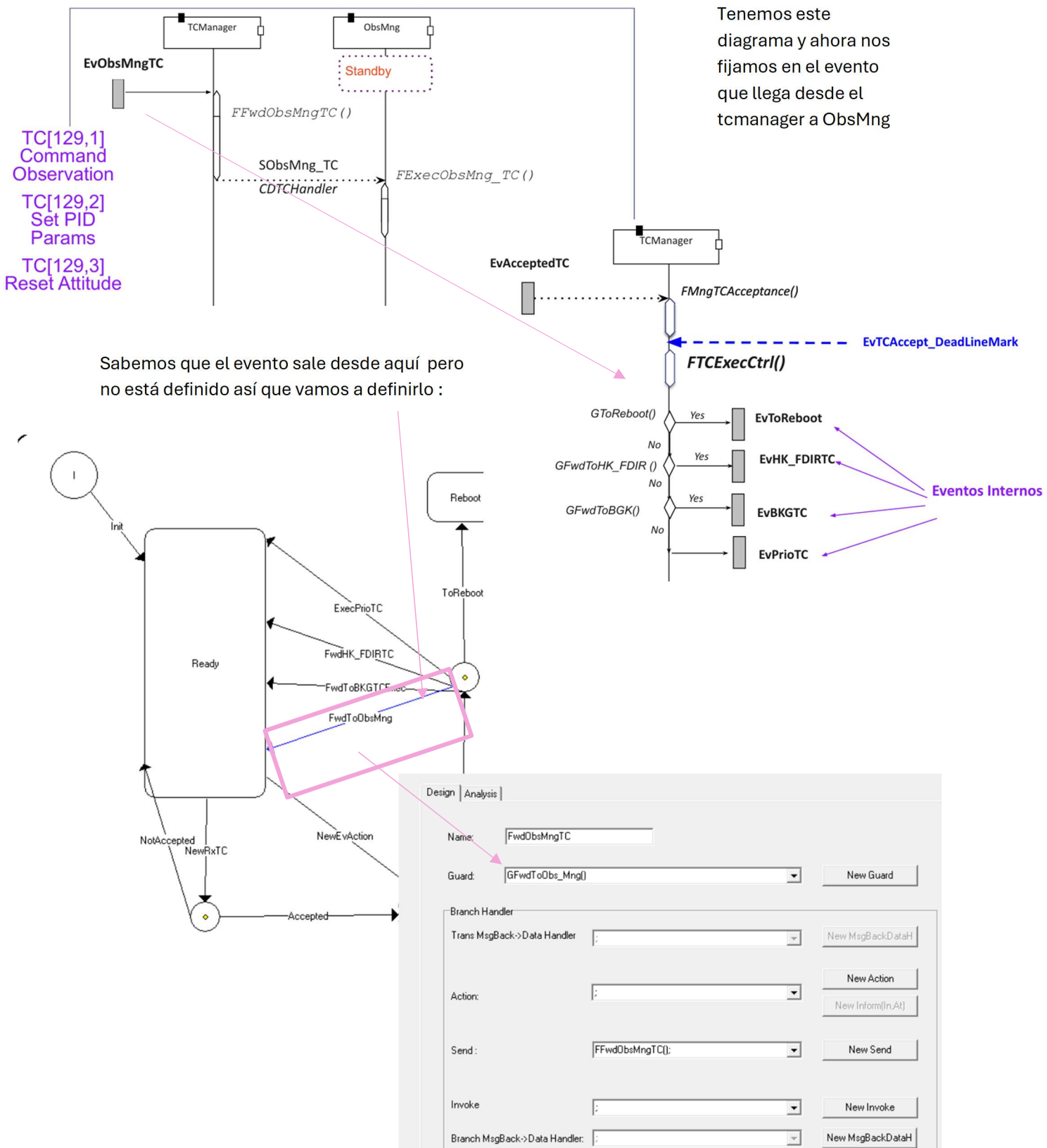
`pus_service129_take_image();`

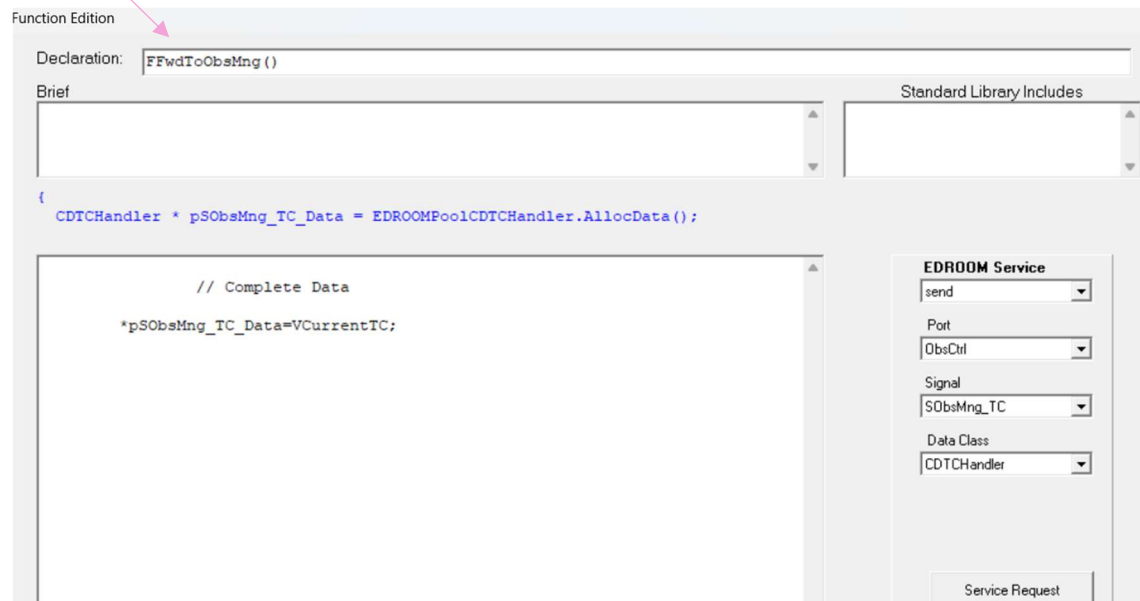




Y por ultimo volvemos de nuevo al estado de Standby con la transición true 1

## RETOQUES EN EL DIAGRAMA DE FLUJO DEL TCMANAGER





Por ultimo aquí vemos el Send que ejecuta el TC manager