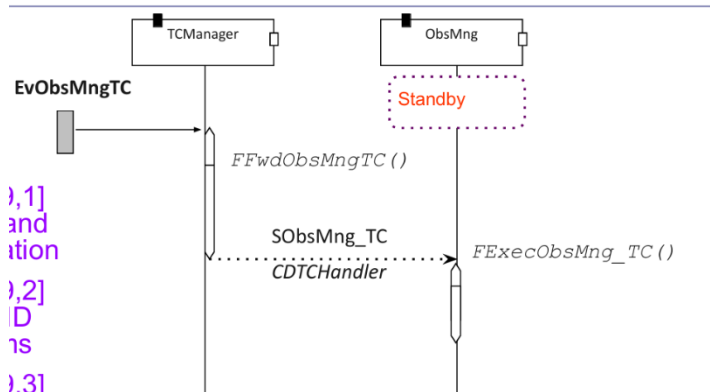


**Definición de la clase Protocolo a añadir al Modelo EDROOM, aportando toda la información de cada mensaje:**

La clase protocolo O\_Mng\_Ctrl es un puente de comunicación entre el componente TC Manager y ObsMng.



Mensaje de entrada:

SObsMng\_TC

Tipo del dato: CDTCHandler

Sin mensaje de salida

Name: CPD\_Mng\_Ctrl

Design | Analysis

Input Messages :      Output Messages:

SObsMng\_TC

Protocol Brief

Message Edition Box:

Signal Name: SObsMng\_TC

Data Class : CDTCHandler

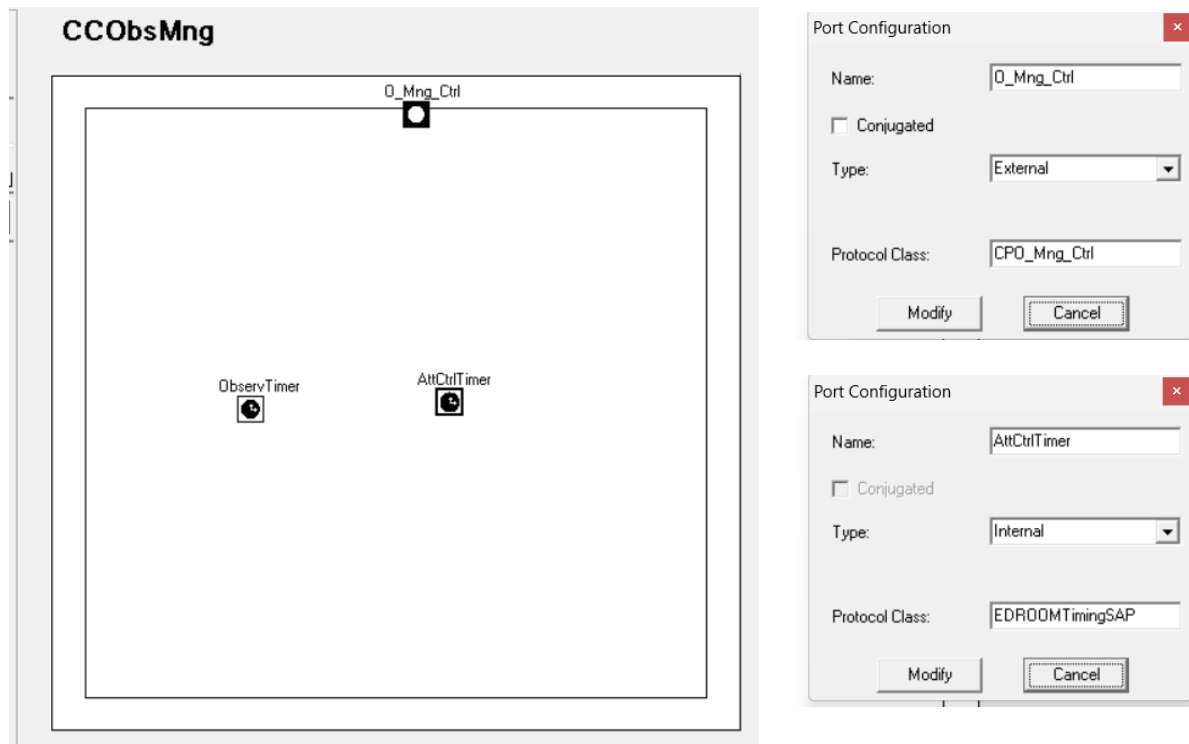
☐ Synchronous Invoke      ☒ Asynchronous

☐ Synchronous Reply To ---->     

Edit Message

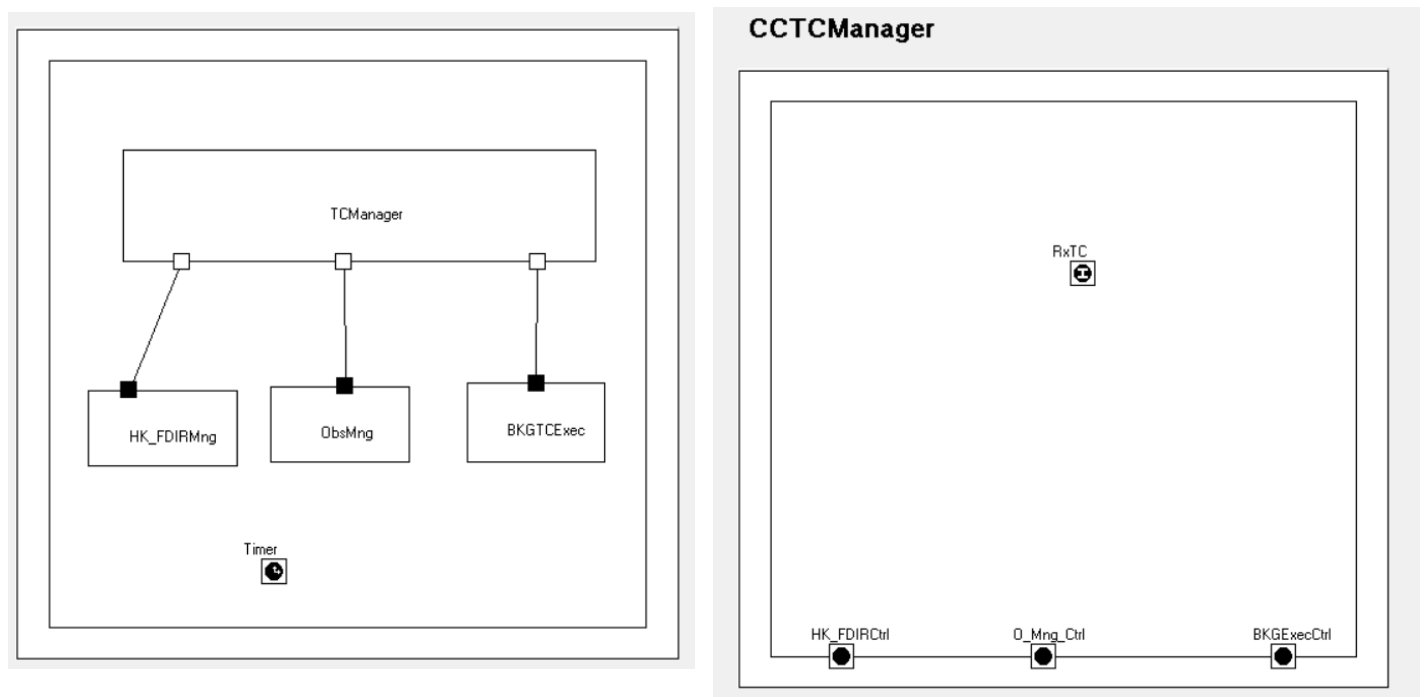
**Diseño de la interfaz de la clase componente CCObsMng empleando la misma notación gráfica que se ha proporcionado durante las prácticas, y que debe definir:**

- 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.



Un puerto de comunicaciones y dos timers (permite servicios de temporización)

- Para los puertos de comunicaciones, indicad la clase protocolo de cada puerto y el tipo de asociación (conjugada o nominal):



Como vemos tenemos 3 puertos principales que conectan el componente TC Manager con los demás. El nombre que aparece encima de cada puerto es la clase protocolo a la que está asociado. En el componente **ObsMng**, como hemos visto antes, el puerto es nominal, mientras que en **TC Manager** los puertos son conjugados:

Port Configuration

Name:

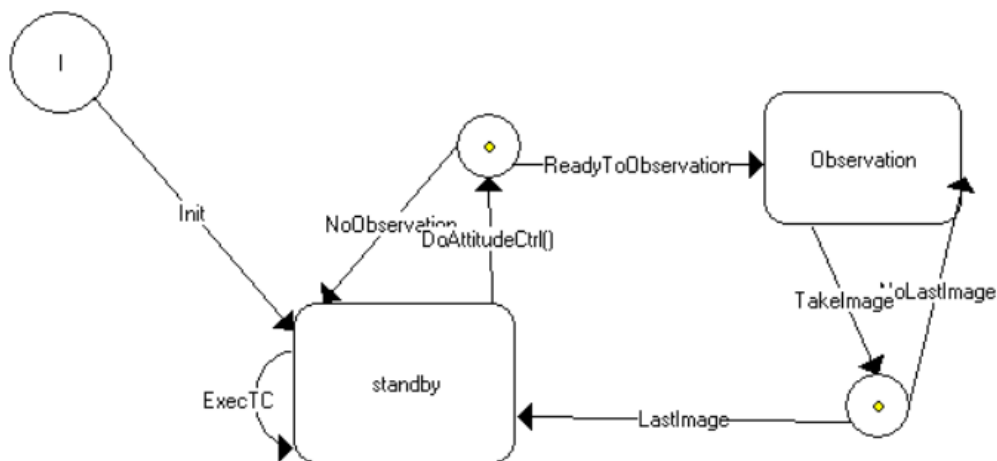
☒ Conjugated

Type:

Protocol Class:

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

- La máquina de estados de la clase componente.



- La declaración de las Variables y constantes de la clase componente.

Variables and Constants Edition

Name:

Init Value:

Class:

☐ Constant ☒ Variable

Array ☐

Dimension

Variables and Constants Edition

Name:

Init Value:

Class:

☒ Constant ☐ Variable

Array ☐

Dimension

- Definición del trigger de cada transición y la guarda de cada rama

### Primera transición: DoAttitudeCtrl

Trigger:

Transition Edition

Design | Analysis

Name: DoAttitudeCtrl

Trigger

port: AttCtrlTimer Signal: EDROOMSignalTimeout

Guard: true New Guard

Transition Handler

Msg->Data Handler: ; New MsgDataHand

Action: FDoAttitudeCtrl(); New Action

Send: ; New Send

Invoke: ; New Invoke

MsgBack->Data Handler: ; New MsgBackDataH

OK Cancel

Function Edition

Declaration: FDoAttitudeCtrl()

Brief

pus\_servicel29\_do\_attitude\_ctrl()

Ramas:

- No observation:

Branch Edition

Design | Analysis

Name: NoObservation

Guard: true New Guard

Branch Handler

Trans MsgBack->Data Handler: ; New MsgBackDataH

Action: FProgAttitudeCtrl(); New Action

Send: ; New Send

Invoke: ; New Invoke

Branch MsgBack->Data Handler: ; New MsgBackDataH

Declaration: `FProgAttitudeCtrl()`

Brief

---

```

{
    Pr_Time time;

    //Timing Service useful methods

    //time.GetTime(); // Get current monotonic time
    //time.Add(X,Y); // Add X sec + Y microsec

    VNextTimeout+= Pr_Time(0,1000000);
    time = VNextTimeout;

    AttCtrlTimer.InformAt( time );
}

```

Standard Library Includes

**EDROOM Service**

InformAt

Port

AttCtrlTimer

Signal

EDROOMSignalTimeout

Data Class

CDTCHandler

Service Request

Cap  
Gu  
cap

## 2. Ready to Observation:

Branch Edition

Design Analysis

Name:

Guard:

Branch Handler

Trans MsgBack->Data Handler:

Action:

Send:

Invoke:

Branch MsgBack->Data Handler:

Declaration: `GReadyToObservation()`

Brief

---

```

return pus_servicel29_is_observation_ready();

```

Declaration: `FTToObservation()`

Brief

---

```

pus_servicel29_start_observation();

```

## Segunda transición: Take Image

Trigger:

Design | Analysis

Name:

Trigger

port:  Signal:

Guard:

Transition Handler

Msg->Data Handler:

Action:

Send:

Invoke:

MsgBack->Data Handler:

Declaration:

Brief

Ramas:

### 1. No last Image:

Design | Analysis

Name:

Guard:

Branch Handler

Trans MsgBack->Data Handler:

Action:

Send:

Invoke:

Branch MsgBack->Data Handler:

Declaration:

Brief

Standard Library Includes

```

{
  Pr_Time interval;

  interval = CImageInterval;

  ObservTimer.InformIn( interval );
}

```

EDROOM Service

InformIn

Port

ObservTimer

Signal

EDROOMSignalTimeout

Data Class

CDTCHandler

Service Request

## 2. Last Image:

Branch Edition

Design

Analysis

Name:

Guard: 

New Guard

Branch Handler

Trans MsgBack->Data Handler

:

New MsgBackDataH

Action:

FCompositeEndAtt();

New Action

New Inform(In,At)

Send :

:

New Send

Invoke

:

New Invoke

Branch MsgBack->Data Handler:

:

New MsgBackDataH

Declaration:

Brief

Declaration:

- Actions a ejecutar (teniendo en cuenta que ya hemos visto varias de ellas):

FInit → asociada a una transición, de tipo action

Design | Analysis

Name:

Trigger

port:  Signal:

Guard:

Transition Handler

Msg->Data Handler:

Action:

Declaration:

Brief

Standard Library Includes

```

{
  Pr_Time time;

  time.GetTime(); // Get current monotonic time
  time+=Pr_Time(0,100000); // Add X sec + Y microsec
  VNextTimeout=time;

```

**EDROOM Service**

Port:

Signal:

Data Class:



ExecTC → asociada a una transición, de tipo MsgDataHandler

Design | Analysis

Name:

Trigger

port:  Signal:

Guard:

Transition Handler

Msg->Data Handler:

Declaration:

Brief

Standard Library Includes

```
{  
    CDTHandler & varSObsMng_TC = *(CDTHandler *)Msg->data;  
  
    varSObsMng_TC.ExecTC();  
  
}
```

**EDROOM Service**

Msg->data

Port

Signal

Data Class

FProgTakeImage → asociada a la entrada de un estado, de tipo action

State Edition

Name:

Entry Action:

Send at Entry:

Exit Action:

Send at Exit:

New Action  
New Inform(In,At)  
New Send  
New Action  
New Inform(In,At)  
New Send

OK Cancel

FEndObservation → asociada a una transición, de tipo action

Function Edition

Declaration:

Brief