

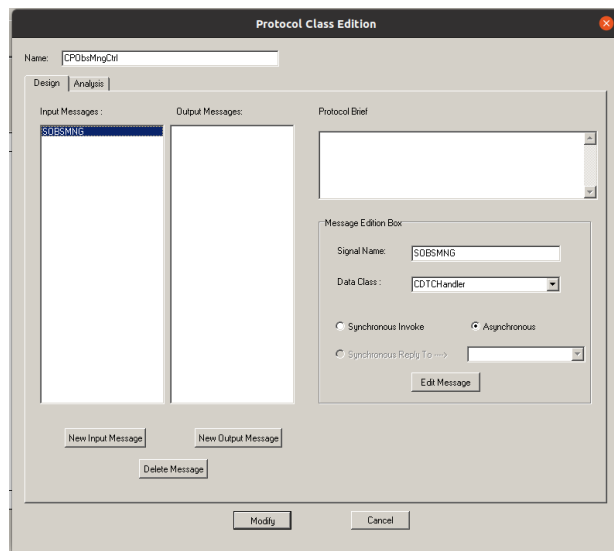
Entregable Modelo EDROOM Julien Pérez

1. Escenarios en los que está involucrado el nuevo componente ObsMng que controla la observación.

El componente ObsMng (Observation Manager) está involucrado en los escenarios de:

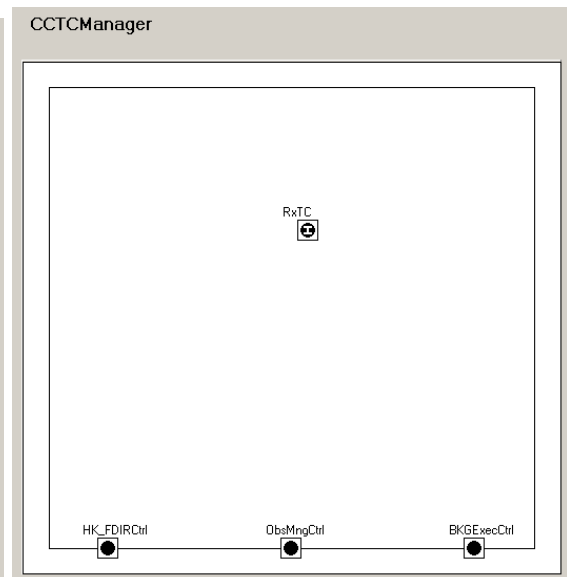
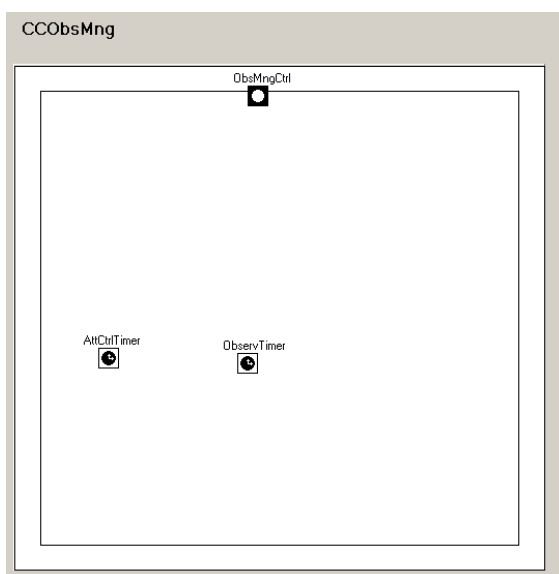
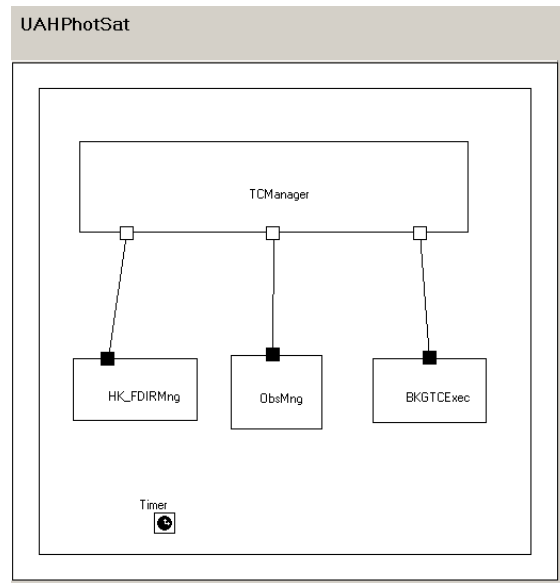
- Ejecución del control de actitud de acuerdo a la orientación del objetivo y actualización de los parámetros del sistema.
- Inicio de la observación una vez la cámara ha sido correctamente posicionada.
- Cambio de estado de standby a observación.
- Programación de la toma de fotografías.
- Cambio de estado de observación a standby tras la última fotografía.
- Tras la salida de la observación, ejecución de los telecomandos del servicio ST[129], enviados desde el TCManager.

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



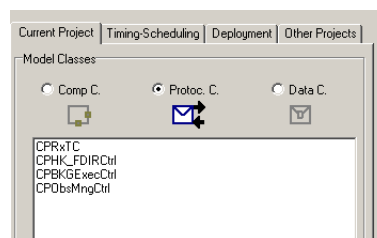
3. Diseño de la interfaz de la clase componente CCObsMng empleando la misma notación gráfica que se ha proporcionado durante las prácticas, y que debe definir:

- 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).



Para el caso de la clase ObsMng, este tiene dos relojes y un puerto de comunicación externo nominal (ObsMngCtrl) que viene dado por la clase de protocolo CPObsMngCtrl.

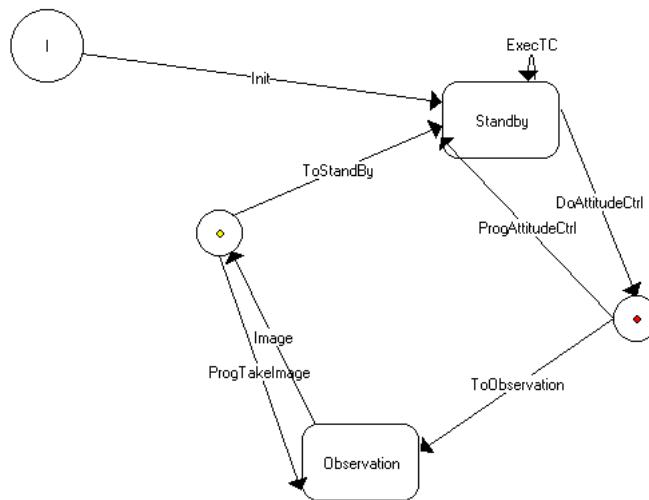
Mientras que para la clase CCTCManager, se tiene un reloj; tres puertos de comunicación externos conjugados (ObsMngCtrl, HK_FDIRCtrl y BKGExecCtrl) que conectan la clase TCManger con su respectivas clases y tiene asociadas las clases de protocolo CPObsMngCtrl, CPHK_FDIRCtrl y CPBKGExecCtrl y un puerto asociado a una interrupción (RxTC) cuya clase de protocolo asociada es CPRxTC.



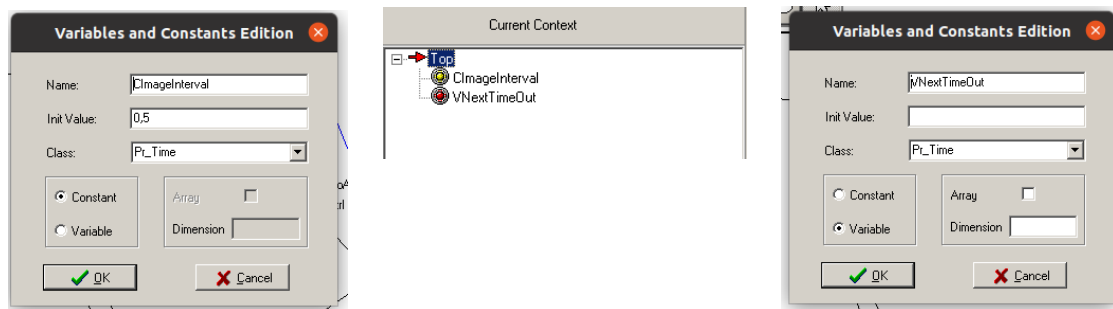
Clases de protocolo

4. Diseño del comportamiento de la clase componente CCObsMng 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.

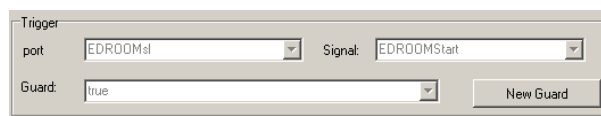


b. La declaración de las Variables y Constantes de la clase componente.

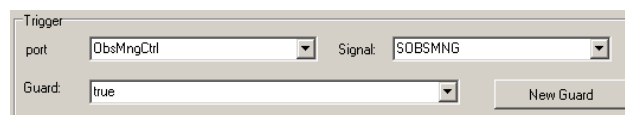


c. Definición del trigger de cada transición y la guarda de cada rama

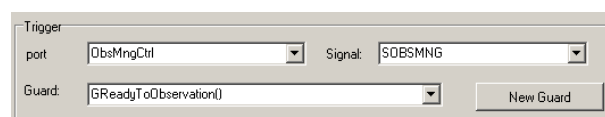
Transición Init:



Transición ExecTC:



Transición DoAttitudeCtrl:



Rama ProgAttitudeCtrl:

Name:	<input type="text" value="ProgAttitudeCtrl"/>	
Guard:	<input type="text" value="true"/>	<input type="button" value="New Guard"/>

Rama ToObservation:

Name:	<input type="text" value="ToObservation"/>	
Guard:	<input type="text" value="GReadyToObservation()"/>	<input type="button" value="New Guard"/>

Transición Image:

Trigger		
port:	<input type="text" value="ObsMngCtrl"/>	Signal: <input type="text" value="SOBSMNG"/>
Guard:	<input type="text" value="GLastImage()"/>	<input type="button" value="New Guard"/>

Rama ProgTakeImage:

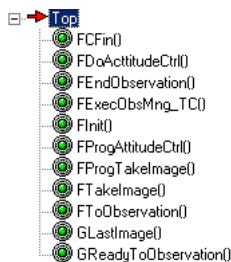
Name:	<input type="text" value="ProgTakeImage"/>	
Guard:	<input type="text" value="true"/>	<input type="button" value="New Guard"/>

Transición ToStandBy:

Name:	<input type="text" value="ToStandBy"/>	
Guard:	<input type="text" value="GLastImage()"/>	<input type="button" value="New Guard"/>

d. Actions a ejecutar, indicando:

- Si está asociada a una transición, a una rama, 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.



La función FInit() está asociada a la transición Init, en EDROOM su código se deja vacío ya que será declarado en eclipse.

La transición ExecTC tiene asociado FExecObsMng_TC(), un MsgDataHandler con código adjuntado al final.

La función FDoActitudeCtrl() está asociada a la transición DoAttitudeCtrl y en ella se hace una llamada a pus_service129_do_attitude_ctrl().

Declaration:	<input type="text" value="FDoActitudeCtrl()"/>
Brief	<div></div>
	<pre>pus_service129_do_attitude_ctrl();</pre>

Declaration: `FToObservation()`

Brief

`pus_service129_start_observation();`

Declaration: `FTakeImage()`

Brief

`pus_service129_take_image();`

Declaration: `FTakeImage()`

Brief

`pus_service129_take_image();`

Declaration: `FTakeImage()`

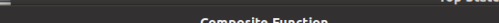
Brief

`pus_service129_take_image();`

Declaration: `FTakeImage()`

Brief

`pus_service129_take_image();`



Composite Function

Declaration:

Declaration: `VEndObservation()`

Brief

`VNextTimeOut_GetTime();`
`pus_service129_end_observation();`

Function Edition

Declaration: FExecObsMng_TC()

Brief

Standard Library Includes

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

 // Data access
 varSOBSMNG.ExecTC();

}

EDROOM Service

Msg>data

Port

ObsMngCtrl

Signal

SOBSMNG

Data Class

CDTCHandler

Service Request