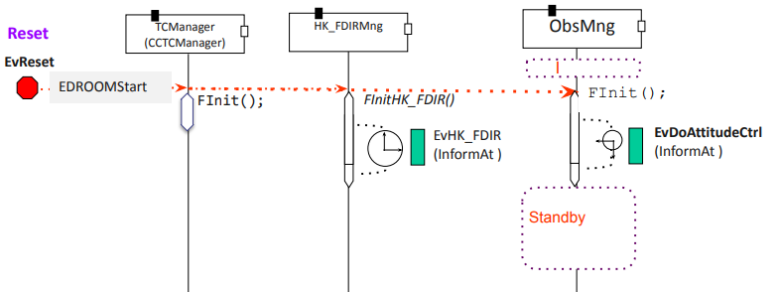
ENTREGABLES:

DESARROLLO E IMPLEMENTACIÓN DEL MODELO EDROOM

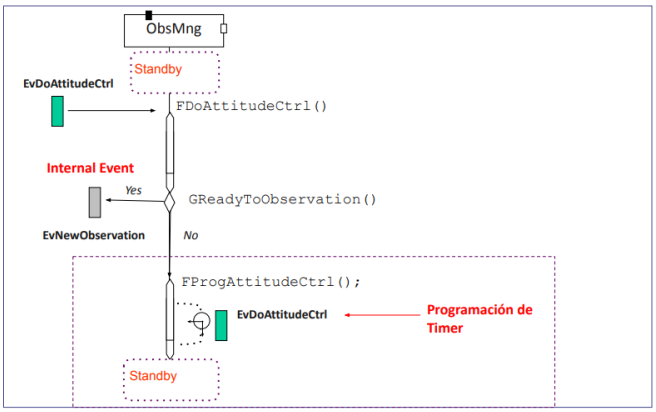
1. ***Escenarios en los que está involucrado el nuevo componente ObsMng que controla la observación***

Tal y como se refleja tanto en el Diagrama de Estados como en el modelo de Edroom, el componente ObsMng se encuentra implicado en los siguientes 4 escenarios:

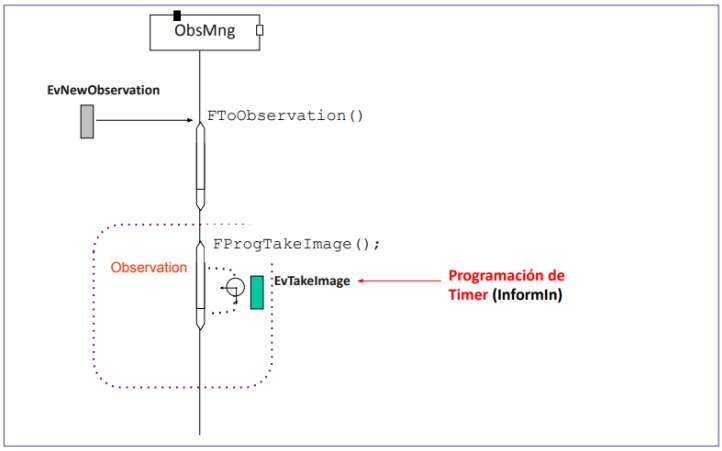
- **Inicialización**: En este escenario es donde se configura el *timer* que controla el sistema de actitud.



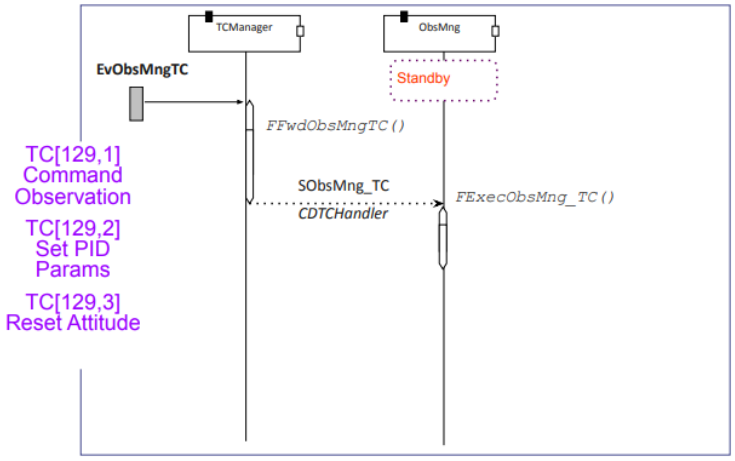
- **Standby (Espera):** Este escenario supervisa el control de actitud hasta que se alcanza el valor deseado, momento en el cual se activa el estado Observation. Si la observación no está lista, o no ha sido programada ninguna observación se regresará al estado Standby



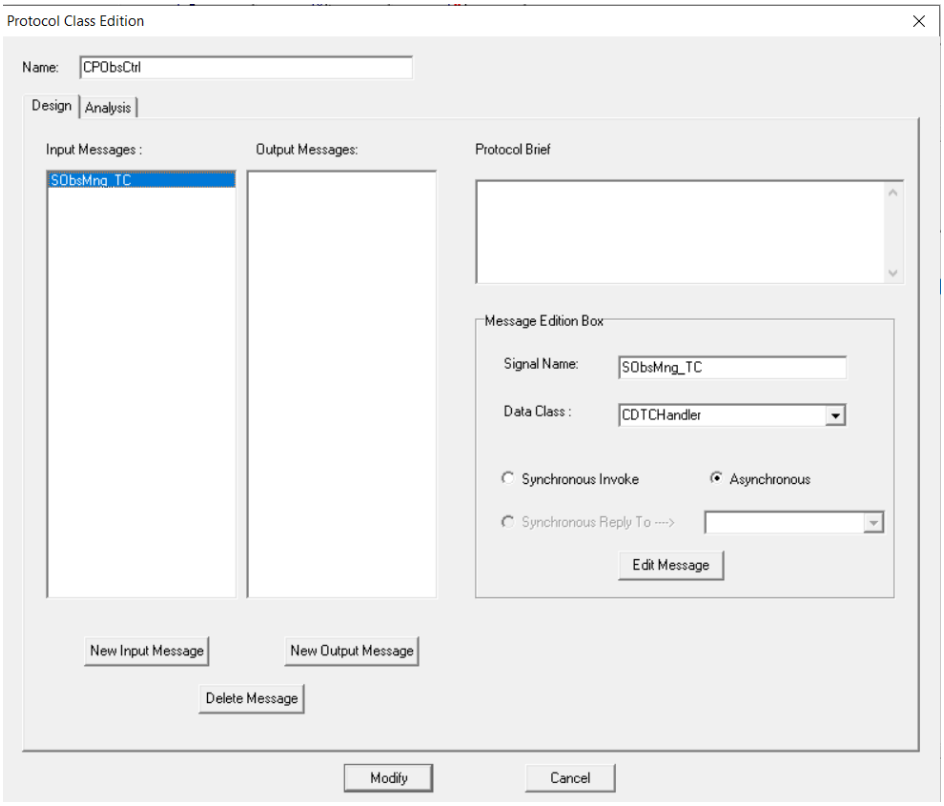
- **Observación (Captura de imágenes):** Si, por el contrario, la observación sí está lista, se ejecutará la toma de imágenes como parte de la tarea de observación, tras lo cual retorna automáticamente al estado Standby.



- **<u>Conexión con el TCManager:</u>** El componente ObsMng ejecutará todos los telecomandos del servicio ST[129] que se envían desde TCManager



2. ***<u>Definición de la clase protocolo a añadir al Modelo EDROOM, aportando toda la información de cada mensaje</u>***

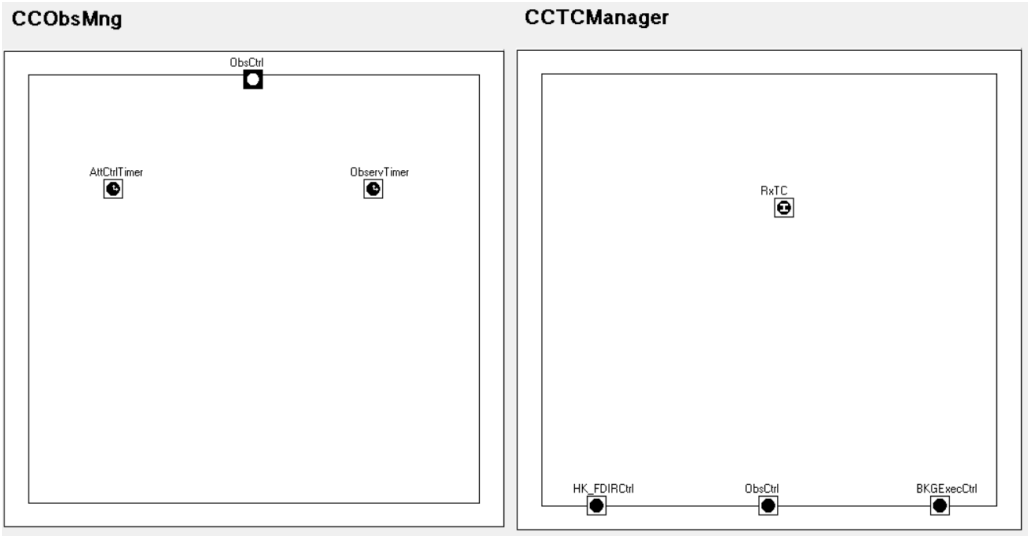


Tal y como se muestra en la imagen, la nueva clase de protocolo definida en el modelo, denominada CPObsCtrl, incluye únicamente un mensaje de entrada de tipo asíncrono con los siguientes parámetros:

- **<u>Nombre de señal</u>:** SObsMng_TC
- **<u>Clase de datos asociada</u>:** CDTCHandler

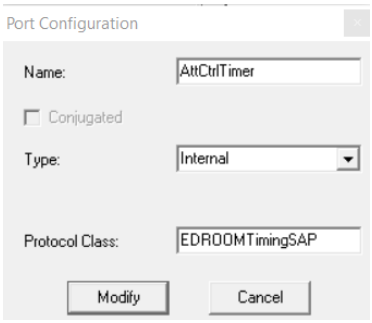
Esta configuración permite que el componente receptor procese comandos desde el TC Manager mediante una señal directa al componente ObsMng.

3. ***Diseño de la interfaz de la clase componente CCObsMng***

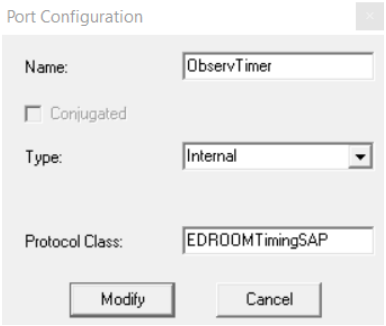


La clase componente CCObsMng presenta una interfaz compuesta por tres elementos principales:

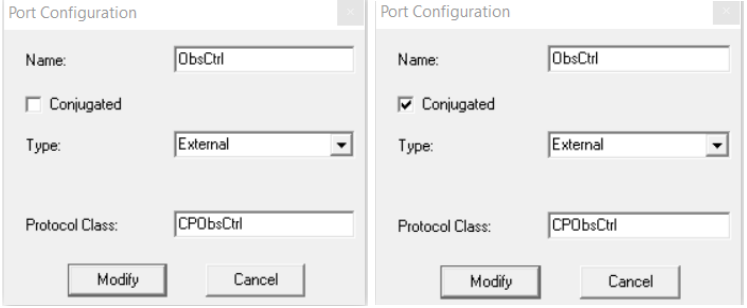
- **AttCtrlTimer**: Es el temporizador responsable del control de actitud, encargado de gestionar los intervalos de activación del sistema de orientación.



- **ObservTimer**: Temporizador dedicado a la toma de imágenes, se encarga de regular la frecuencia con la que se ejecuta la función de observación.

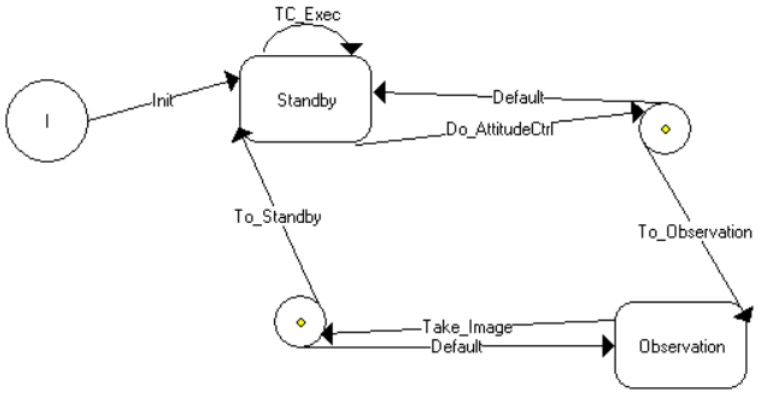


- **ObsCtrl**: Corresponde con el puerto de comunicación nominal vinculado al protocolo CPObsCtrl, el cual permite la conexión directa entre CCObsMng y CCTCManager (conjugado), facilitando la recepción de telecomandos.



4. Diseño del comportamiento de la clase componente CCObsMng

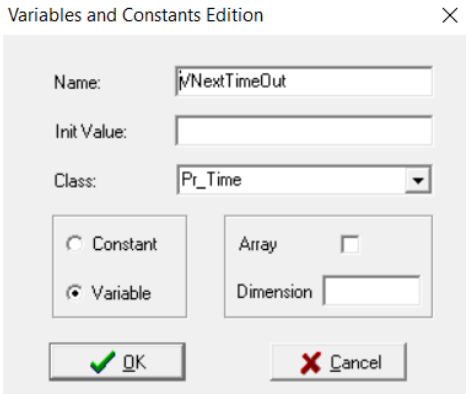
4.1 La máquina de estados de la clase componente



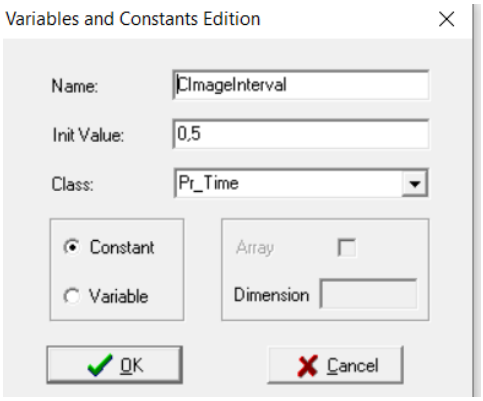
4.2 Variables y Constantes de la clase componente

En la clase componente se han definido los siguientes elementos

- **VNextTimeOut**: Variable de tipo Pr_Time utilizada para gestionar y actualizar los valores de temporización asociados al componente AttCtrlTimer.

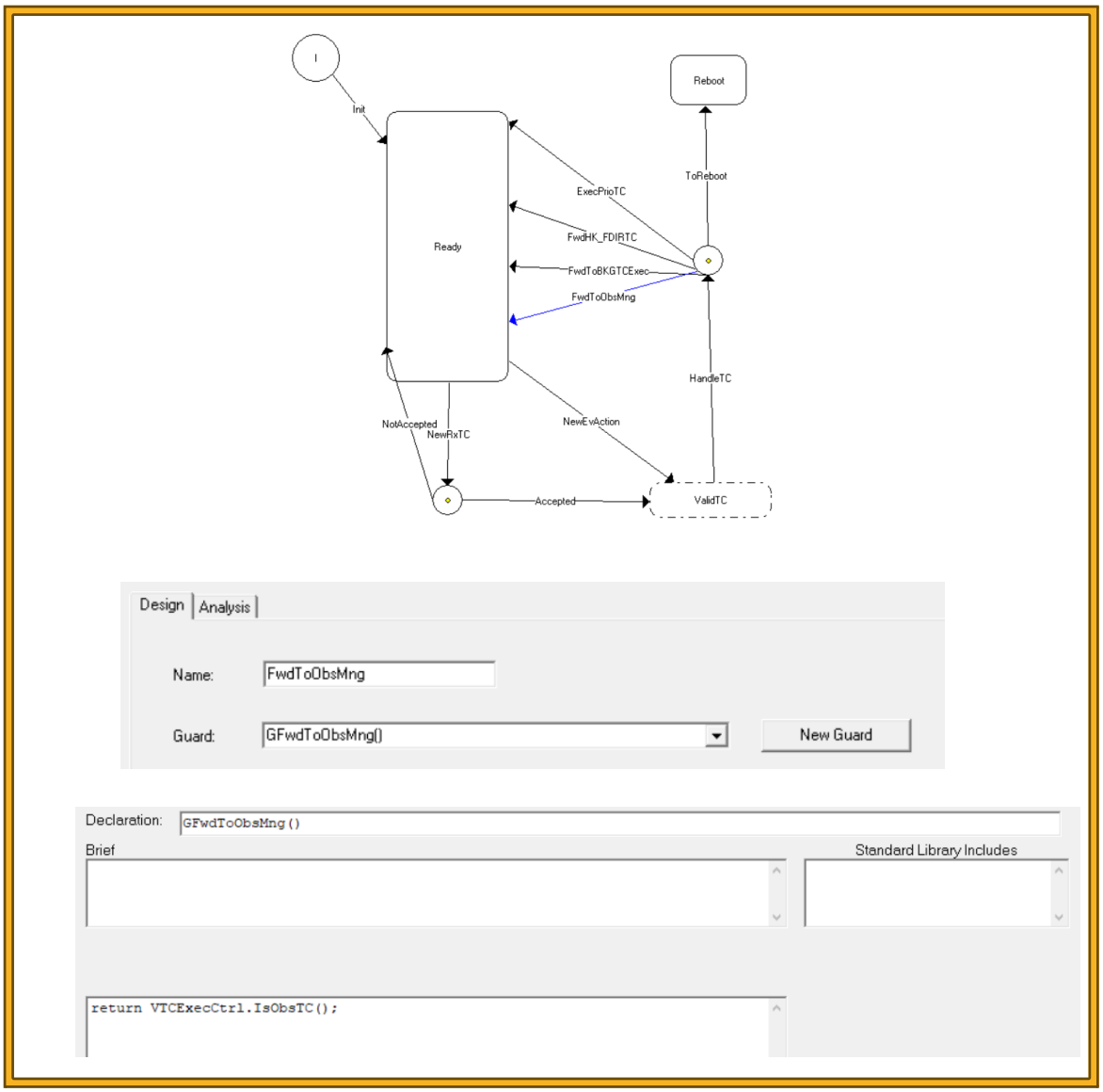


- **CImageInterval**: Constante de tipo Pr_Time con valor inicial 0.5, que establece el intervalo temporal empleado por el temporizador ObservTimer durante las fases de captura de imágenes.



4.3 *Triggers y Guardas (CCTCManager)*

- **FwdToObsMng:** Añadimos en la guarda una rama para enviar el telecomando al ObsMng En el caso de esta función, al no ser una transición sino una rama, no presenta ningún trigger.

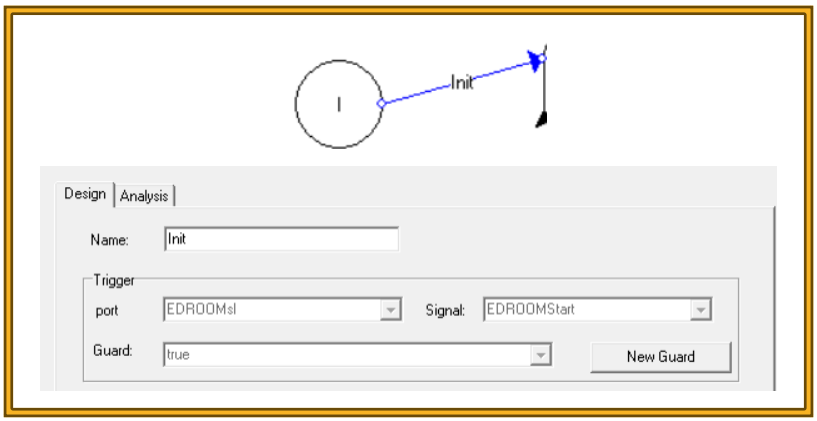


Para la implementación de la guarda se ha seguido el mismo procedimiento que en el resto de ramas.
Se ha elegido el nombre "IsObsTC" siguiendo el esquema del código eclipse.

```
//TODO DONE Add in EDROOM model the guard to handle ST[129] TCs
bool IsObsTC(){return (ExecCtrlObservTC==mExecCtrl);};
```

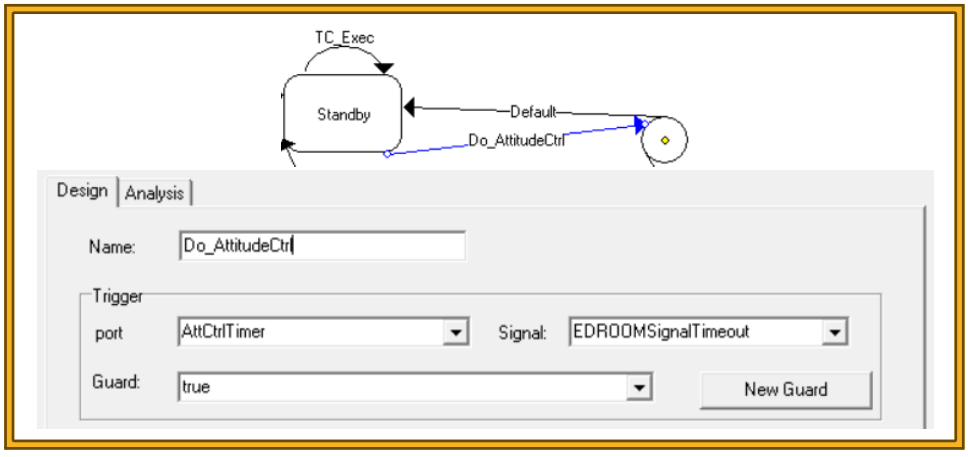
4.4 *Triggers y Guardas (CCObsMng)*

- **Init:** Al ser una transición no requiere de guarda porque no tiene ninguna condición que cumplir más allá del propio trigger.

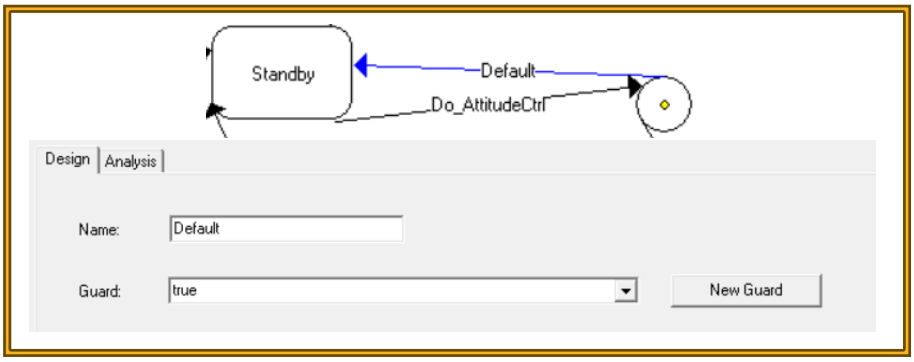


- **TC_Exec:** Al ser una transición no requiere de guarda porque no tiene ninguna condición que cumplir más allá del propio trigger.

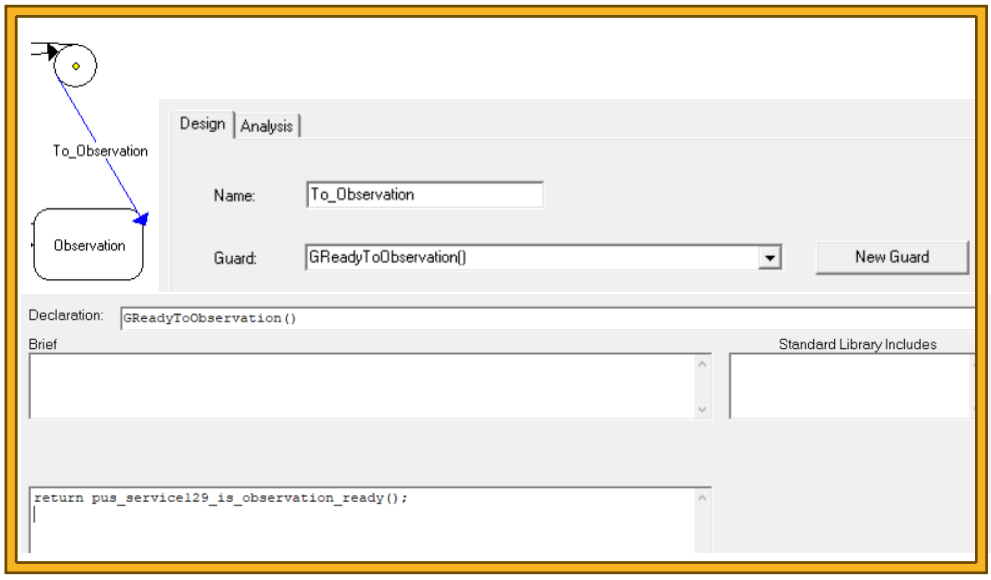
- **Do_AttitudeCtrl:** Al ser una transición no requiere de guarda porque no tiene ninguna condición que cumplir más allá del propio trigger.



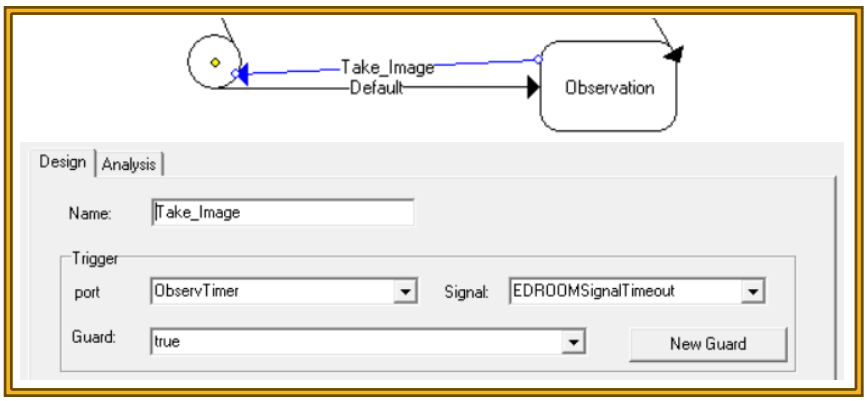
- **Default:** Al ser una transición no requiere de guarda porque no tiene ninguna condición que cumplir más allá del propio trigger.



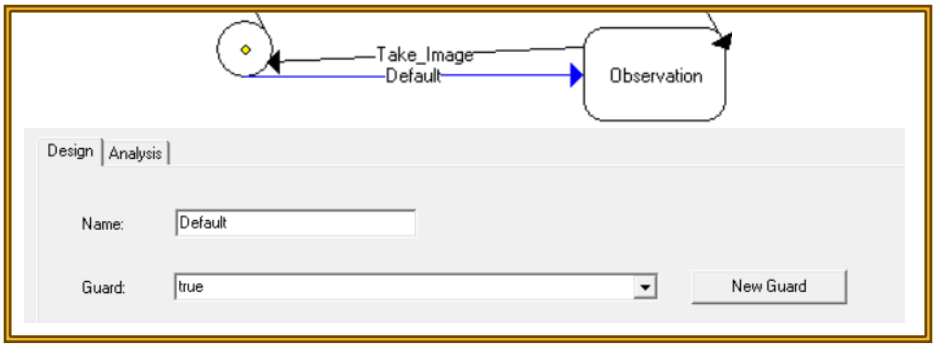
- **To_Observation:** Esta rama evalúa la guarda GReadyToObservation() que indica que se debe iniciar la observación programada. En caso de no cumplirse la condición de la guarda, se regresaría al estado Standby por medio del Default



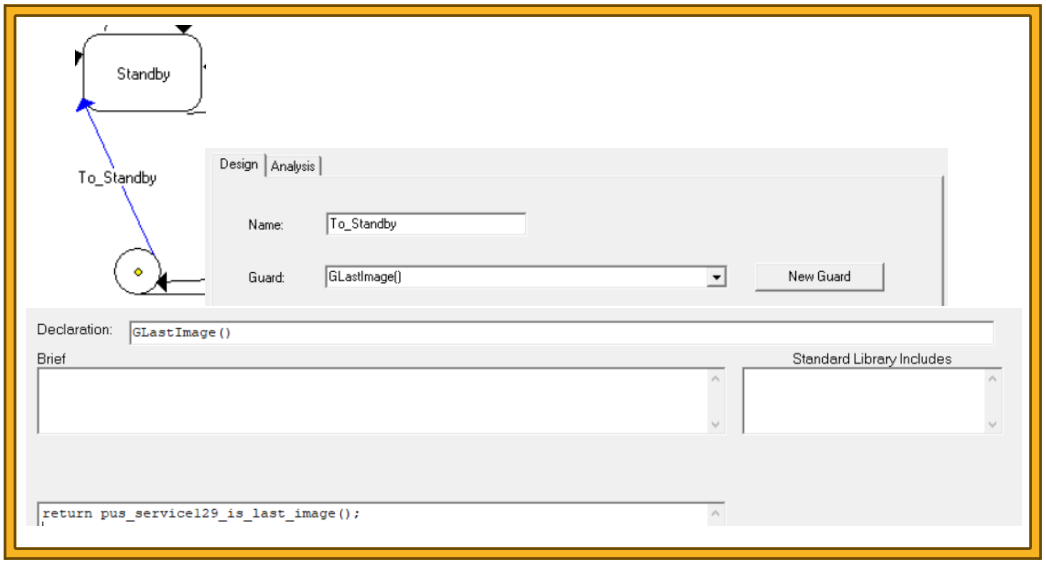
- **Take_Image:** Al ser una transición no requiere de guarda porque no tiene ninguna condición que cumplir más allá del propio trigger



- **Default:** Al ser una transición no requiere de guarda porque no tiene ninguna condición que cumplir más allá del propio trigger

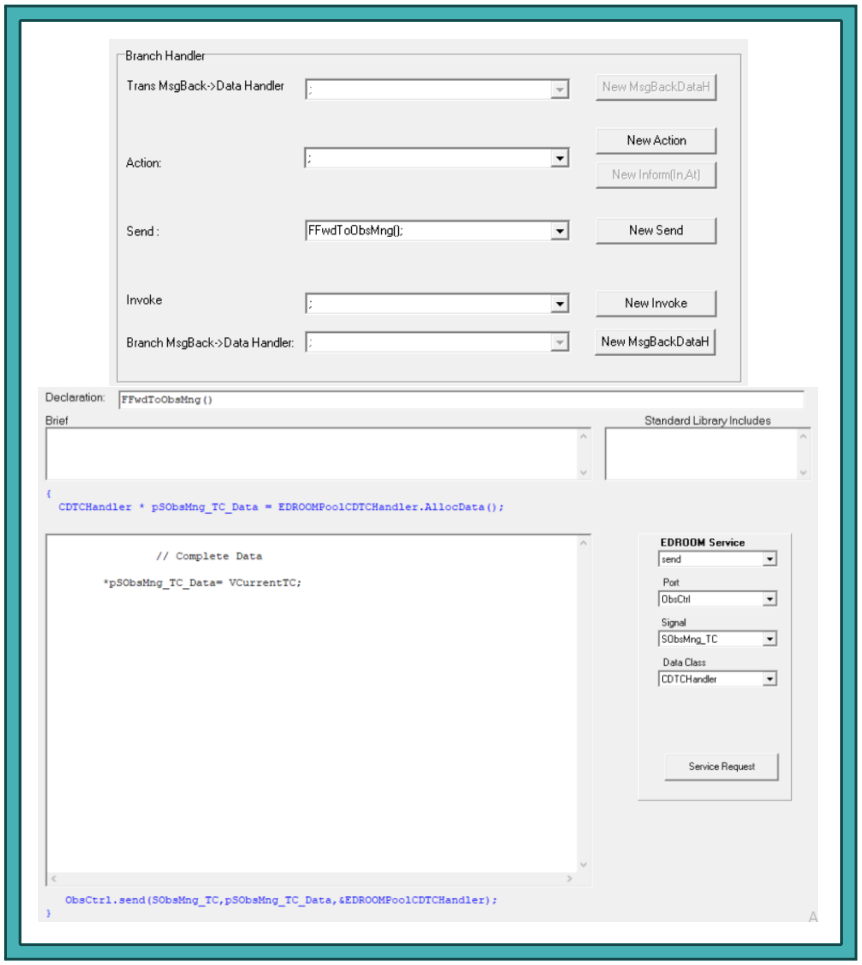


- **To_Standby:** Tras la toma de imágenes el componente analiza si es la última imagen de la observación mediante la guarda GLastImage(). Si la guarda GLastImage() retorna true, se regresa al estado Standby



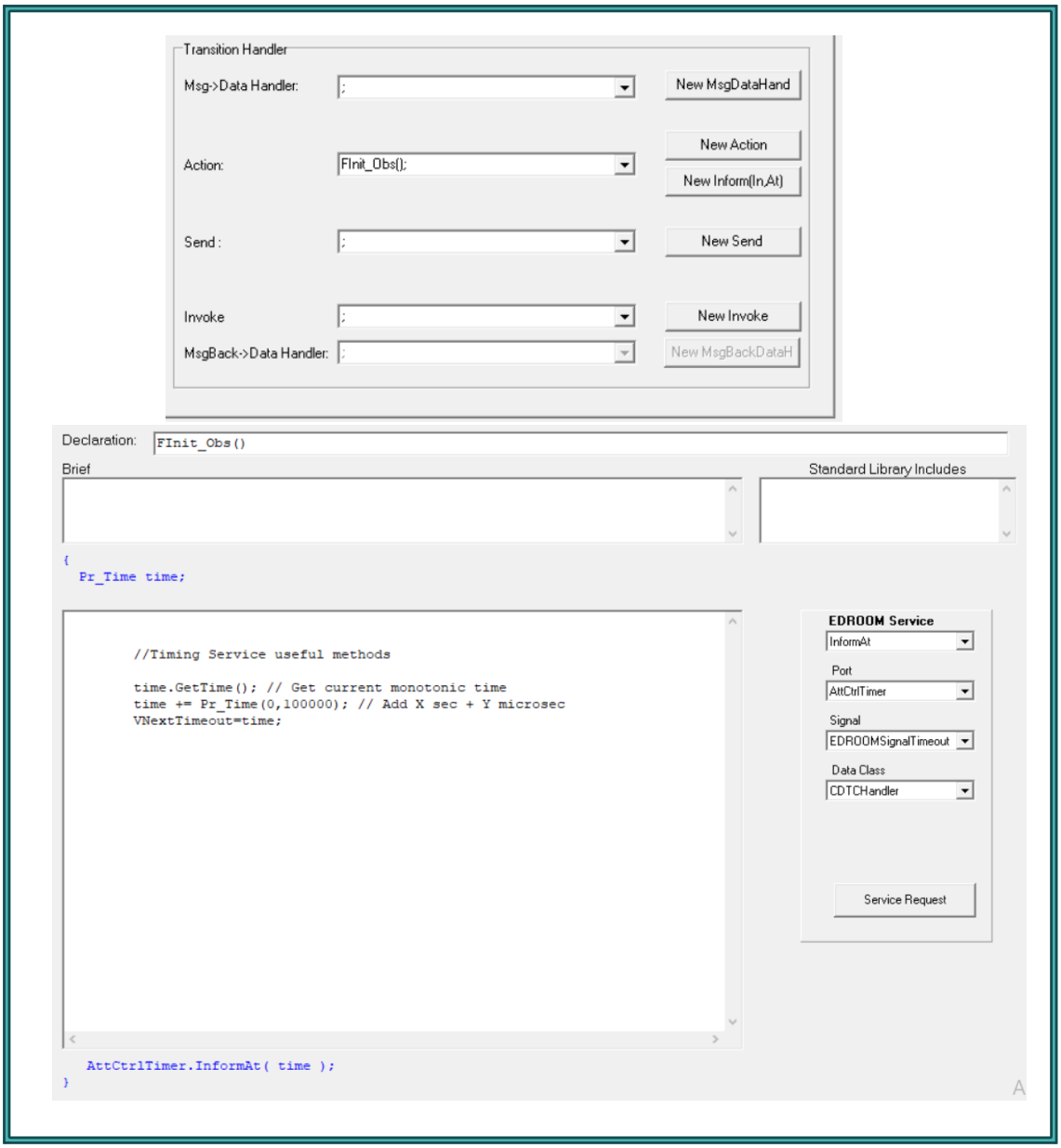
4.5 ***Actions (CCTCManager)***

- **FwdToObsMng → FFwdToObsMng (Send):** Confirma el envío del evento desde el TcManager al ObsMng.

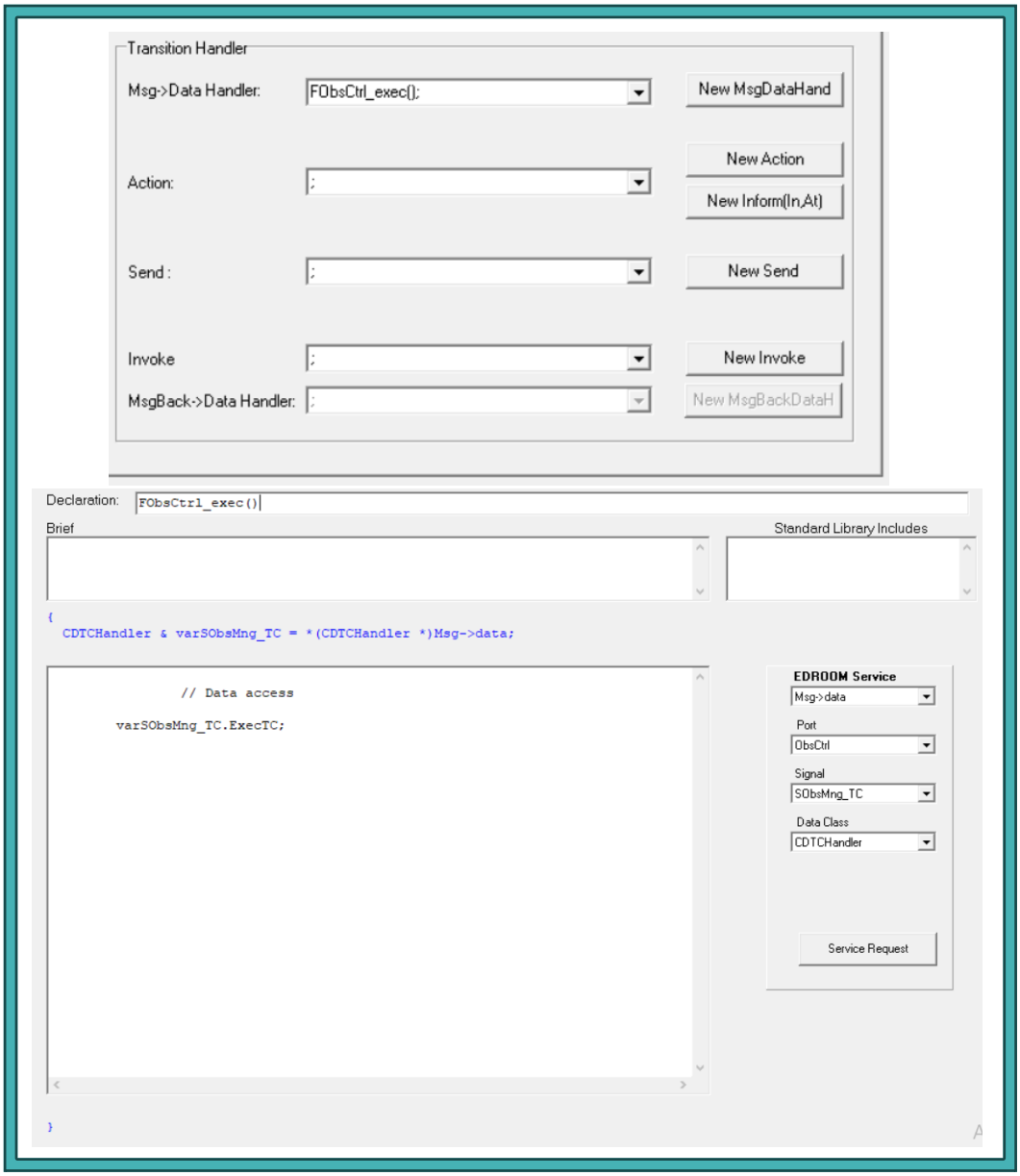


4.6 *Actions (CCObsMng)*

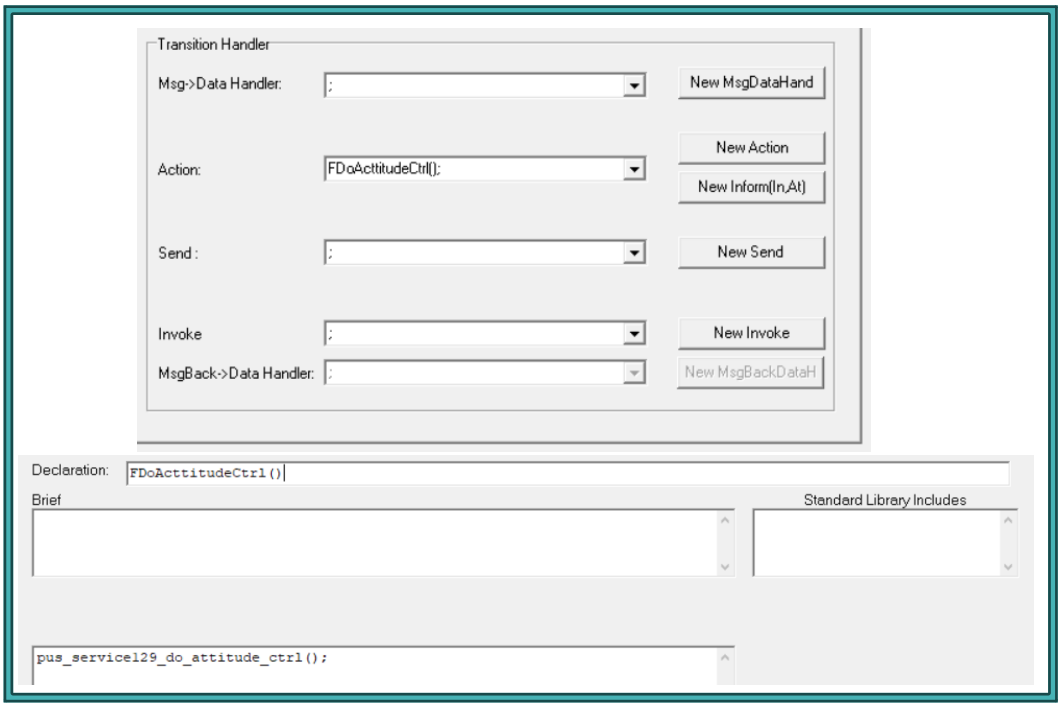
- **Init → FInit_Obs (Inform InAt):** Ejecuta en la inicialización del componente antes de alcanzar el estado Standby con un perido de ejecución de 100 ms



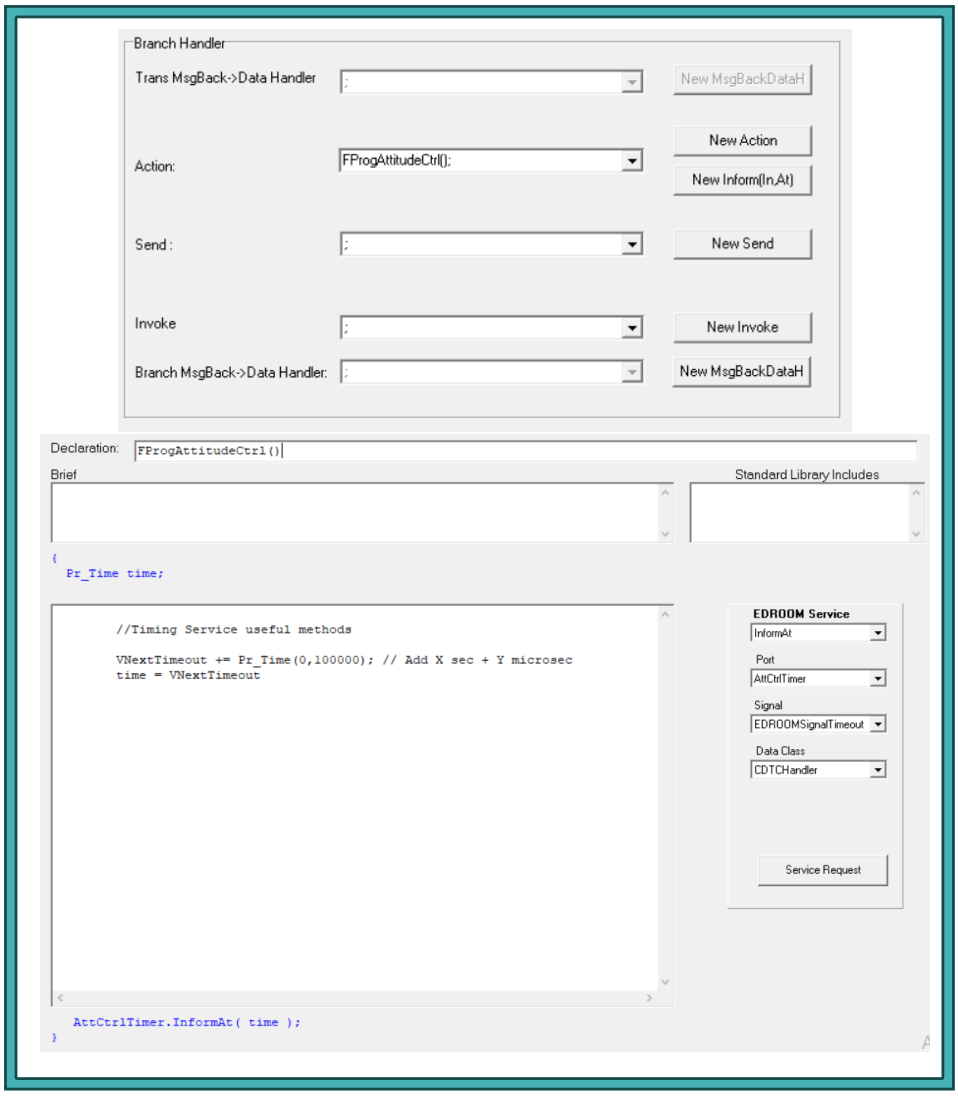
- **<u>TC_Exec → FObsCtrl_exec (MsgDataHandler):</u>** Procesa la ejecución de los telecomandos enviados desde el TCMnager



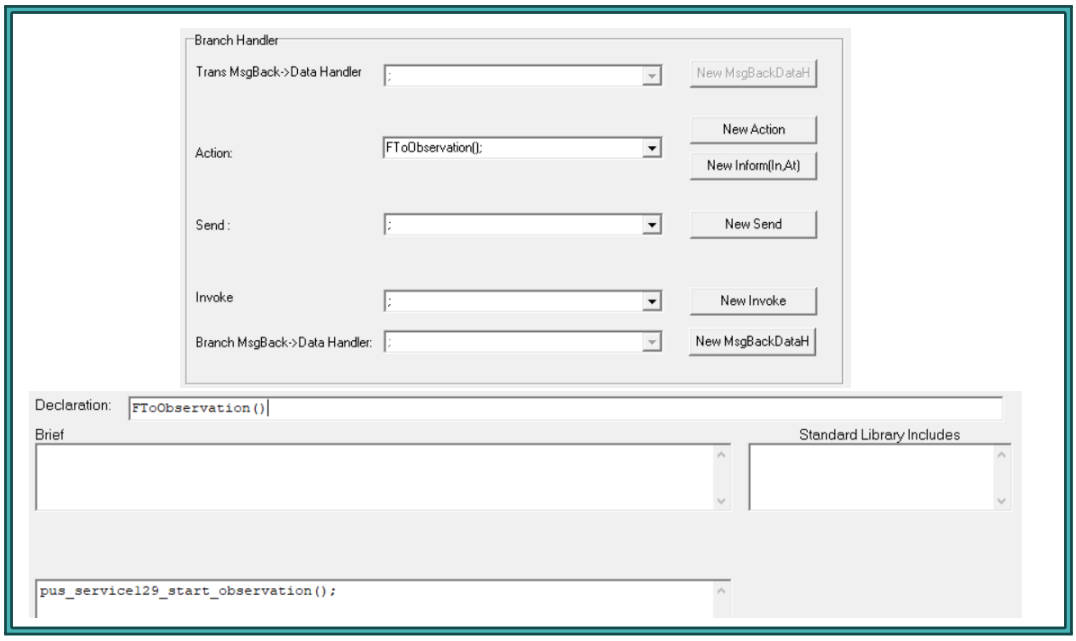
- **<u>Do_AttitudeCtrl → FDoActtitudeCtrl (Action):</u>** Por medio de la llamada al servicio 129, se aplica el algoritmo de control de actitud de acuerdo a la orientación objetivo y actualiza los parámetros del system data pool que gestiona este servicio



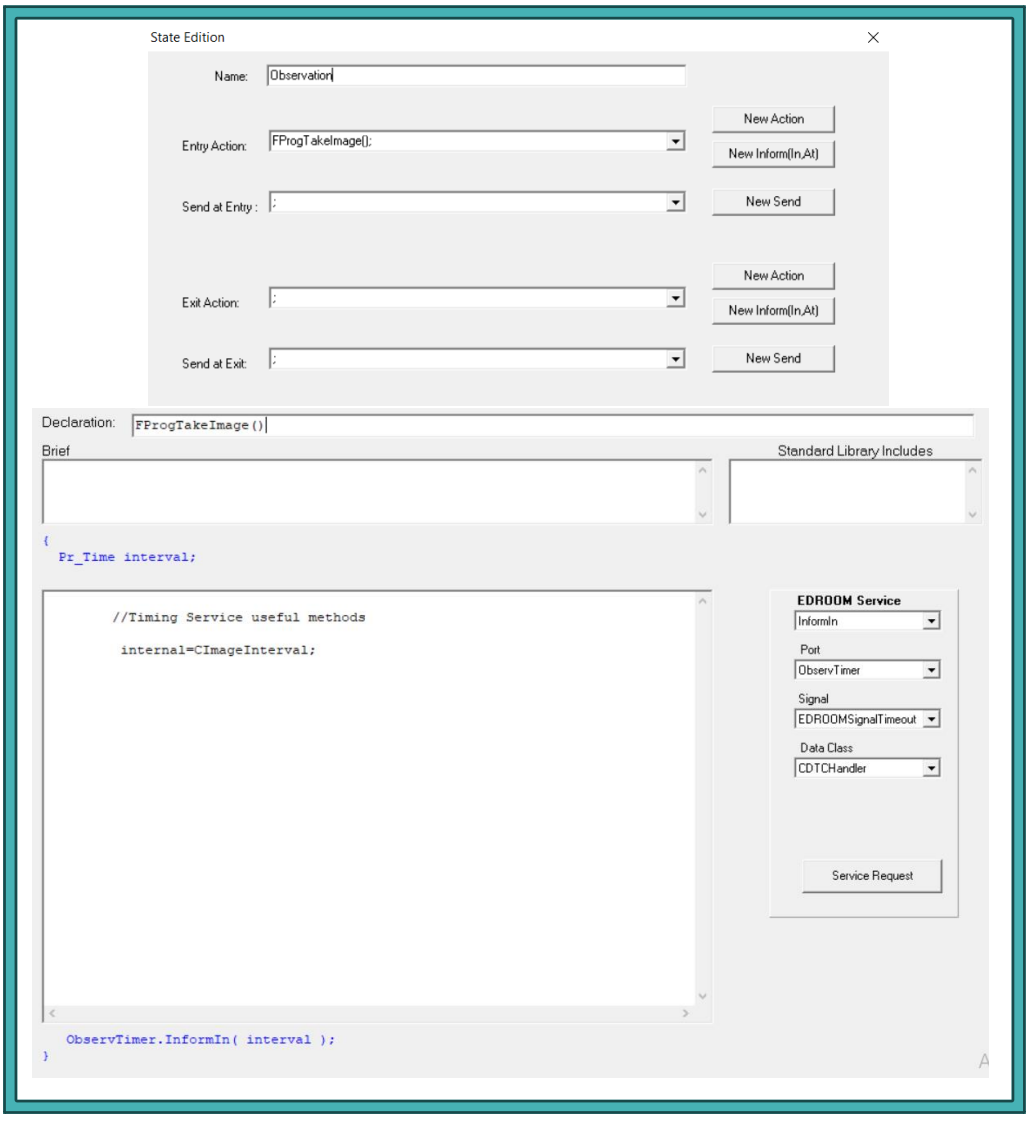
- **<u>Default → FProgAttitudeCtrl (Inform InAt):</u>** Debido al que el "timer" no se programa en el Standby este debe de ejecutarse en la propia transición



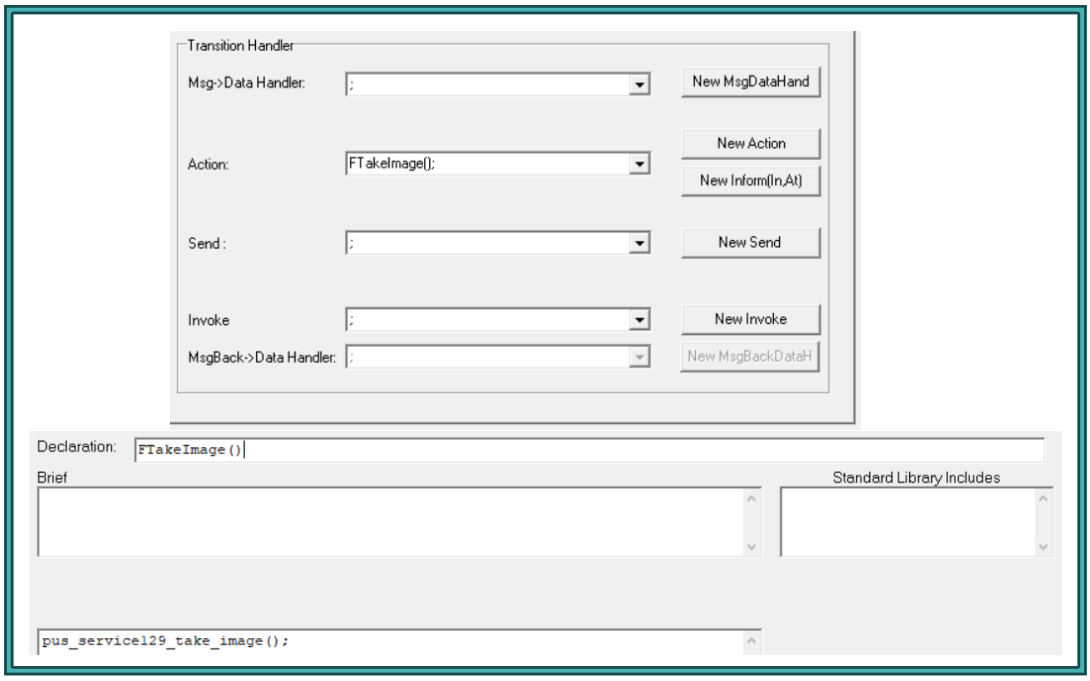
- **<u>To_Observation → FToObservation (Action):</u>** Por medio de la guarda GReadyToObservation se analiza si la observación programada está lista para realizarse, en caso afirmativo el evento que se dispara se gestionará con la ejecución de esta acción



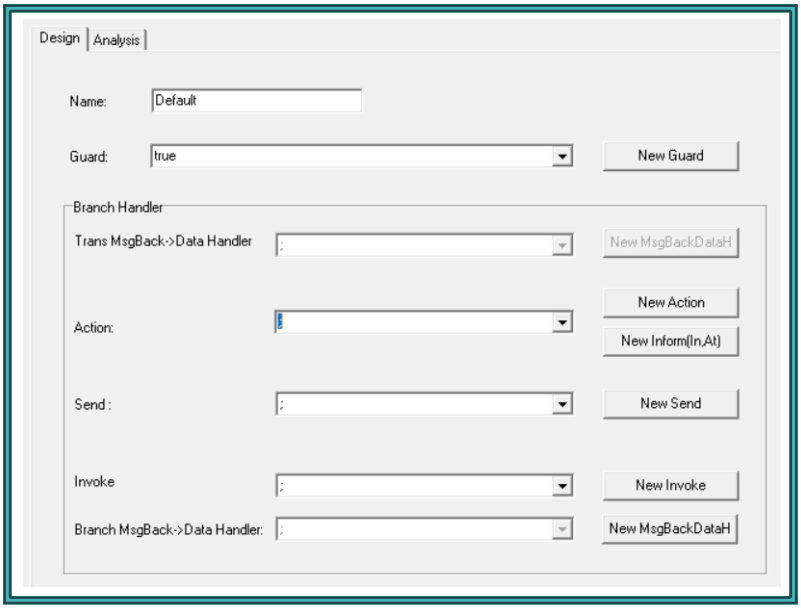
- **<u>Observation → FProgTakeImage (Inform InAt):</u>** Asociada a la constante CImageInterval, programa la toma de una imagen mediante el servicio InformIn. Cada vez que se pasa por el estado, este se actualiza el timer.



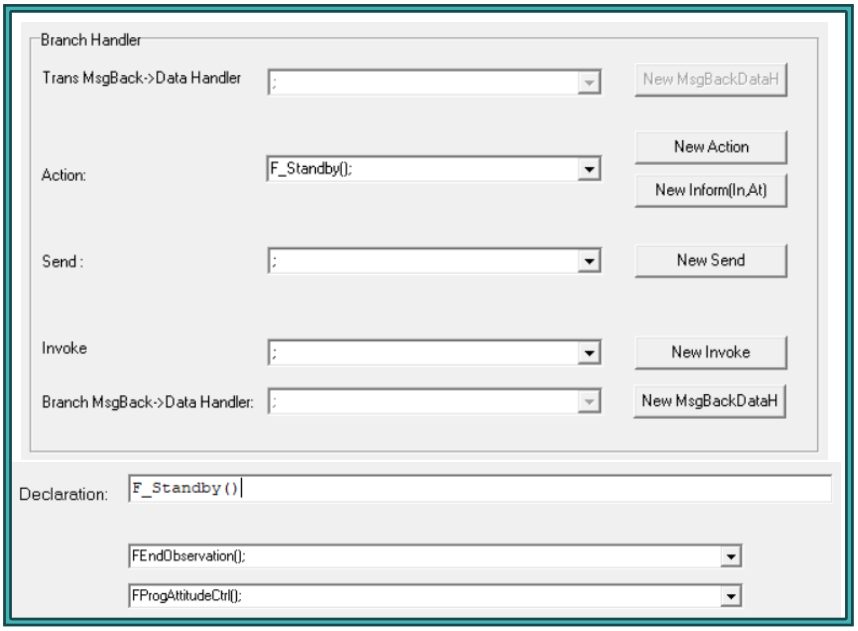
- **<u>Take_Image → FTakeImage (Action):</u>** Su ejecución permite la toma de imágenes, su código es una llamada a pus_service129_take_image() que genera la telemetría TM[129,4] con la imagen tomada.



- **<u>Default:</u>** Si no se han tomado todas las imágenes regresamos al estado Observation mediante otra transición default. En este caso, la transición no requiere de ninguna acción, pues es el propio estado Observation el encargado de actualizar y programar el timer cada vez que se pasa por él.



- **<u>To_Standby → F_Standby:</u>** Si se han tomado todas las imágenes, la vuelta al estado Standby se gestiona con la ejecución secuencial de dos acciones, la FEndObservation y la FProgAttitudeCtrl



Ambas acciones marcan el fin de la observación y son las encargadas de devolver el sistema al estado Standby y prepararlo así para una futura observación.