

## Queued Message Handler

La plantilla Queued Message Handler (QMH) facilita la ejecución en paralelo de múltiples secciones de código y el envío de datos entre ellas. Cada sección de código representa una tarea, como la adquisición de datos, y está diseñada de forma similar a una máquina de estados. Gracias a este diseño, Ud. puede dividir cada tarea en estados.

La plantilla QMH es una versión del patrón de diseño Productor/Consumidor, donde la interfaz de usuario (productor) produce mensajes y las tareas (consumidores) consumen los mensajes. Sin embargo, en la plantilla QMH, también Ud. puede producir mensajes desde un bucle consumidor.

Esta plantilla incluye un bucle productor y un bucle consumidor. Puede añadir bucles consumidores según sea necesario.

### Requerimientos de Sistema

LabVIEW Base, Full, o Professional Development System

### Casos de Uso

La plantilla QMH es útil para aplicaciones en las que se producen varias tareas en paralelo, a menudo, a diferentes velocidades. Por ejemplo, considere una aplicación que continuamente adquiere, registra y muestra dos señales: una señal RS-232 y una señal analógica. Estas señales se producen a velocidades diferentes, por lo que la aplicación debe tener dos bucles que se ejecuten en paralelo. Además, cada bucle se divide en los siguientes estados:

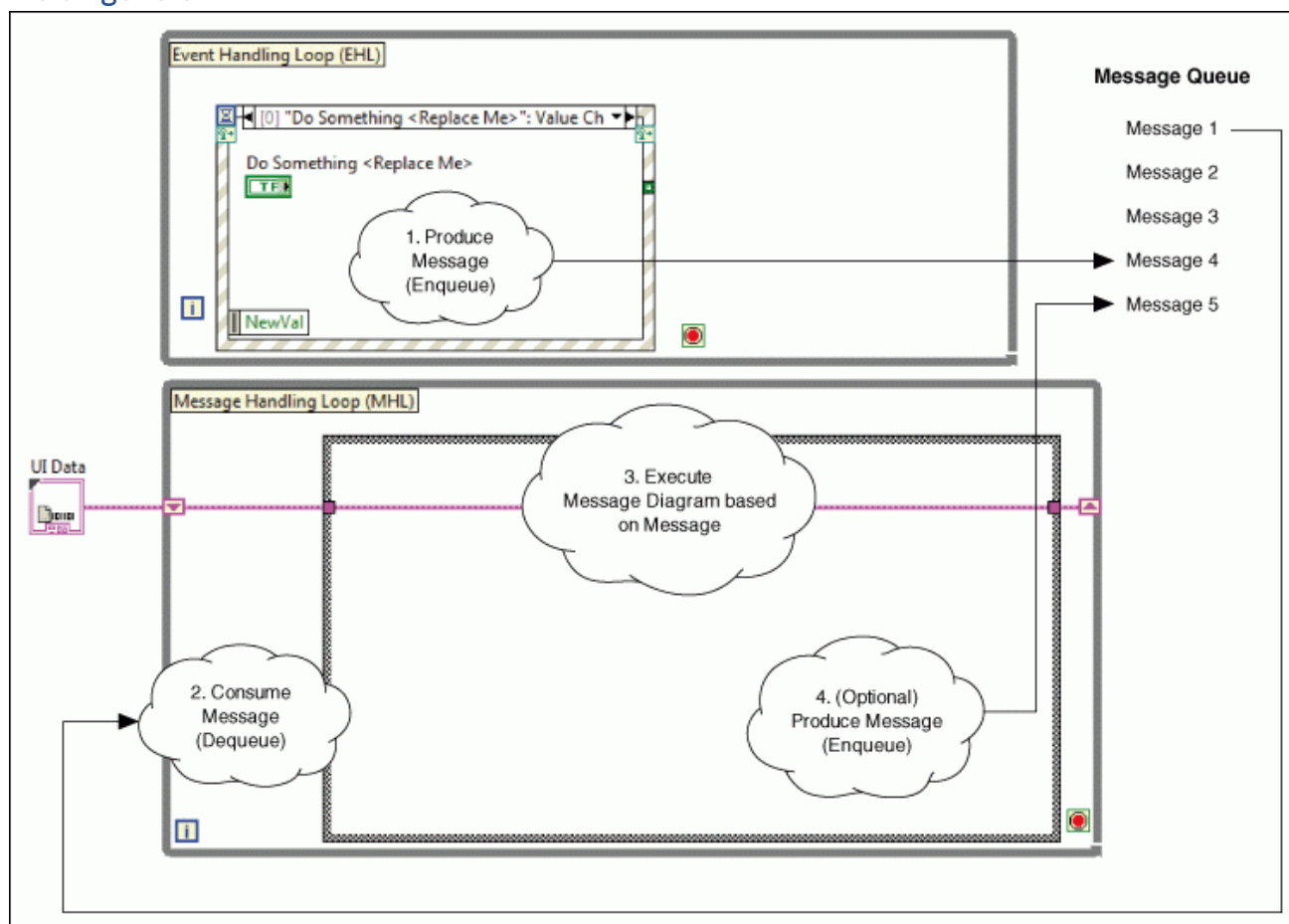
- Inicializar (Initialize) el hardware de adquisición de datos
- Adquirir (Acquire) datos
- Registrar (Log) en disco los datos adquiridos
- Visualizar (Display) los datos adquiridos en un gráfico de forma de onda
- Poner (Set) el hardware en estado seguro
- Detener (Stop) la adquisición de datos y apagar (Shutdown) el hardware

La aplicación requiere una interfaz de usuario con capacidad de respuesta; es decir, los usuarios deben poder pulsar *botones* incluso mientras la aplicación está ejecutando otro comando. Por lo tanto, la aplicación requiere un tercer bucle paralelo que monitorice continuamente el panel frontal en busca de eventos, como los siguientes comandos:

- Iniciar (Start) la adquisición RS-232
- Detener (Stop) la adquisición RS-232
- Habilitar el registro (Logging) RS-232
- Deshabilitar el registro (Logging) RS-232
- Iniciar la adquisición (Start) de datos analógicos
- Detener la adquisición (Stop) de datos analógicos
- Habilitar el registro (Enable) de datos analógicos
- Deshabilitar el registro (Disable) de datos analógicos

La plantilla QMH ofrece un punto de partida para escribir este tipo de aplicación.

## Visión general



Esta plantilla ejecuta repetidamente los siguientes pasos:

1. Un usuario interactúa con el panel frontal, causando que la estructura Event en el **Event Handling Loop** (EHL) produzca un mensaje. LabVIEW almacena el mensaje en una cola.
2. El **Message Handling Loop** (MHL) lee un mensaje de la cola de mensajes y lo remueve.
3. El mensaje es una cadena que coincide con uno de los sub diagramas de la estructura Case en el MHL. Por lo tanto, la lectura del mensaje hace que se ejecute el sub diagrama correspondiente de la estructura Case. Este sub diagrama se llama *diagrama de mensaje* porque corresponde a un mensaje.
4. Opcionalmente, el *diagrama de mensaje* produce otro mensaje, almacenándolo en la cola de mensajes.

Notas:

- El EHL es el bucle productor. El MHL es el bucle consumidor. Estos bucles se ejecutan en paralelo y están conectados por la cola de mensajes, que facilita la comunicación entre ellos.
- La cola de mensajes es una cola de LabVIEW que almacena mensajes para su consumo. Como el EHL envía mensajes a esta cola y no directamente al MHL, el EHL puede producir mensajes mientras el MHL no los está consumiendo. Cada cola de mensajes pertenece a un único MHL.

- Cada iteración del MHL lee el mensaje más antiguo de la cola de mensajes y, a continuación, ejecuta el diagrama de mensajes correspondiente. Aunque el MHL consume principalmente mensajes, también puede producirlos.
- Cada bucle gestiona los errores mediante un subVI gestor de errores específico del bucle. Si el QMH encuentra un error en el EHL, LabVIEW muestra un mensaje de error sin detener el QMH. Si el QMH encuentra un error en el MHL, LabVIEW muestra un mensaje de error y detiene el QMH.
- Su aplicación puede tener múltiples MHL. Cada MHL corresponde a una tarea que la aplicación realiza, como adquirir o registrar datos.
- Observe el cluster **UI Data** en el diagrama anterior. Este cluster contiene datos a los que cada diagrama de mensajes en un MHL puede acceder y modificar. En esta plantilla, el cluster se define como un typedef, UI Data.cti. Cada typedef pertenece a un único MHL.

### Ejecute esta plantilla

1. En la ventana del **Explorador de proyectos**, abra y ejecute Main.vi.
2. Haga clic en los controles del panel frontal y observe cómo el indicador **Display** muestra mensajes.

### Modificación de esta plantilla

#### Determinar sus necesidades

La siguiente tabla resume las decisiones de diseño que debe tomar al modificar esta plantilla:

Decisiones de Diseño	Ejemplo	Información detallada
Debe determinar cuántos MHL añadir. Cada MHL define una tarea que se ejecuta en paralelo con otras tareas.	Usted tiene una aplicación que adquiere datos y, en paralelo, registra estos datos en el disco.  Esta aplicación consta de dos tareas: adquisición de datos y registro de datos. Por lo tanto, necesita dos Message Handling Loops.	<a href="#">Creación de un Message Handling Loop</a>
Para cada MHL, es necesario determinar qué diagramas de mensajes añadir.  Un diagrama de mensaje es un sub diagrama de la estructura Case en un MHL. Cada sub diagrama corresponde a un estado en el que puede estar la tarea; por lo tanto, para determinar los diagramas de mensajes que hay que añadir, separar cada tarea en estados.	Desea separar la tarea de adquisición de datos en tres estados: Initalize, Acquire Data y Exit. Por lo tanto, cree estos diagramas de mensajes en el MHL que adquiere datos.  Usted desea separar la tarea de registro de datos en tres estados: Initialize, Log y Close. Por lo tanto, cree estos diagramas de mensajes en el MHL que registra datos.	<a href="#">Creando un Diagrama de Mensaje</a>
Debe determinar qué datos necesitan los diagramas de mensajes de un MHL.	Cada diagrama de mensajes del MHL de adquisición de datos necesita acceder a una referencia de hardware. El diagrama de mensajes Initialize necesita abrir esta referencia, el diagrama Acquire Data utiliza esta	<a href="#">Definición de los datos que necesita un Message Handling Loop</a>

	referencia para adquirir datos y el diagrama de mensajes Exit cierra la referencia.	
Es necesario determinar cuándo ejecutar cada diagrama de mensajes. Un diagrama de mensajes se ejecuta después de que su MHL reciba el mensaje correspondiente. Por lo tanto, es necesario determinar cuándo enviar cada mensaje al MHL. Puede enviar un mensaje desde un control del panel frontal o desde un diagrama de mensajes.	Desea añadir un botón que envíe el mensaje Inicializar al MHL de adquisición de datos.  A continuación, desea que el diagrama de mensajes Initalize envíe el mensaje Acquire Data al mismo MHL.	<a href="#">Adición de un control que envía un mensaje a un Message Handling Loop</a> o <a href="#">Envío de un mensaje a un Message Handling Loop</a> dependiendo de si desea que un control envíe el mensaje.
Debe determinar si desea que el mensaje Exit detenga cada MHL. El Dequeue Message VI utiliza este mensaje porque es capaz de apagar un MHL.	Desea que cada MHL se apague cuando reciba el mensaje Stop en lugar del mensaje Exit.	<a href="#">Cambiano el Mensaje que Detiene un Message Handling Loop</a>
Debe determinar si desea ignorar algún error específico dentro del EHL o del MHL.	Cuando se leen mensajes de la cola de mensajes, se desea ignorar los errores de timeout de la red.	<a href="#">Ignorando Errores en el Bucle de Manejo de Eventos y en el Bucle de Manejo de Mensajes</a>

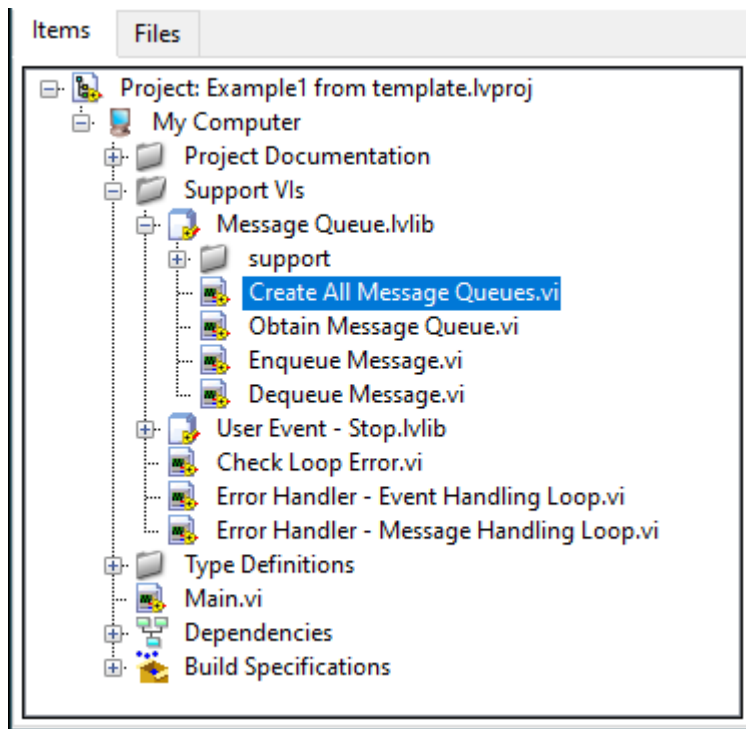
### Creación de un Message Handling Loop

Un Message Handling Loop (MHL) representa una (sola) tarea que la aplicación puede realizar, como la adquisición de datos o el registro de datos, en paralelo con otras tareas. Cada MHL puede dividirse en subtareas que corresponden a estados. Los MHL constan de los siguientes componentes:

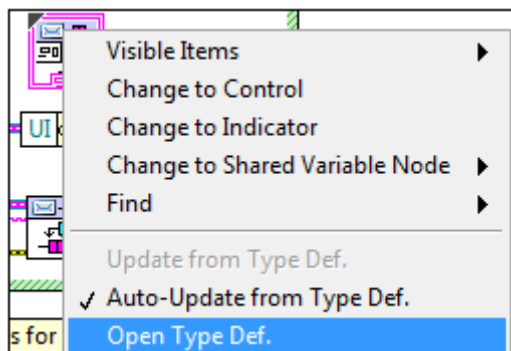
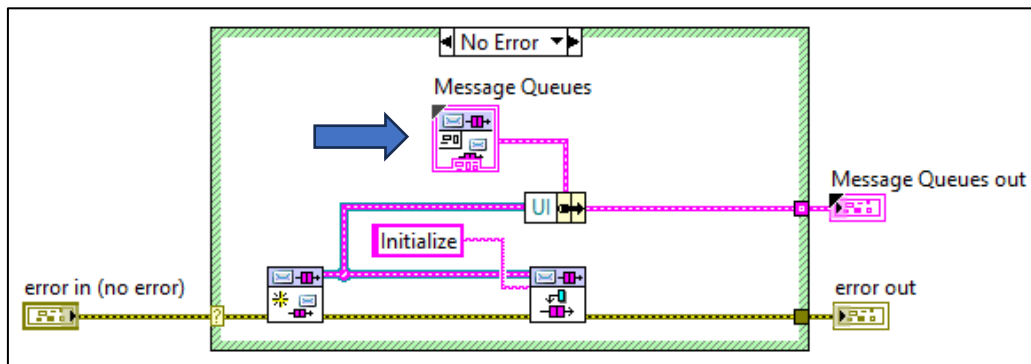
- Una cola de mensajes
- Un bucle While que lee mensajes de la cola de mensajes
- Una estructura Case que contiene un sub diagrama, también conocido como *diagrama de mensajes*, para cada mensaje posible que se puede leer, donde cada mensaje corresponde a un estado de la tarea
- (Opcional) Datos a los que puede acceder cada diagrama de mensajes del MHL

Complete los siguientes pasos para añadir un MHL:

1. Esta plantilla contiene un typedef que define el cluster que contiene los refnums para todas las colas de mensajes. Por defecto, este typedef sólo tiene espacio para una cola. Complete los siguientes pasos para añadir una segunda cola a este typedef:
  - a. En la ventana **Project Explorer**, abra Message Queue.lvlib:Create All Message Queues.vi.

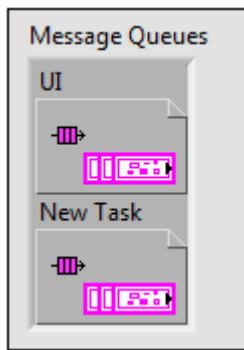


- b. Haga clic con el botón derecho en la constante **All Message Queues** y abra el typedef:



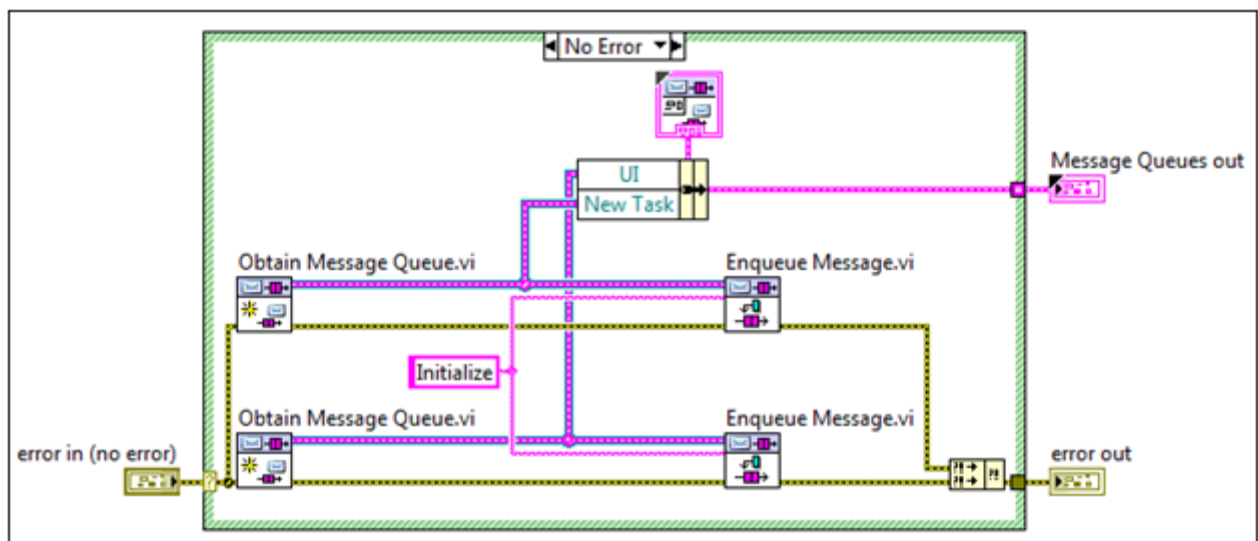
LabVIEW lanza la ventana del **Control Editor**.

- c. Expanda el borde del cluster **Message Queues**.
- d. Duplique el refnum de la cola **UI** en el cluster.
- e. Nombre la nueva cola refnum. Por ejemplo: New Task

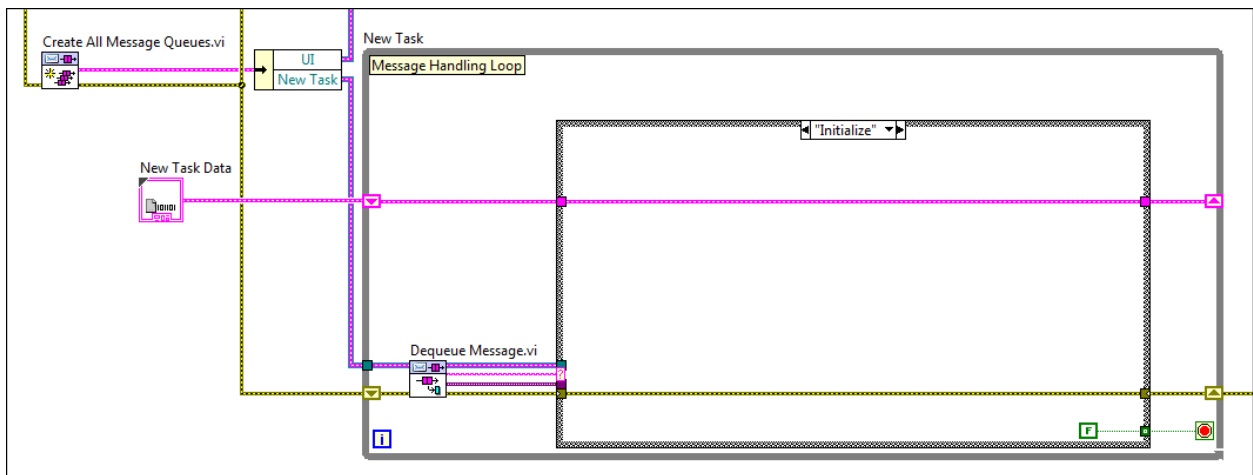


- f. Seleccione **File » Apply Changes** y cierre la ventana del **Control Editor**. El typedef de colas de mensajes contiene ahora una cola de mensajes adicional.
2. Modifique Create All Message Queues.vi para ejecutar los siguientes pasos:
- Obtener la referencia de la cola de mensajes
  - Agrupar (Bundle) esta cola en el clúster **Message Queues out**
  - (Opcional) Enviar un mensaje inicial al nuevo MHL

La siguiente captura de pantalla muestra un ejemplo de código que realiza los pasos anteriores:

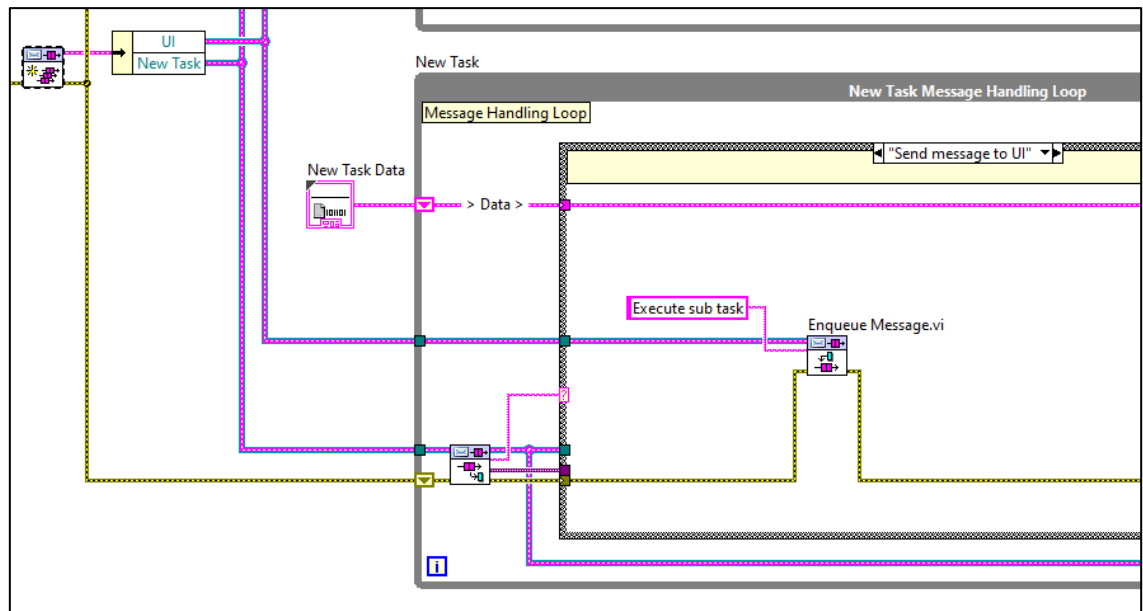


3. (Opcional) Si el MHL necesita acceso a datos, cree un typedef que represente estos datos.
4. En Main.vi, cree el Message Handling Loop que representa la tarea:




#### Notas:

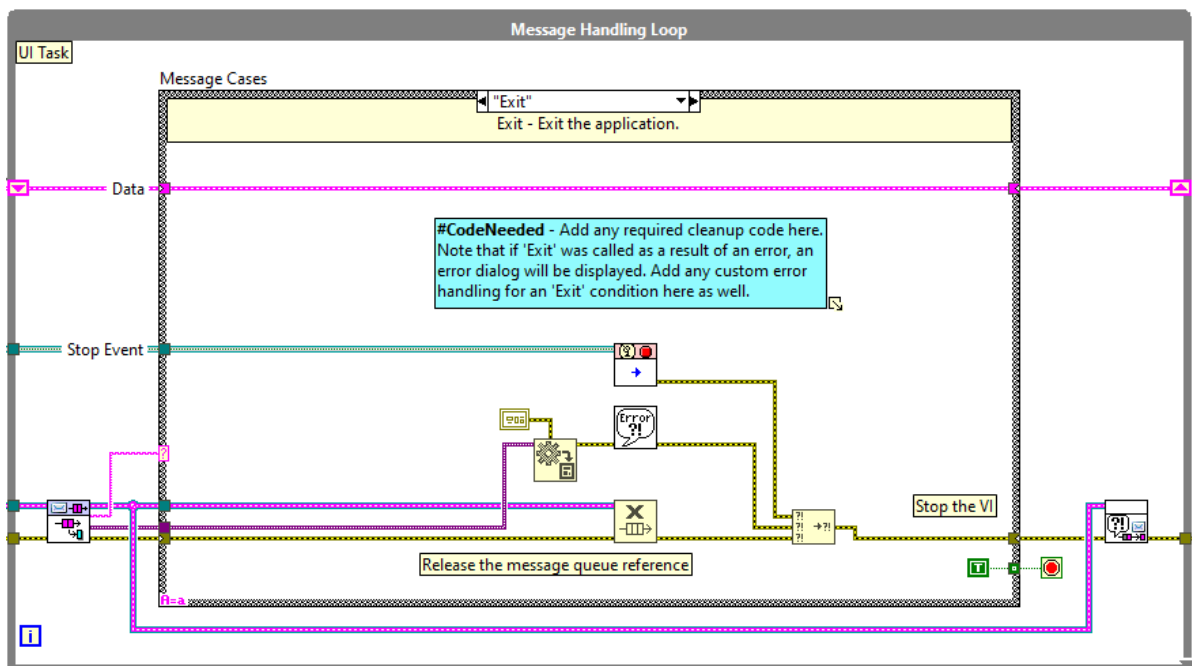
- Recuerde que en el paso 2, usted agrupó (bundled) el refnum de la cola **New Task** en el cluster **Message Queues out**. El código anterior muestra dónde se desagrega (unbundle) este cluster y cómo la rama de cable para la cola de New Task va al MHL de New Task.
- Recuerde que en el paso 2, tenía la opción de enviar un mensaje inicial a la cola de mensajes. El código del paso 2 muestra el mensaje inicial Initialize. El código anterior muestra el diagrama de mensajes (Initialize) que se ejecuta cuando se recibe este mensaje.
- Recuerde que en el paso 4, tenía la opción de crear un typedef. El código anterior muestra cómo cablear este typedef, **New Task Data**, para que el MHL pueda utilizarlo.
- Si desea que la nueva tarea envíe mensajes a la cola de la **UI**, bifurque el cable para el refnum de la cola de la UI en el bucle **New Task**.



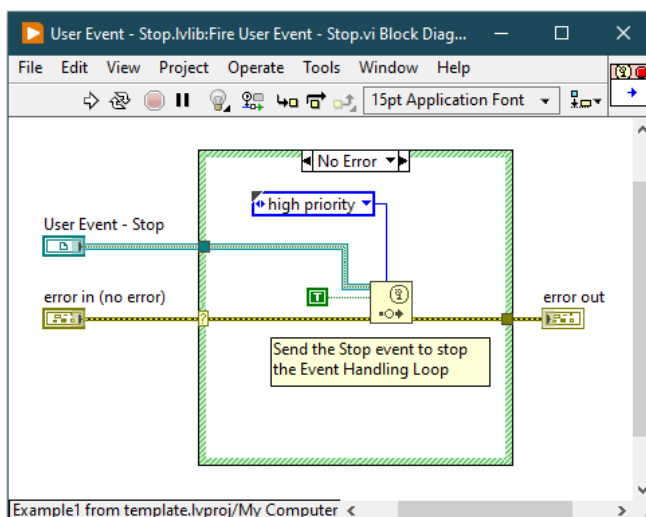
- El código anterior muestra la constante FALSE conectada al terminal condicional del bucle While. En cada MHL, sólo un diagrama de mensajes

debe ser capaz de detener el MHL. Este diseño previene paradas accidentales y parciales garantizando las siguientes condiciones:

- El código de apagado sólo se ejecuta justo antes de que se apague el MHL.
  - El código de apagado siempre se ejecuta hasta el final.
- f. Para mantener el diagrama de bloques de Main.vi compacto y legible, Ud. puede encapsular cada MHL en un subVI. Para organizar aún más el proyecto, Ud. puede poner cada subVI, cualquier VI de apoyo, y su typedef de datos en una biblioteca de proyecto. Consulte el proyecto de ejemplo Continuous Measurement and Logging, disponible en el cuadro de diálogo **Create Project**, para ver un ejemplo de este diseño.
- g. El MHL mostrado arriba no necesita acceso al cable Stop Event. El MHL en la plantilla usa este cable para ejecutar el Fire User Event - Stop VI , que cierra el Event Handling Loop. Ningún otro MHL necesita hacer esto.



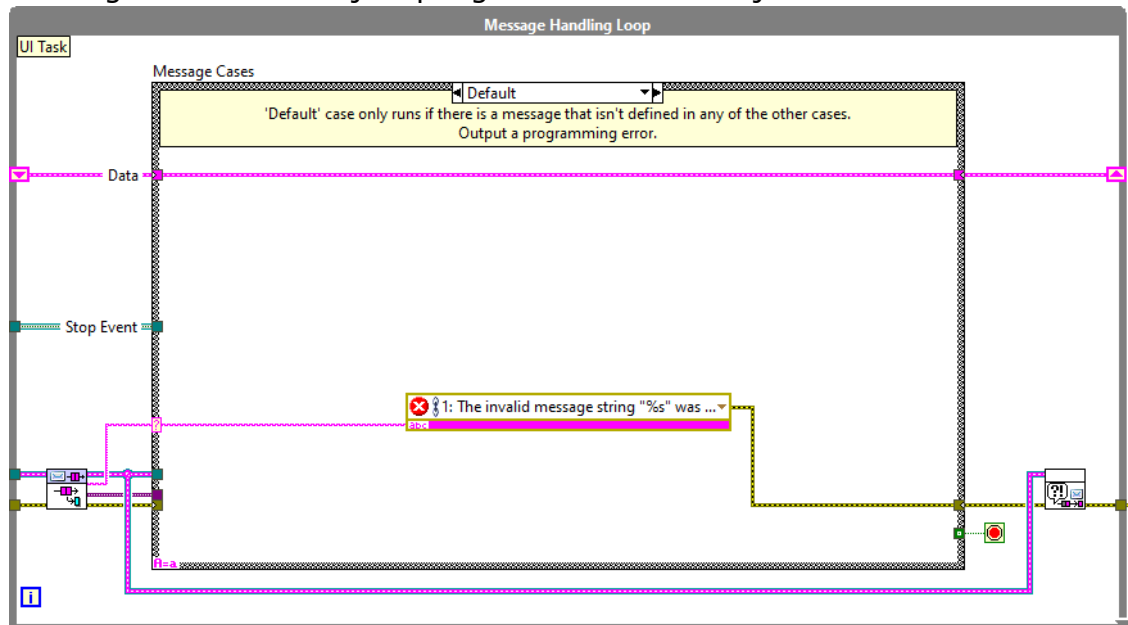
- h. Agregar a la función Generate User Event VI dentro del Fire User Event - Stop VI, el enum high priority.



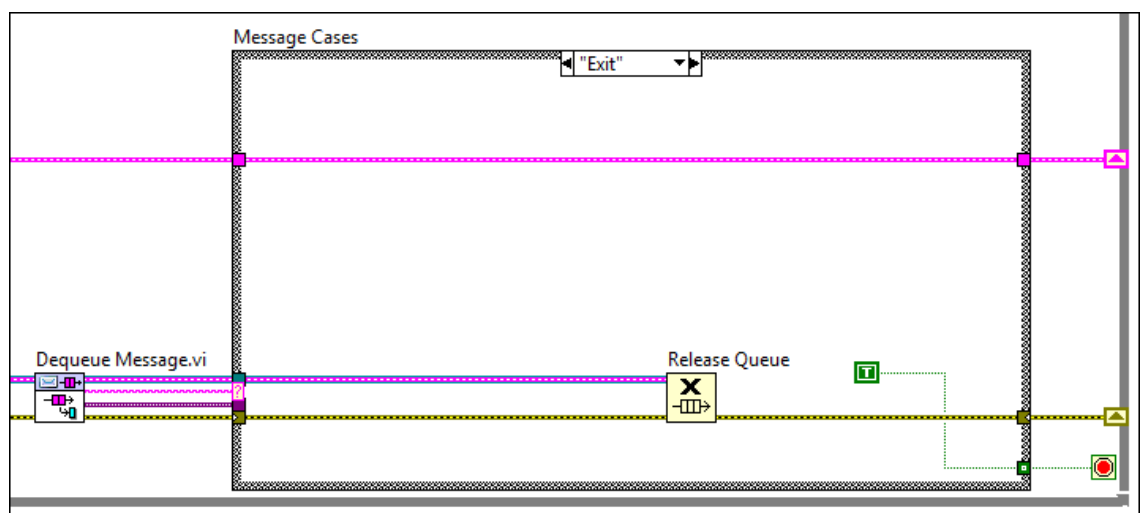


5. Añada diagramas de mensajes a la estructura Case en el MHL. Para minimizar errores y comportamientos inesperados, asegúrese de que cada MHL tenga los siguientes diagramas de mensajes:

- Un diagrama de mensajes que inicializa la tarea; por ejemplo, este diagrama podría conectarse a un dispositivo de hardware, abrir archivos para el registro de datos, etc.
- Un diagrama de mensajes que gestione los mensajes no reconocidos.



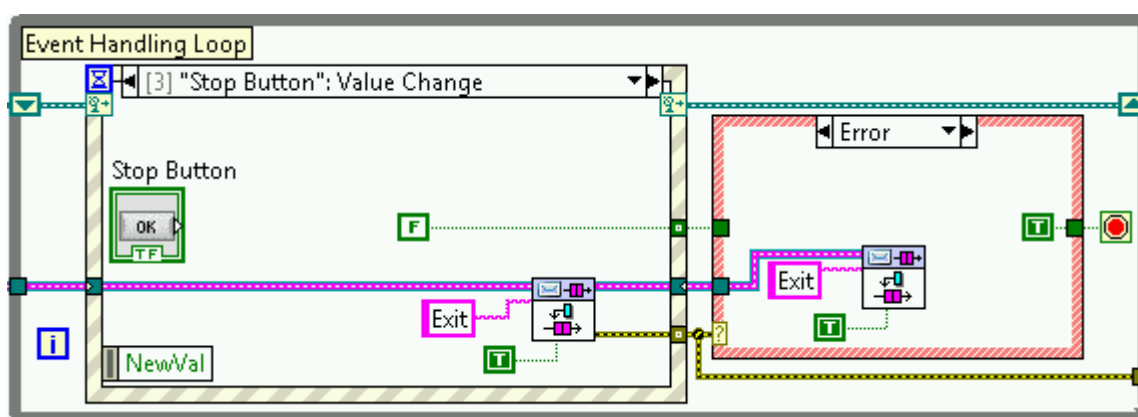
- Un diagrama de mensajes que, cuando se ejecute, libere la cola de mensajes y detenga el bucle. Por ejemplo:



Por defecto, el mensaje que dispara este diagrama de mensajes es Exit, pero puede cambiarlo.

- (Opcional) Si la aplicación requiere que el MHL deje de realizar su tarea, pero permanezca activo (para potencialmente reiniciar la tarea), cree un diagrama de mensajes que utilice la función Flush Queue para eliminar cualquier mensaje pendiente.

6. Agregue código al EHL que ordene al nuevo MHL detenerse en caso de error o cuando la aplicación se detenga; es decir, ejecute el diagrama de mensajes que creó en el paso 5c. Añada este código a la Event Structure y al caso de Error que se muestra en la siguiente figura:

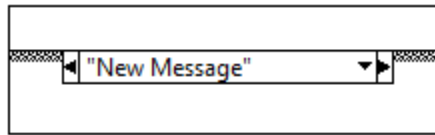


## Creando un Diagrama de Mensaje

Un *diagrama de mensaje* es un sub diagrama de una estructura Case que es etiquetado con una cadena. Se ejecuta cuando el MHL recibe un mensaje que coincide con esta etiqueta.

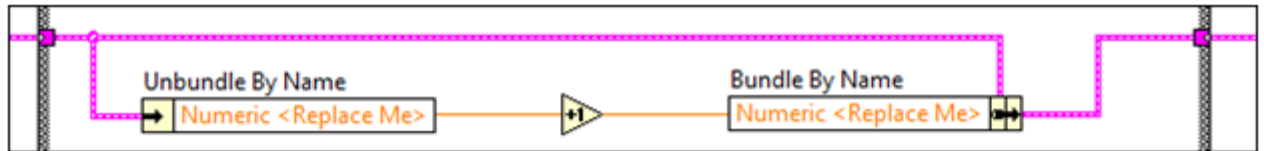
1. En el MHL que representa la tarea, agregue un sub diagrama a la estructura Case.

- Página 10 | 16

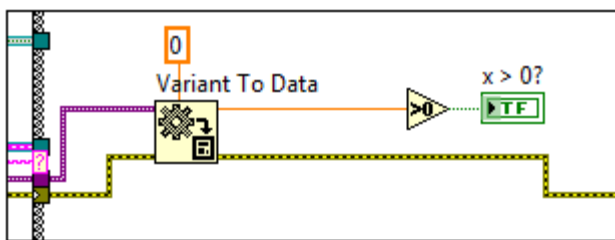


3. Añada código que se ejecute al recibir este mensaje. Al hacerlo, preste atención a las siguientes directrices:

- Para acceder a los datos de la tarea y modificarlos, utilice las funciones Unbundle By Name y Bundle By Name:



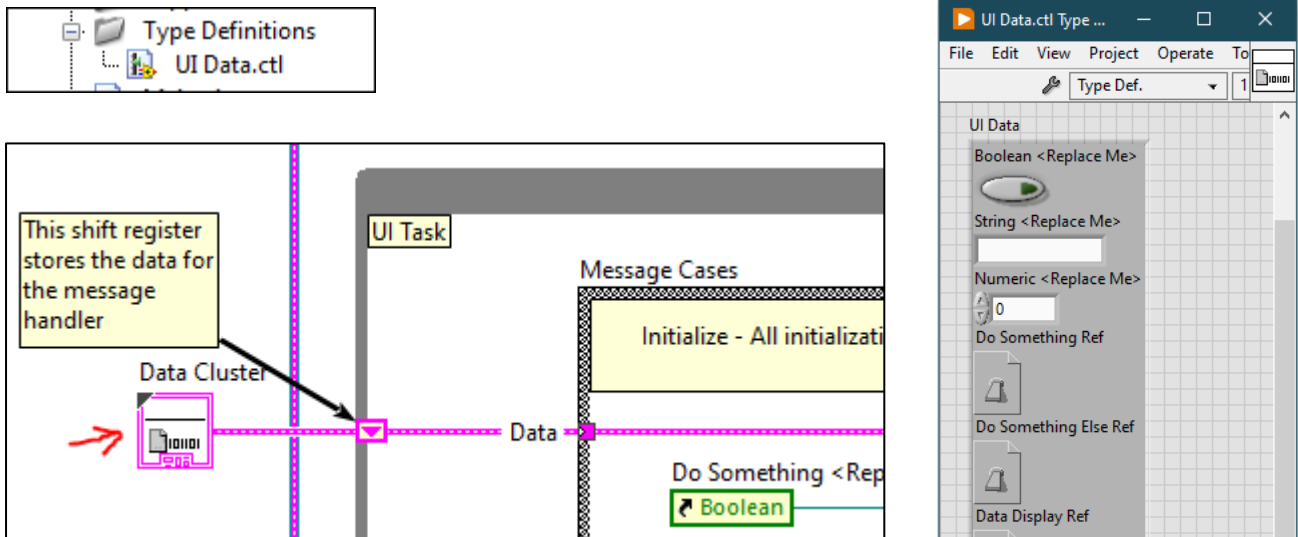
- Para garantizar el seguimiento de todos los errores, utilice la función Merge Errors para combinar los errores procedentes de todos los nodos del sub diagrama.
- Para acceder a los datos de los mensajes, conecte el túnel de entrada **Message Data** a una función Variant to Data:



- Para enviar un mensaje a una cola de mensajes, utilice la función Enqueue Message VI.
4. Para ejecutar el diagrama de mensajes, envíe un mensaje al MHL. El mensaje que envíe debe coincidir con la etiqueta del diagrama de mensajes que introdujo en el paso 2.

## Definición de los datos que necesita un Message Handling Loop

En la plantilla, UI Data.ctl es el typedef que define el cluster de datos al que el MHL puede acceder:

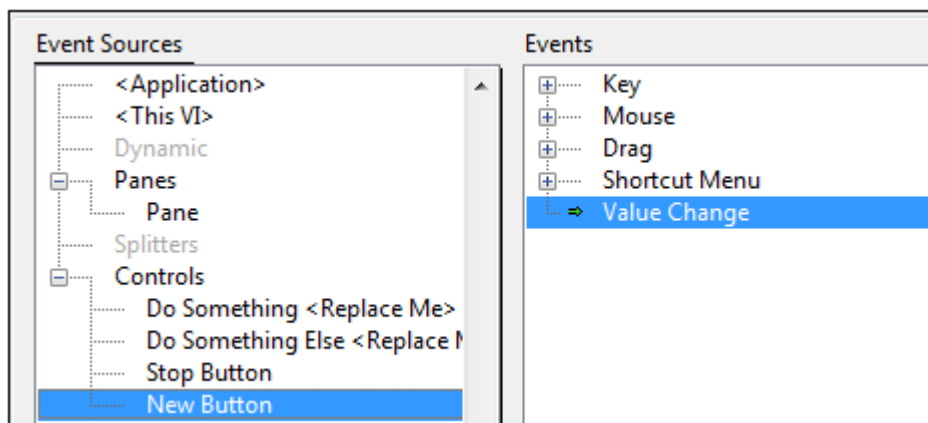


Modifica este typedef según las necesidades de tu aplicación. Por ejemplo, si más de un diagrama de mensajes en un MHL necesita modificar el mismo control booleano, añade un control booleano a este typedef.

Si tiene más de un MHL, cree un typedef para cada uno. Este diseño evita errores al garantizar que un MHL no pueda acceder a los datos de otro.

## Adición de un control que envía un mensaje a un Message Handling Loop

1. Añada un control al panel frontal.
2. (Opcional) Si desea que un diagrama de mensajes modifique programáticamente este control, incluya el refnum del control en el typedef para ese MHL.
3. Agregue un Event case a la estructura Event en el Event Handling Loop.
4. Configure el evento para que se dispare cuando cambie el valor de este nuevo botón:



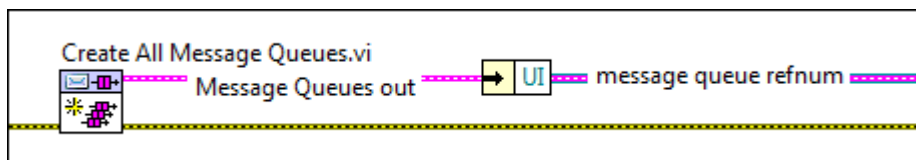
5. Haga clic en OK. LabVIEW crea un caso de evento en la estructura Event.
6. Asocie el terminal con el evento arrastrando el terminal del diagrama de bloques para el nuevo control dentro de este caso de evento.
7. Añada código a este caso de evento que envíe un mensaje a un MHL.

## Envío de un mensaje a un Message Handling Loop

Los mensajes son cadenas que ordenan a un MHL que ejecute uno de sus diagramas de mensajes. Los mensajes son producidos por el EHL y se almacenan en la cola de mensajes. Cada iteración del MHL lee el mensaje más antiguo de la cola de mensajes y ejecuta el diagrama de mensajes correspondiente.

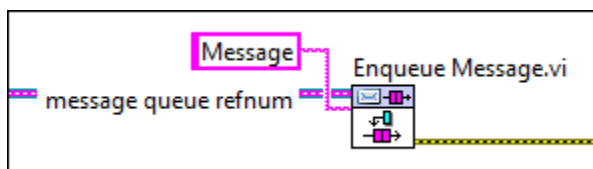
Completa los siguientes pasos para enviar un mensaje a un MHL:

1. Decide qué parte de la aplicación enviará el mensaje y qué MHL lo recibirá. Puede enviar mensajes desde el EHL o desde un diagrama de mensajes.
2. Decida qué diagrama de mensajes se ejecutará cuando el MHL reciba este mensaje. Asegúrese de que el diagrama de mensajes existe y tiene el mismo nombre que el mensaje que desea enviar. Si el diagrama de mensajes no existe, créelo.
3. En Main.vi, accede al cable que representa la cola de mensajes del MHL receptor. Se accede a este cable desagregándolo del cluster Message Queues out que se devuelve desde el Create All Message Queues VI. Main.vi ya contiene el siguiente código que desagrupa el UI queue refnum:



Expanda esta función Unbundle by Name para acceder a los cables de las colas de mensajes de todos los MHL.

4. En la parte de la aplicación que enviará el mensaje, crea el siguiente código:

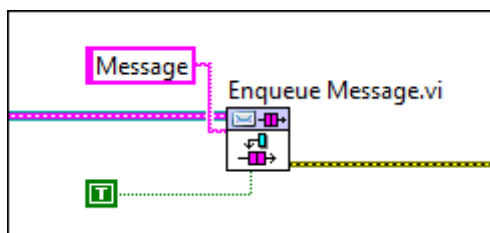


En el código anterior, Message es el texto que coincide con el diagrama de mensajes que identifiqué en el paso 2, y el refnum de la cola de mensajes es el cable que identifiqué en el paso 3.



**Nota.** Acceda al Enqueue Message VI desde la ventana **Project Explorer** de la plantilla o usando Quick Drop.

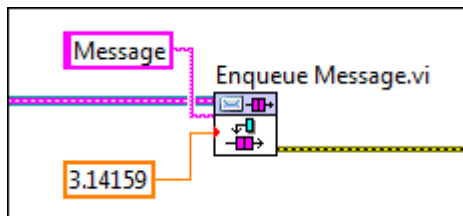
5. (Opcional) Para especificar que este mensaje sustituye a otros que ya están en la cola, conecte TRUE a la entrada **Priority Message?** del Enqueue Message VI:



Un mensaje de alta prioridad suele reservarse para situaciones de parada de emergencia. El mensaje de alta prioridad se coloca al principio de la cola de mensajes, garantizando que el MHL receptor consuma ese mensaje a continuación.

Esta plantilla sólo permite un mensaje como mensaje de alta prioridad. Por defecto, esta plantilla utiliza el mensaje Salir como mensaje de alta prioridad.

6. (Opcional) Para enviar datos con el mensaje, cablee un valor a la entrada **Message Data** del VI Enqueue Message. Este terminal puede aceptar cualquier tipo de datos. Por ejemplo, el siguiente código envía un número de punto flotante de doble precisión junto con el mensaje:



LabVIEW muestra un punto de coerción en el terminal de entrada porque el tipo de datos de este terminal es ariant.

### Cambiando el Mensaje que Detiene un Message Handling Loop

Los mensajes son cadenas, lo que significa que puede crear o cambiar un mensaje sin modificar un typedef. Sin embargo, esta plantilla define un mensaje: el mensaje Exit se define en Message Queue.lvlib:Dequeue Message.vi:



Si desea que sus MHLs se apaguen con un mensaje diferente a Exit, cambie el mensaje en este VI.

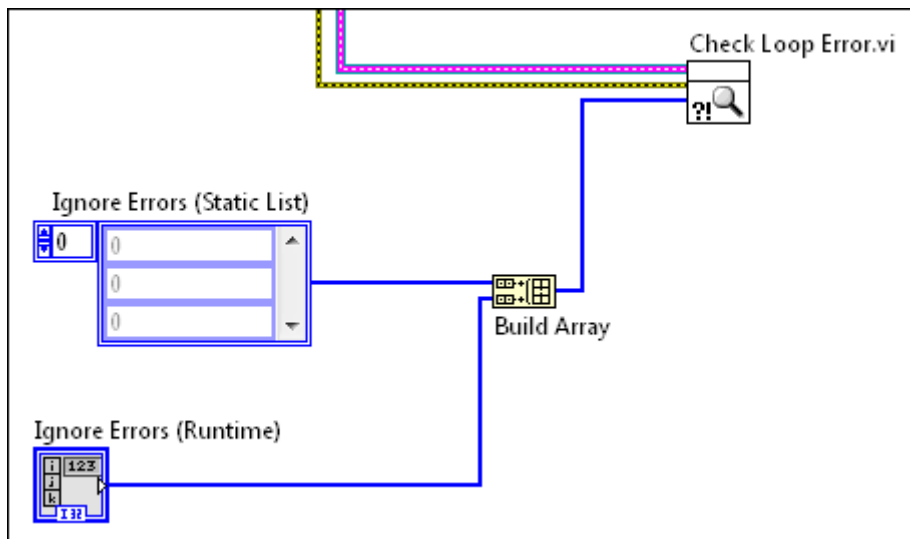
### Manejo de mensajes no reconocidos

Asegúrese de que todas las estructuras Case en un MHL tienen un diagrama de mensajes **Default**. El código en este diagrama de mensajes se ejecuta cuando el MHL lee un mensaje que no tiene un diagrama de mensajes correspondiente. Tener un diagrama de mensajes **Default** es importante porque los mensajes son cadenas que usted ingresa mientras programa, no valores que selecciona de un enum.

### Ignorando Errores en el Bucle de Manejo de Eventos y en el Bucle de Manejo de Mensajes

Puedes especificar una lista de errores a ignorar tanto para el EHL como para el MHL utilizando sus respectivos subVIs manejadores de errores completando los siguientes procedimientos:

1. En la ventana del Explorador de Proyectos, abra Error Handler - Event Handling Loop.vi o Error Handler - Message Handling Loop.vi, dependiendo de donde quiera que LabVIEW ignore los errores, y muestre el diagrama de bloques.
2. Encuentre la constante de matriz etiquetada **Ignore Errors (Static List)**:



Para ignorar errores en el EHL o MHL, añada códigos de error al array. Especifique un código de error por elemento de la matriz. Los errores a ignorar dependen de su aplicación. Por ejemplo, si está leyendo la cola a través de una red, es posible que desee ignorar los errores de tiempo de espera.

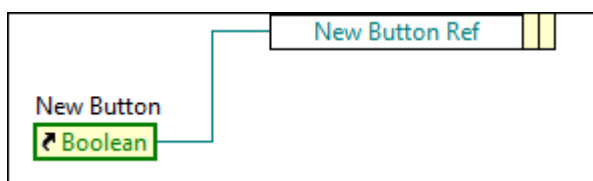
También puede especificar ignorar errores dinámicamente en tiempo de ejecución cableando un control de array a la entrada **Ignore Error (Runtime)** del subVI manejador de errores. Especifique un código de error por elemento del array.

LabVIEW trata todos los errores que no especifique como errores ignorados como errores regulares. Si los errores regulares ocurren en el EHL, LabVIEW envía mensajes de Error a la cola de mensajes. Puede definir cómo el MHL consume los mensajes de Error. Si los errores regulares ocurren en el MHL, LabVIEW envía mensajes de Salida a la cola de mensajes. Asegúrese de que el MHL se detiene ejecutando el diagrama de mensajes Exit. Los errores regulares en el MHL pueden colocar el QMH en un estado desconocido.

### Habilitación de un Message Handling Loop para que Modifique Programáticamente Controles e Indicadores

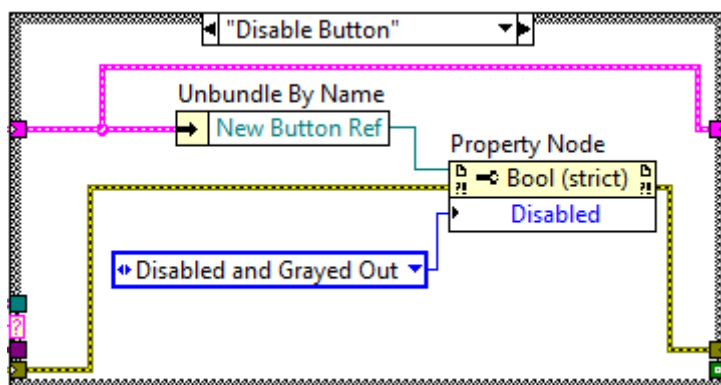
Para habilitar que un MHL pueda modificar programáticamente un control, cree un refnum para el control y agréguelo al typedef que almacena datos para ese MHL. El siguiente procedimiento utiliza UI Data.ctl como ejemplo.

1. Crea el control refnum y muévelo al subdiagrama Initialize del MHL.
2. Abre UI Data.ctl y añade espacio para el refnum al cluster.
3. En el subdiagrama Initialize, expanda este nuevo terminal en la función Bundle by Name y conéctele el refnum del control:



El refnum de este control ahora está disponible para cualquier diagrama de mensajes que tenga acceso a UI Data.ctl. Por ejemplo, el siguiente código muestra el diagrama de

mensajes Disable Button, que utiliza el refnum del control para desactivar y poner en gris el botón del panel frontal:



### Información relacionada

Consulte la *Ayuda de LabVIEW*, disponible seleccionando **Help»LabVIEW Help** desde LabVIEW, para obtener información sobre conceptos u objetos de LabVIEW utilizados en esta plantilla. También puede utilizar la ventana de **Context Help** para aprender información básica sobre los objetos de LabVIEW mientras mueve el cursor sobre cada objeto. Para mostrar la ventana de **Context Help** en LabVIEW, seleccione **Help»Show Context Help**.

Consulte el proyecto de ejemplo Continuous Measurement and Logging, disponible en el cuadro de diálogo **Create Project**, para ver un ejemplo de adaptación de esta plantilla a una aplicación de medida.

Consulte ni.com para una guía para desarrolladores de esta plantilla.