# NI QMH Template

May 8, 2015 by Fabiola De La Cueva

<http://www.walkingthewires.com/2015/05/08/ni-qmh-template/>

Durante la Cumbre CLA en Roma, me sorprendió ver sólo un puñado de manos levantadas cuando pregunté cuántas personas estaban familiarizadas con la plantilla de proyecto Queued Message Handler de National Instruments (NI) que se entrega con LabVIEW 2012 y posteriores. Puedo entender que la gente no mire las plantillas de proyecto, pensando que son similares a las viejas plantillas VI que han estado disponibles en el cuadro de diálogo *File > New* durante muchos años (Producer-Consumer, State Machine, etc.). A diferencia de las antiguas plantillas de VI, que son más instructivas, las nuevas plantillas de proyecto están pensadas como puntos de partida para aplicaciones del mundo real. También hay varios proyectos de ejemplo que implementan algunas aplicaciones comunes como la adquisición continua y el registro.

Algunos arquitectos han creado sus propias plantillas de proyecto, algunas de las cuales parten de la antigua plantilla Producer-Consumer. El QMH de NI es un buen punto de partida para muchas aplicaciones, y aborda muchas cuestiones que no son manejadas por la plantilla Producer-Consumer, tales como detener el bucle de manejo de eventos (EHL) desde el bucle de manejo de mensajes (MHL) a través de un evento de usuario, manejo de errores simple, etc. Sin embargo, hay algunas cuestiones a tener en cuenta cuando se trabaja con la plantilla de proyecto QMH de NI:

**User Event Registration envuelto dentro de un subVI** - En el QMH de NI, la función [Register for Events](https://www.ni.com/docs/en-US/bundle/labview-api-ref/page/functions/register-for-events.html) existe en un subVI, a diferencia del mismo diagrama propietario que el Event Handling Loop.

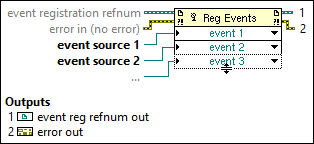


Ilustración 1: Register For Events

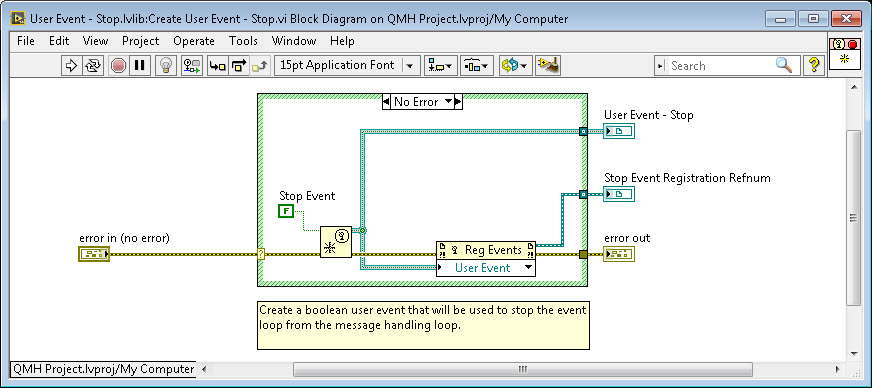


Ilustración 2: QMH Create User Event – Stop.vi

Hay dos problemas con este enfoque. En primer lugar, ya que el *event registration refnum* pasa a través del límite de un subVI, el event case que maneja el user event obtiene un nombre no descriptivo de "User Event". El otro problema es que los desarrolladores menos experimentados podrían tener la tentación de bifurcar el cable de registro. Siempre debería haber una relación de uno a uno entre una función de Register for Events y un Event Structure. El principal problema con el uso de un único *event registration refnum* para múltiples funciones de Register For Events que se crea un único motor de eventos, y esto puede llevar a que una de las estructuras de eventos se pierda los eventos para los que está registrada. Si desea obtener más información acerca de este y otros matices de los User Events, le recomiendo tomar el curso National Instruments [Advanced Architectures in LabVIEW](https://www.ni.com/es/shop/services/education-services/customer-education-courses/advanced-architectures-in-labview-course-overview.html). Se rumorea que en la versión de la plantilla de proyecto QMH de NI de LabVIEW 2015, la función Register For Events ya no está envuelta en un subVI. Eso es verdad.

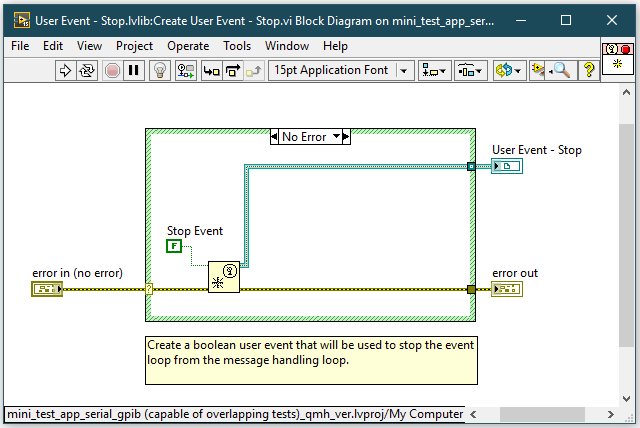


Ilustración 3: Actual LV 2015 Create User Event – Stop.vi

**La plantilla de proyecto crea su propia copia de la librería de User Event así como la Message Queue library** - Esto significa que cada nuevo proyecto tendrá una copia de estas librerías. Esto puede ser confuso para desarrolladores inexpertos, y podrían hacer modificaciones a una de estas librerías en un proyecto QMH dado y olvidarse de las modificaciones cuando vayan a empezar un nuevo proyecto QMH. Además, todos los VIs acaban teniendo los mismos bonitos iconos, haciendo difícil detectar de un vistazo que estos VIs pueden haber sido modificados. Esto también significa que, si hay mejoras hechas a la plantilla del proyecto NI QMH, usted tendrá que volver a sus proyectos existentes y hacer las modificaciones usted mismo. Esto no es necesariamente malo, pero es algo para tener en cuenta. Si tiene que dar soporte a un proyecto existente, no asuma que los desarrolladores originales dejaron las librerías User Event y Message Queue sin cambios.

**Priority enqueueing (enqueueing at the front of the queue)** - Esta característica es muy útil para asegurar que el mensaje de salida se pone por delante de cualquier otro mensaje en la cola de mensajes. Sin embargo, a menudo he visto a desarrolladores dar alta prioridad a varios mensajes. Cuando hay varios mensajes de alta prioridad, no hay garantía de que el mensaje de salida se mantenga al frente de la cola de mensajes. Una medida adicional a tomar si está utilizando LabVIEW 2014 o posterior es hacer que disparar el Stop User Event sea de alta prioridad.

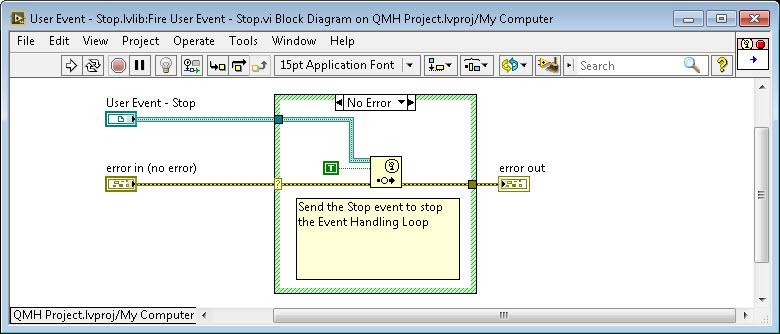


Ilustración 4: Fire User Event – Stop.vi

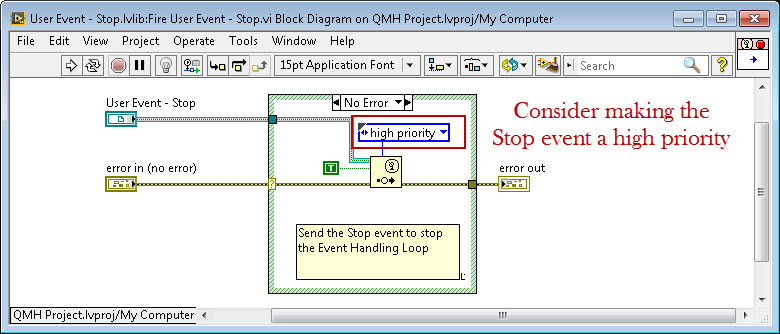


Ilustración 5: Modified QMH Fire User Event – Stop.vi

**Usando el QMH como una Máquina de Estado** - Los desarrolladores y arquitectos avanzados de LabVIEW se han alejado del uso del QMH debido a malas experiencias tratando de usarlo como una máquina de estado. El principal problema es tratar de poner en cola una serie de mensajes que necesitan ser ejecutados uno detrás del otro sin interrupción (excepto para el mensaje Exit). Una forma sugerida para evitar esto es poner en cola múltiples mensajes usando un bucle For. El NI QMH logra esto proporcionando una instancia polimórfica "enqueue multiple messages" del VI enqueue. Sin embargo, todavía hay una pequeña posibilidad de que un mensaje pueda ser puesto en cola por un proceso paralelo entre dos iteraciones de bucle dentro de este VI. Mi sugerencia es utilizar un subVI de una máquina de estado que también se registra para escuchar el evento de parada. Esta máquina de estado puede ser implementada con la plantilla de proyecto Simple State Machine con la estructura Event en el caso idle. Otra opción es tener los casos de la máquina de estado en el evento timeout de la estructura Event.

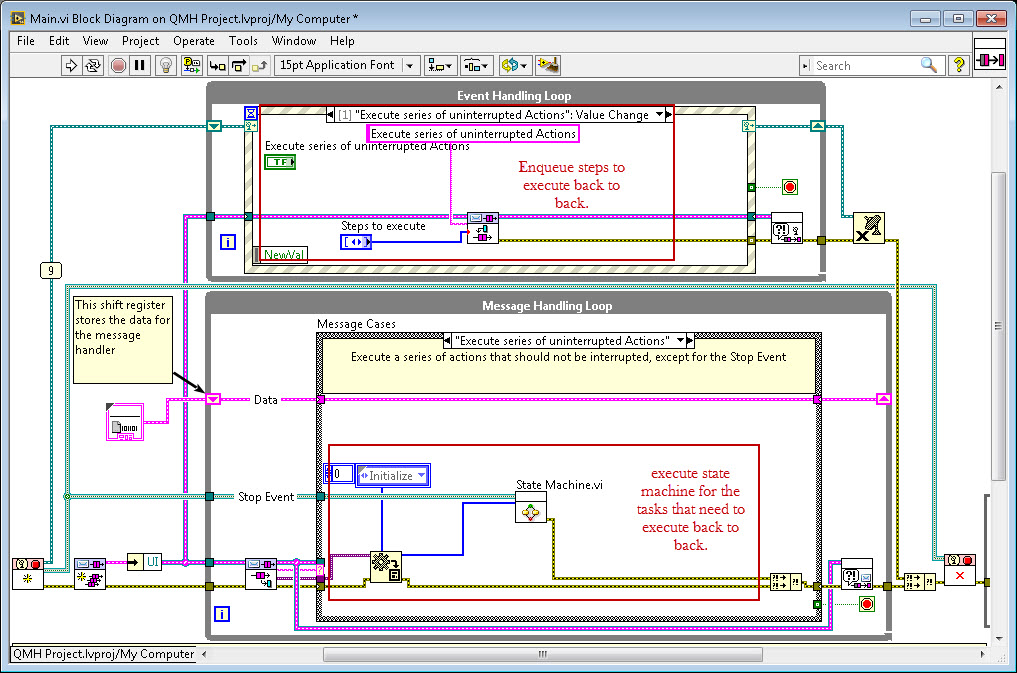


Ilustración 6: Use State Machine VI in MHL of QMH Main.vi. The developer of this VI intends to send the Stop Event to the State Machine.vi. Notice that while the State Machine VI is running, the stop event will not be fired if the VI is left as the original template

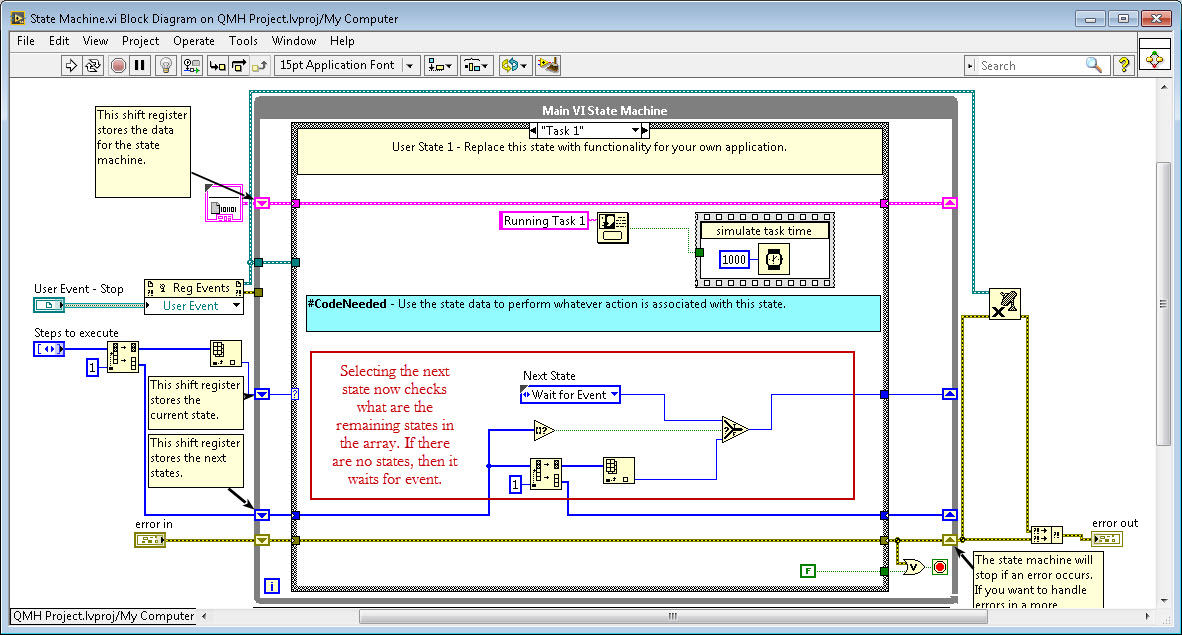


Ilustración 7: State Machine modified to take the steps to execute back to back as an array. This option will have to finish all the series tasks before servicing the Event Structure. This means the tasks back to back have to complete before the Stop Event is addressed.

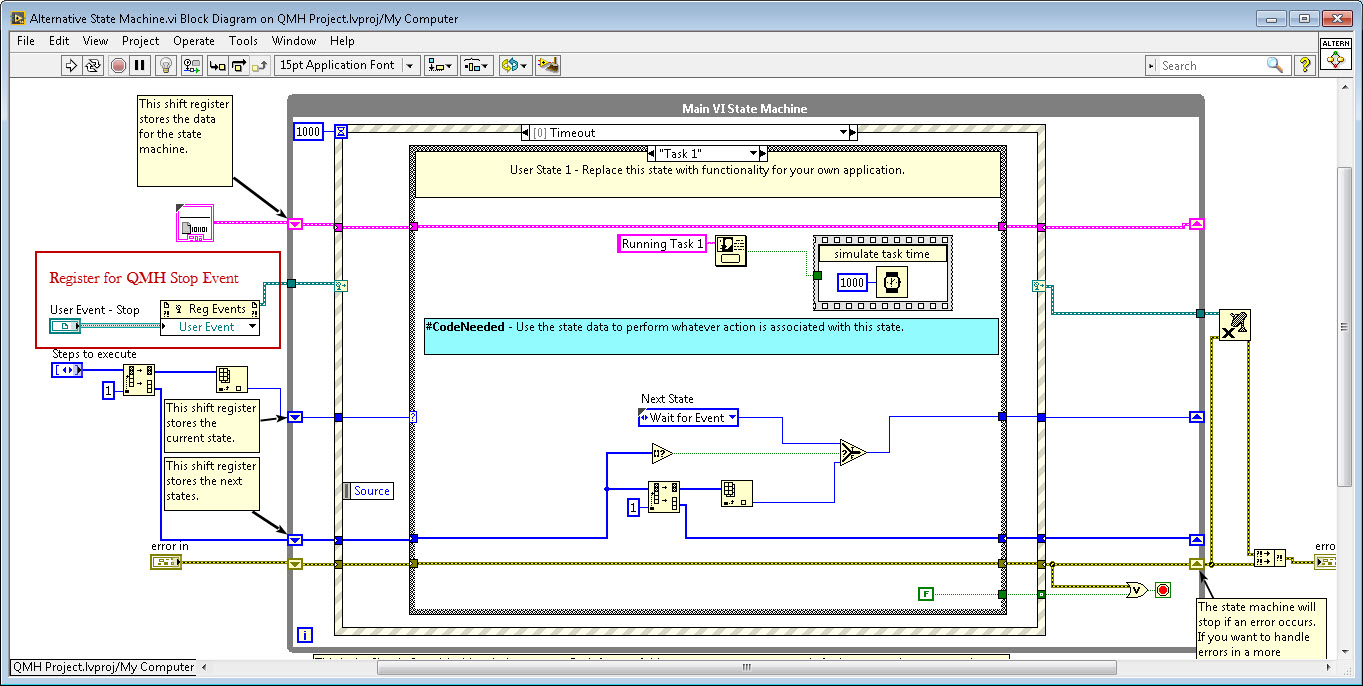


Ilustración 8: If the developer wishes the State Machine to be able to stop in the middle of a series of steps, then the case structure for the steps can be put in the Timeout event case. This means the stop event can be addressed in between tasks.

Llegados a este punto, alguien suele preguntar: *"¿Por qué no implementar la QMH como un único bucle, igual que la máquina de estados JKI?*". Esto aseguraría que los nuevos estados no se ponen en cola hasta que se ejecuta una serie de estados. Esta es definitivamente una buena opción, y tiene el beneficio añadido de mantener todos los datos locales en el registro de desplazamiento del bucle. También hace que sea más fácil añadir datos de los casos de eventos. Sin embargo, el dequeuer necesita un tiempo de espera que envíe la máquina de estado al caso "Idle" para atender eventos, y la estructura Event termina teniendo un tiempo de espera también para asegurar que no se pierda ninguna cola. Esto convierte el bucle de la máquina de estados en una máquina de sondeo de eventos/mensajes. En el NI QMH, no hay necesidad de tiempos de espera en la estructura de eventos o en el dequeuer. Si no hay interacción del usuario con el panel frontal, entonces todo el VI está simplemente inactivo hasta que un evento del panel frontal lo despierte.

Resumiendo - Si desea obtener más información sobre la plantilla de proyecto NI QMH y averiguar cómo puede utilizarla (o no), haga clic en Crear Proyecto desde la Ventana de Introducción en LabVIEW y, a continuación, haga clic en el enlace "Más Información" dentro de la descripción de la plantilla de proyecto QMH. También puedes crear tu primer proyecto basado en esta plantilla, que incluirá una copia de la documentación. Finalmente, asegúrate de revisar la presentación llamada [Queued Message Handler: Caveats and Design Decisions](https://forums.ni.com/t5/Past-NIWeek-Sessions/NIWeek-2013-Queued-Message-Handler-Caveats-and-Design-Decisions/ta-p/3502234).

Próximamente compartiremos nuestra propia plantilla de proyecto QMH. La llamamos Delacor QMH o DQMH. Las principales modificaciones que hemos hecho son abordar el problema con el registro de eventos y utilizar Eventos de Usuario como nuestro mecanismo de comunicación entre procesos. Obtuvimos algunas de estas ideas de la presentación de [Justin Goeres de 2011 sobre Eventos de usuario privados/públicos](https://www.jki.net/blog/events/cla-summit-2011-public-private-events-framework-slides-available). También decidimos envolver la cola de mensajes en una clase, y hacerla privada al módulo DQMH tal y como presenté en la Cumbre Europea CLA de 2013 en París. Pero, por desgracia, me estoy adelantando. Estad atentos para saber más a finales de este año.

Feliz cableado,

Fabiola