LabVIEW UNIVERSITY	Guía de Aprendizaje Creación y documentación de SubVI	FREIBURG MECHATRONIK

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

Objetivo de aprendizaje

- Implementar un VI y documentarlo para luego ser utilizado como un SubVI

Resultados de aprendizaje

- Utilizar vis como SubVis y documentarlos de acuerdo a estándares de programación Labview de la industria.

Introducción

Has llegado a uno de los pilares de la programación en LABVIEW. Creacion de SubVis y **DOCUMENTACIÓN**.

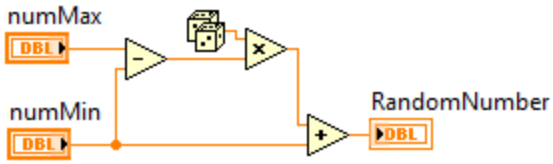
En la guía 3.3 había tenido oportunidad de ver cómo crear un programa que generara un valor aleatorio en un rango de trabajo, recordemos…

Para generar un valor aleatorio en un rango se debe utilizar la ecuación 1 (Eq.1)

$$RandomNumber = (numMax - numMin) * \big(randomNumbert(0-1)\big) + numMin$$
(Eq.1)

Asi si deseas generar números aleatorios en el rango 0-100 por ejemplo, tus variables serian:
numMax=100
numMin=0

Tu vi generador de números aleatorios de acuerdo a la ecuación (Eq. 1) quedaría así, en numMax selecciona el valor 100 como el current default y en el numMin deja el cero como default:



Ahora vas a apredner a convertir este vi en un SubVi, pero antes documentémoslo, sigue la guía y aprende.

Abre la Abre el programa asociado a esta guía que encuentras en la misma sección.

Nota: un SubVi recuerda es un programa que puedes usar dentro de otro vi… un sub vi para que sea Subvi debe cumplir con una serie de requerimientos entre los que listamos:

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

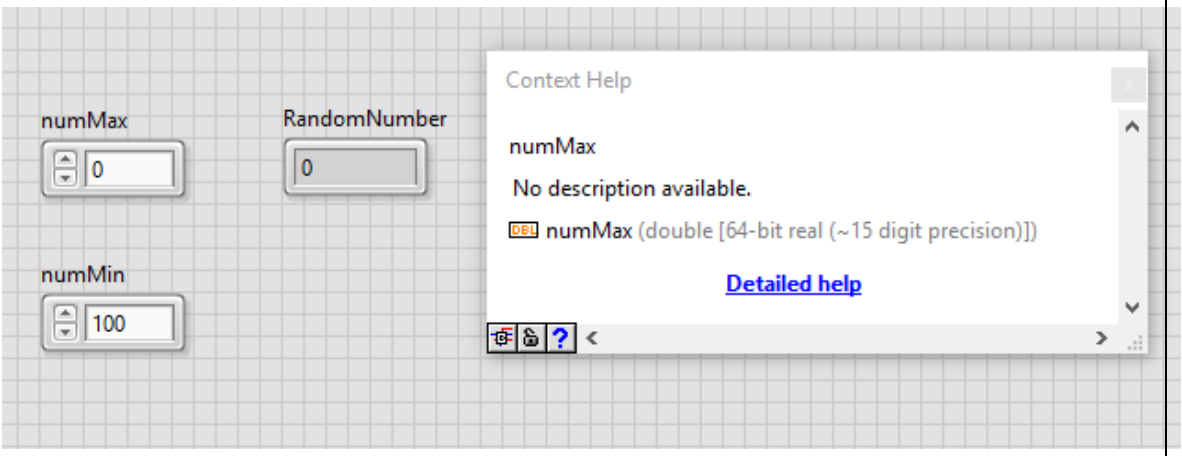
a. Documentación. (Parte I)
b. Gestión de errores. (Parte II)
c. Programa.

Vamos con la primera parte… Documentación para ello recuerda que debes documentarlo todo, acostúmbrate a eso… en grandes equipos de desarrollo trabajaras con diferentes programadores, si alguno de ellos no documenta correctamente entorpece el trabajo del equipo, por eso **JAMAS OLVIDES, un buen desarrollador CLAD documenta siempre sus programas.**

Empezamos.

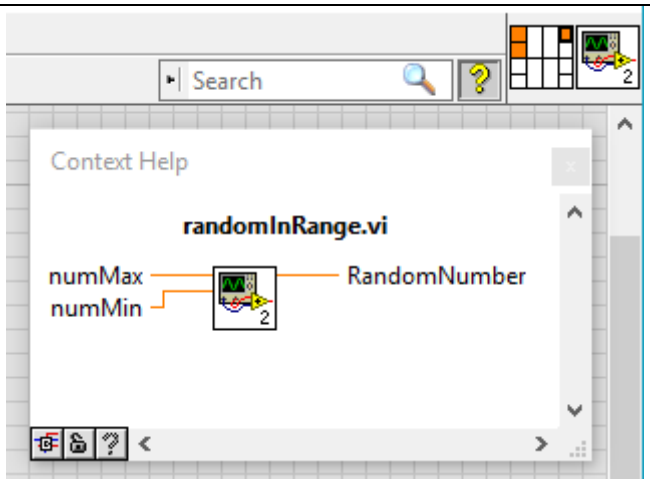
Desarrollo

1. Antes de documentar verifiquemos que tipo de documentación tiene el vi. Escribimos el comando Ctrl+h y nos posicionamos encima de los indicadores y controles, efectivamente no aparecen descripciones para ningún control o indicador. En este caso estamos sobre numMax.

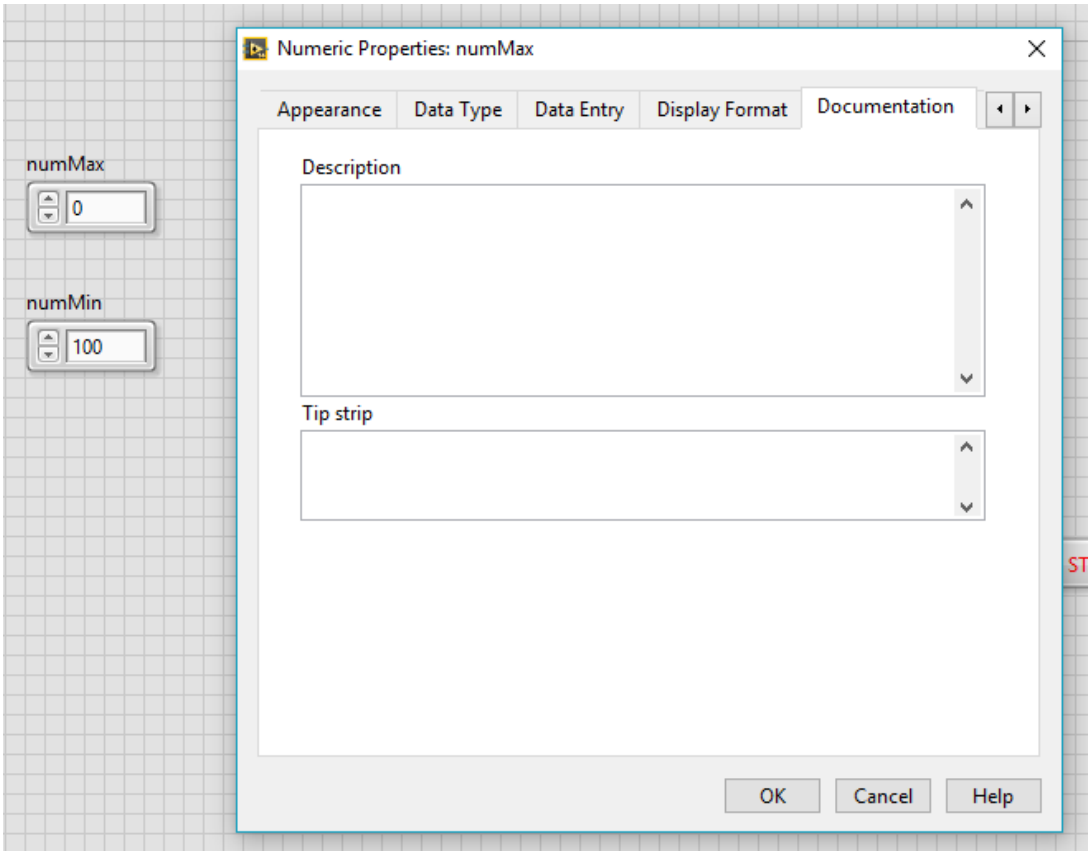


3. En la parte superior derecha solo observamos que esta cableado el icon connector pero no hay más anotaciones, tampoco existe un icono que lo identifique, esto también deberemos documentarlo.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



3. Para documentar un control o un indicador dale click derecho, selecciona propiedades y luego selecciona la pestaña llamada Documentation. (En las demás pestañas podras modificar tipos de datos, display format etc)



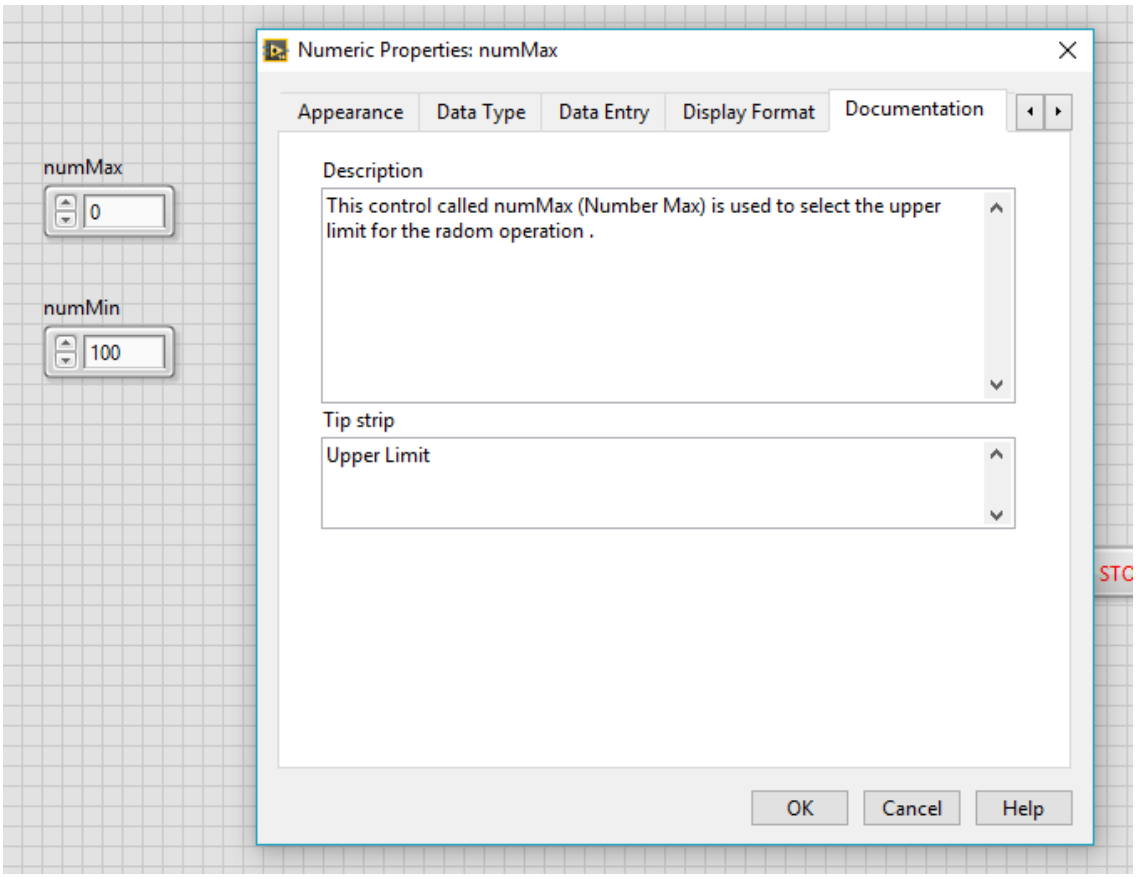
Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

En la ventana description se debe escribir el mensaje de ayuda para nuestros usuarios que queramos que aparezca en la ventana CRTL+H sobre el indicador o control.

En la ventana llamada Tip strip, se puede colocar un mensaje muy breve con el nombre del control o indicador, esto se hace para que cuando el usuario se posiciones encima del control o indicador, aparezca una pequeña ventana amarilla con dicho mensaje.

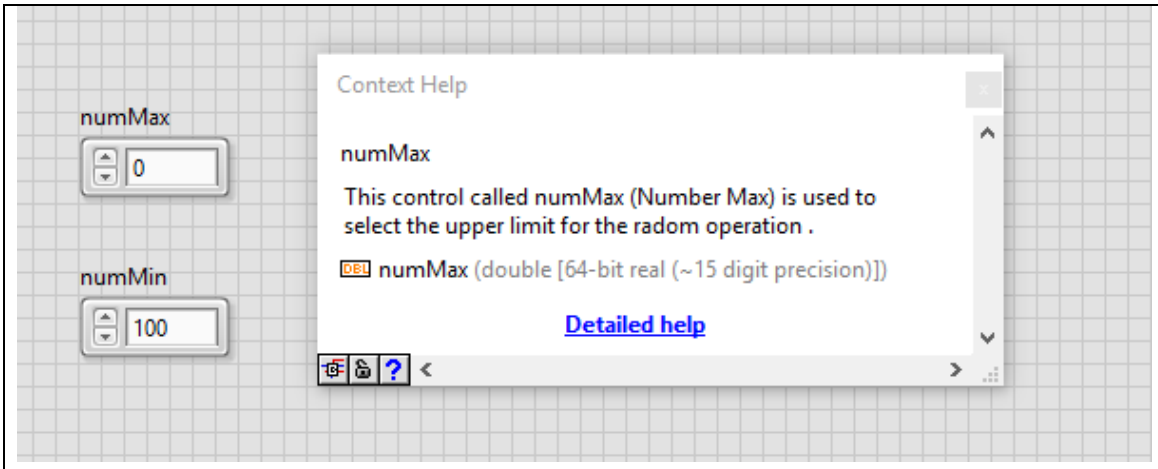
Documentamos este control como aparece a continuación para la descripción el mensaje “This control called numMax (Number Max) is used to select the upper limit for the radom operation .” y para el tipStrip las palabras Upper Limit.

(No olvides que todo se hace en ingles porque ese mercado parlante es mayor que el latino, por esa razón todos los programas están hechos en inglés).

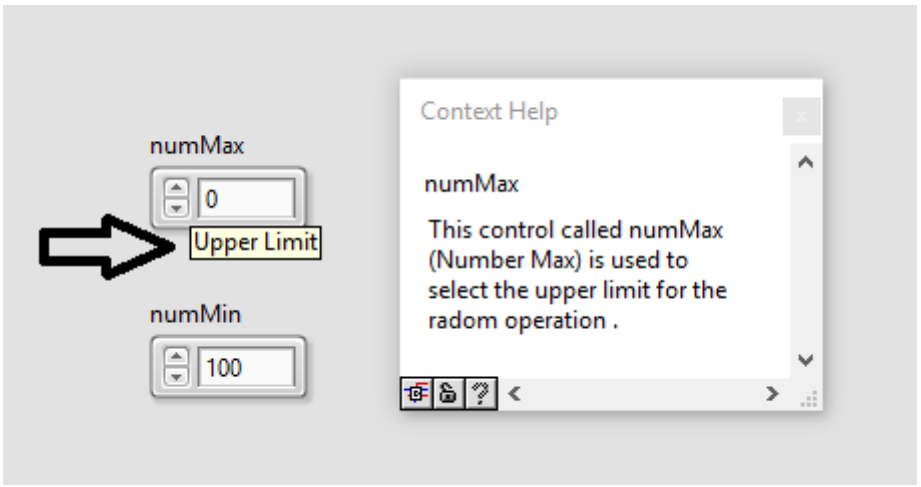


Dale OK y ahora observa como aparece el mensaje para nuestro control numMax una vez le hacemos CTRL+H y nos posicionamos encima del control.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



Ahora ejecuta la aplicación y posiciónate encima de ese control y observa la ventana del tip strip.



4. Realiza el mismo procedimiento del paso 3 para el control numMin con el siguiente mensaje, "This control called numMin(Number Min) is used to select the lower limit for the radom operation"

El tipStrip documentalo con "Lower limit"

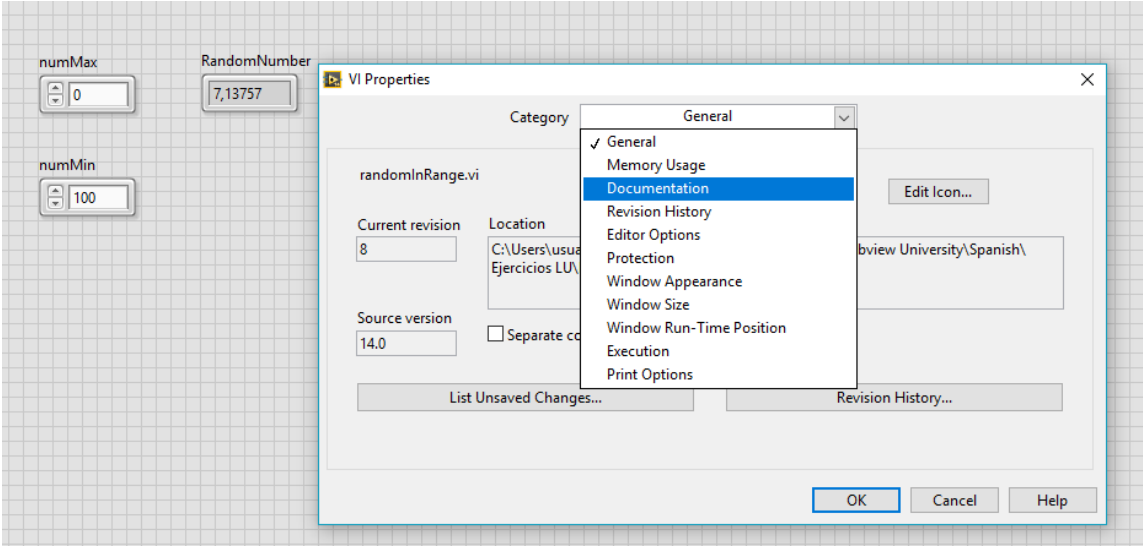
5. Realiza el mismo procedimiento del paso 3 pero para el indicador randomNumber y en documentación asigna el siguiente mensaje "This indicator show a random number between the upper and lower limit selected with the controls, the number changes each second"

El tipStrip documentalo con "Random Number in Range".

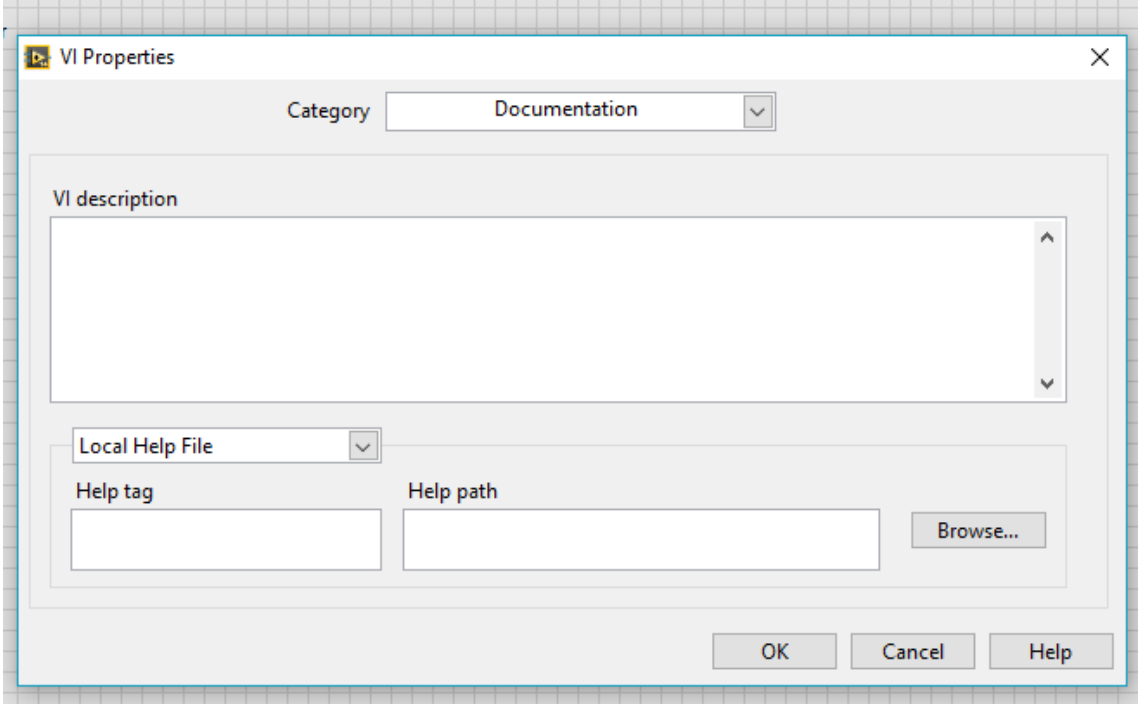
Al finalizar tendrás los 2 controles y el indicador documentados.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

6. Ahora que has documentado los controles e indicadores es hora de documentar el VI en total, para esto sigue la ruta file>>VI Properties, te aparece una nueva ventana, ahí selecciona la opción documentation.



En la nueva ventana veras que aparece una ventana así:

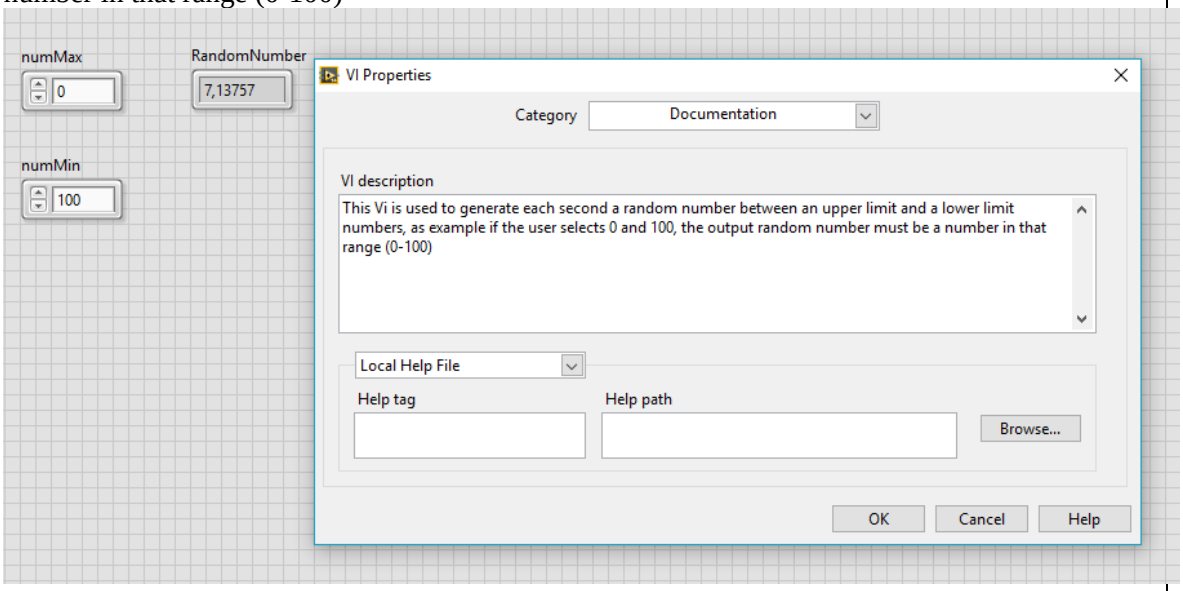


En el campo VI Description deberás escribir la mayor información posible para tus usuarios en donde describas el objetivo de tu VI, recuerda tu usuario final no conoce como desarrollaste tu

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

aplicación y es importante no dejar cabos sueltos, se generoso en la descripción pero tampoco inviertas demasiado tiempo, se preciso(a) y conciso(a) con la información para nuestro caso copia y pega el siguiente mensaje.

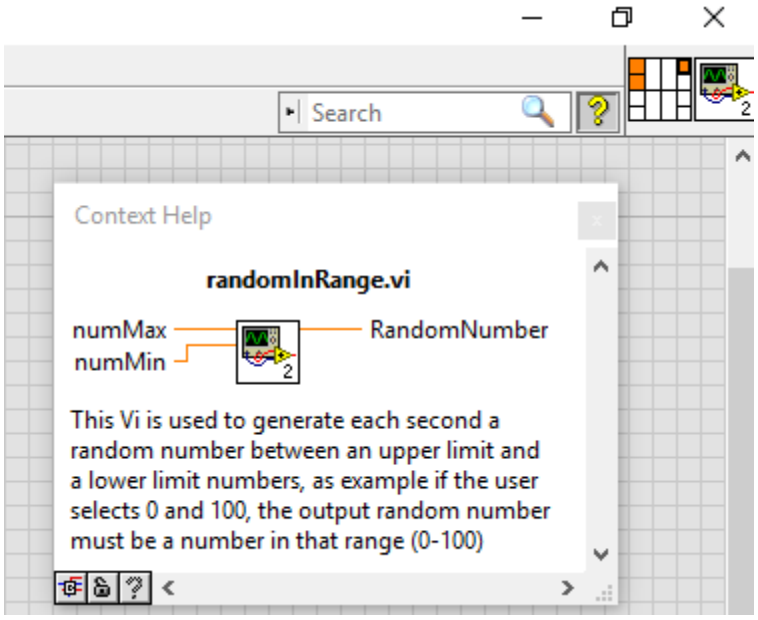
"This Vi is used to generate each second a random number between an upper limit and a lower limit numbers, as example if the user selects 0 and 100, the output random number must be a number in that range (0-100)"



Dale OK a la ventana.

7. Escribe el comando CTRL+H y posicionate en la parte superior derecha del VI y observa como ahora aparece información sobre tu vi que fue agregada en el paso 6.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

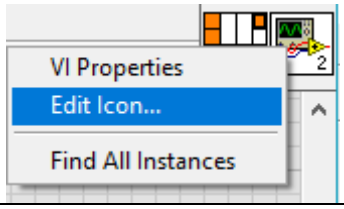


8. Es hora de modificar otro importante ítem en la documentación de cualquier VI y es el icono que se encuentra en la parte superior derecha… ATENCIÓN: Algunos programadores tiene la mala práctica de dejar el icono por defecto en este caso un icono de labview con el numero 2 que se muestra a continuación

Si tienes una aplicación con un subvi talvez este bien AUNQUE NO TE RECOMIENDO QUE DEJES POR FECTO EL ICONO, imagínate si tienes un proyecto con cientos o miles de Vis y todos con un icono similar… EVITA ESTA MALA PRACTICA Y MODIFICA TUS ICONOS GRAFICAMENTE DE TAL MANERA QUE TE AYUDA A IDENTIFICAR SU FUNCION PRINCIPAL.

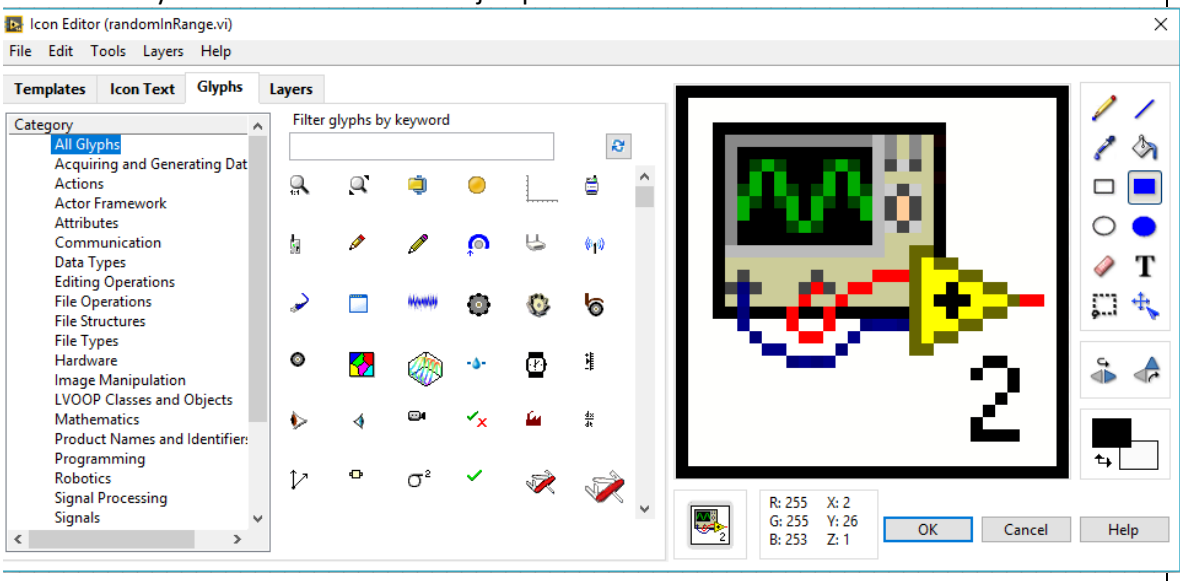
Para esta aplicación que genera números aleatorios podríamos o diseñar el icono o utilizar alguno de la galería, veamos.

9. Dale click derecho al icono a modificar, observa que aparece un menú con 3 opciones, en la primera opción (VI properties) encuentras una ruta alterna para la presentada en el paso 6 de esta guía. En la segunda opción aparece la opción Edit Icon, dale click

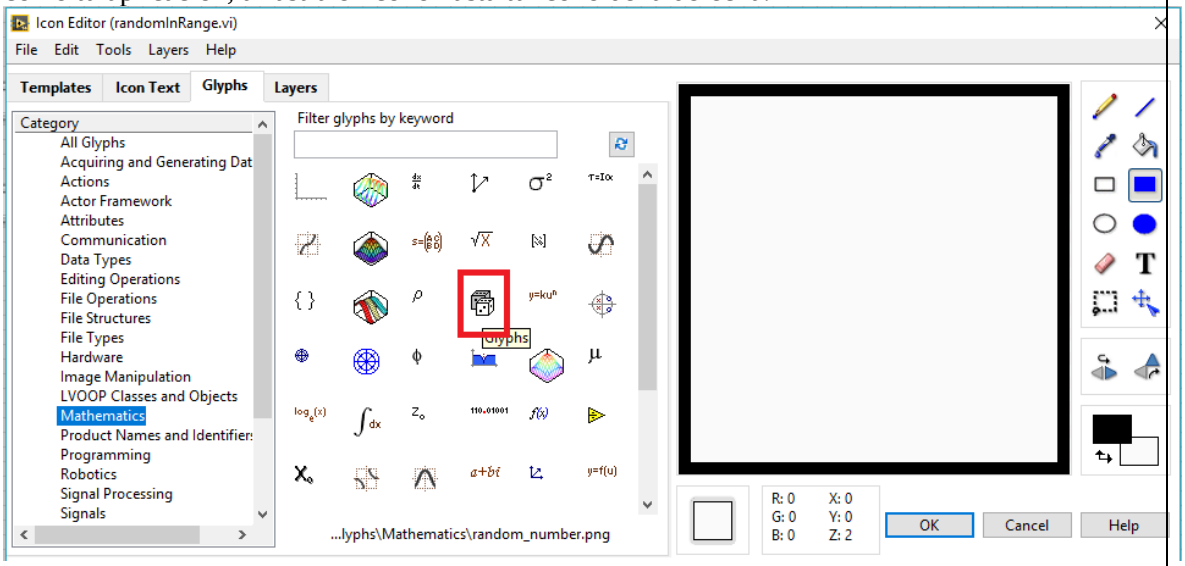


Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

10. Como un mini Paint podrias borrar el icono actual y crear el tuyo propio o podrias borrar el icono actual y seleccionar un icono de ejemplo.

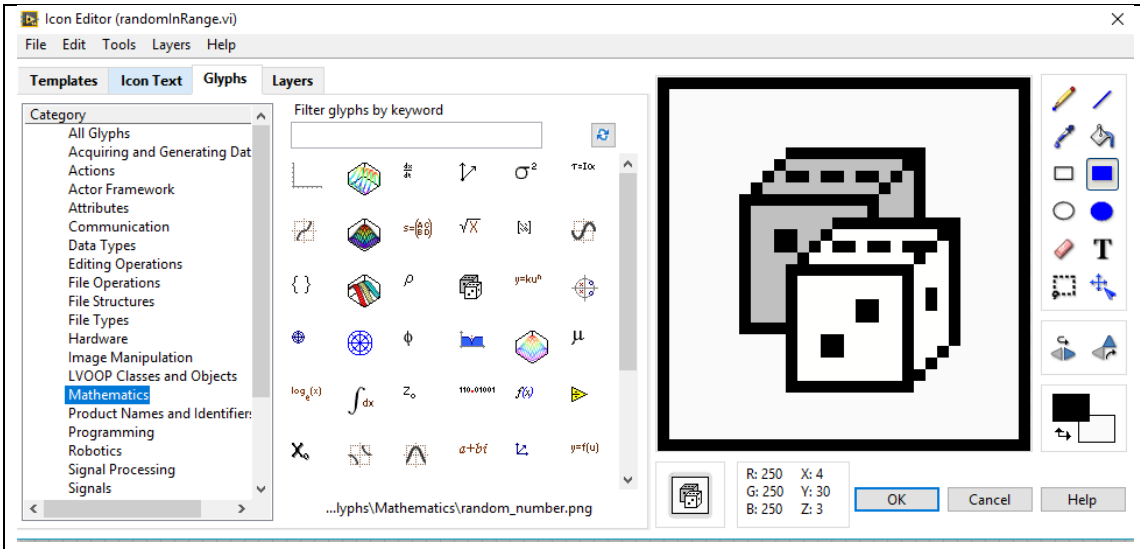


11. En este ejercicio borramos el contenido actual y luego selecciona en el menú de la izquierda el menú Mathematics, selecciona el icono de los dados que te dicen y lo puedes asociar con el azar como tu aplicación, arrastra el icono hasta tu icono de la derecha.



12. Una vez los ubiques en la posición deseada dale click y listo.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



13. Podrías seguir modificando el icono tanto como quieras, para fines de esta guía lo dejaremos así, dale click en OK y observa como tu vi ahora luce un nuevo icono.

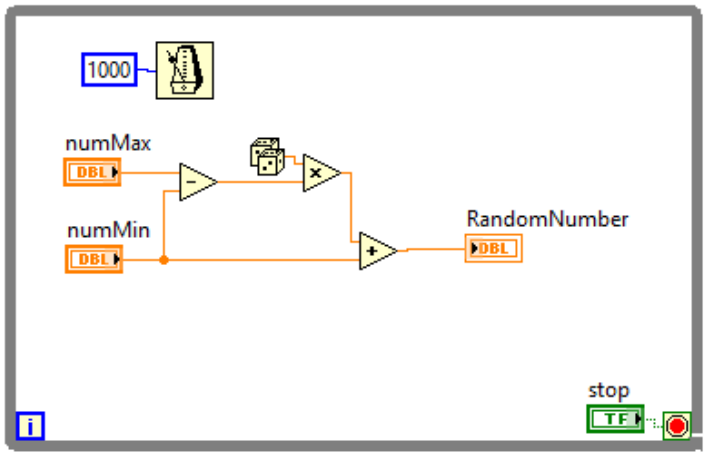
Es importante que tus VI y los de tu equipo de trabajo queden documentados así, por ahora has documentado en el front Panel… hora de irte para el block Diagram… así es, el programa también deberá ser documentado ☺.

14. En el block Diagram existen varias maneras de documentación, una de las más usuales es hacer un copy paste de ecuaciones, tablas y notas y agregarlas como ayuda al programador que esté trabajando con tu VI.

Copia con paint y pega la ecuación 1 (Eq. 1) en el block Diagram así:

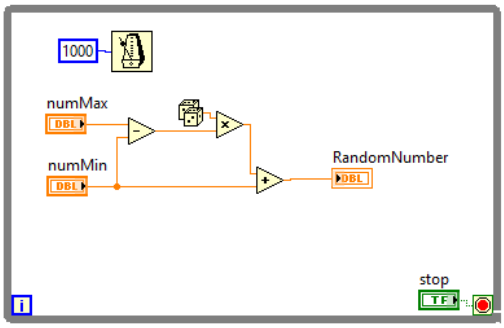
Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

$$RandomNumber = (numMax - numMin) * (randomNumbert(0-1)) + numMin \quad \text{(Eq.1)}$$



Ahora podrias utilizar etiquetas libres para documentar la ecuación, para ello selecciona la opción label del menú tolos y completa y documenta con el siguiente mensaje: “This equation is used to calculate a number between an upper and a lower limit, in this VI the number changes each second due the wait funcion 1000mS = 1seg”

$$RandomNumber = (numMax - numMin) * (randomNumbert(0-1)) + numMin \quad \text{(Eq.1)}$$

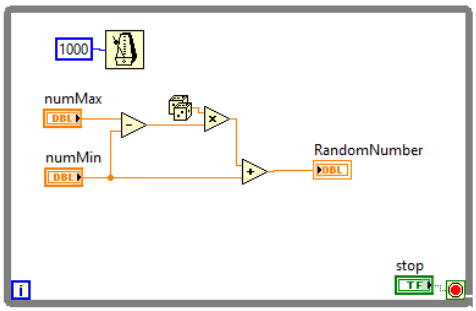


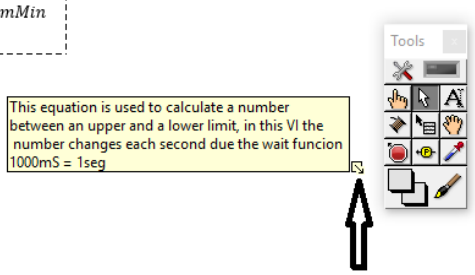
This equation is used to calculate a number between an upper and a lower limit, in this VI the number changes each second due the wait funcion 1000mS = 1seg

15. Selecciona la opción select en la ventana Tools y posicionate en el label creado, arrastra de la esquina inferior del label la flecha hasta lo que desees que quede vinculado con la etiqueta luego posiciona en donde desees esa etiqueta, asi:

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

$RandomNumber = (numMax - numMin) * (randomNumbert(0-1)) + numMin$ (Eq.1)

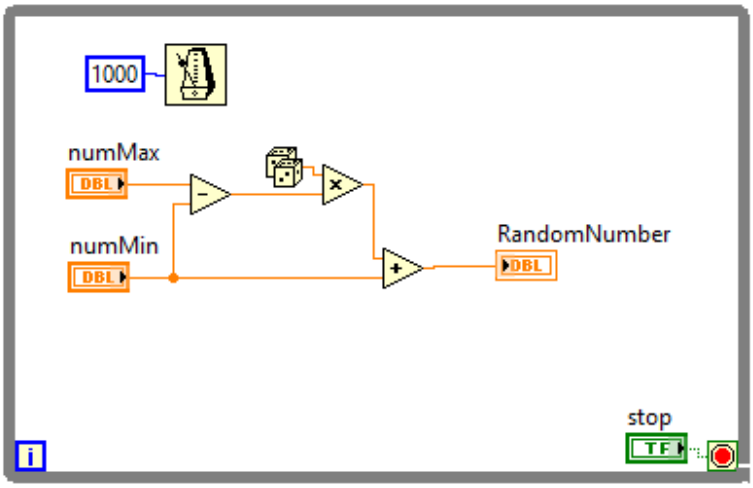




Una vez vinculado a la ecuación cambia si lo deseas la ubicación de la etiqueta obteniendo finalmente lo siguiente:

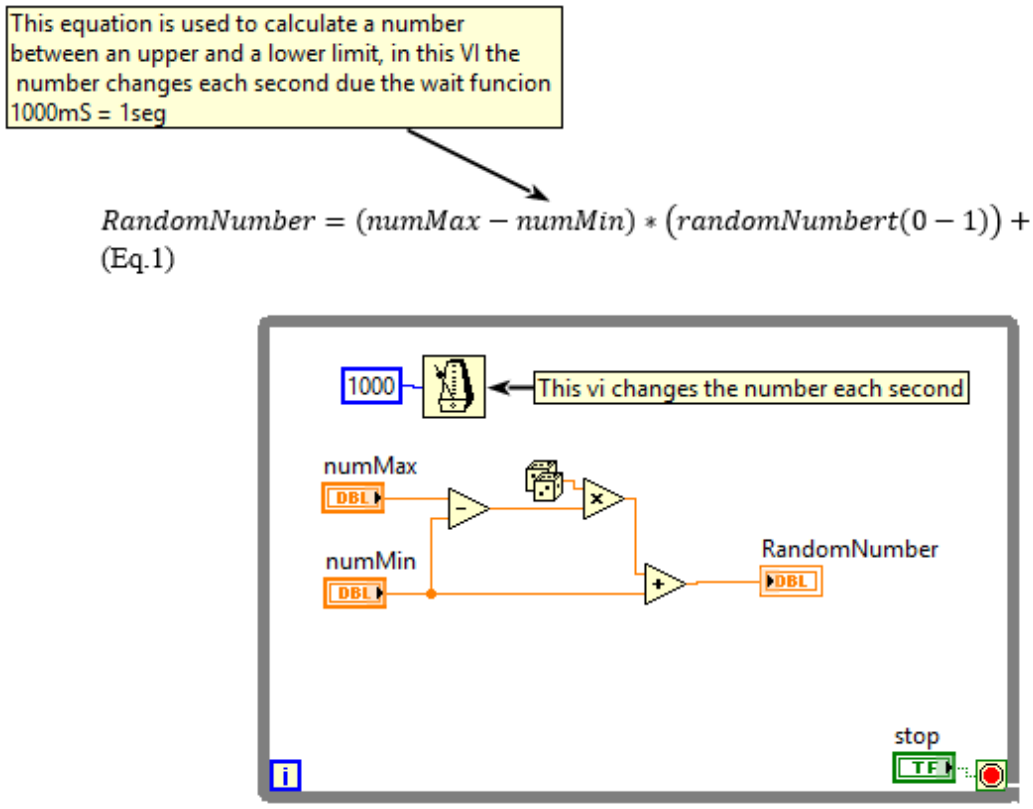
This equation is used to calculate a number between an upper and a lower limit, in this VI the number changes each second due the wait funcion 1000mS = 1seg

$RandomNumber = (numMax - numMin) * (randomNumbert(0-1)) + numMin$ (Eq.1)



16. Agrega toda la documentación que requieras sea suficiente como para que otro desarrollador comprenda tu código.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

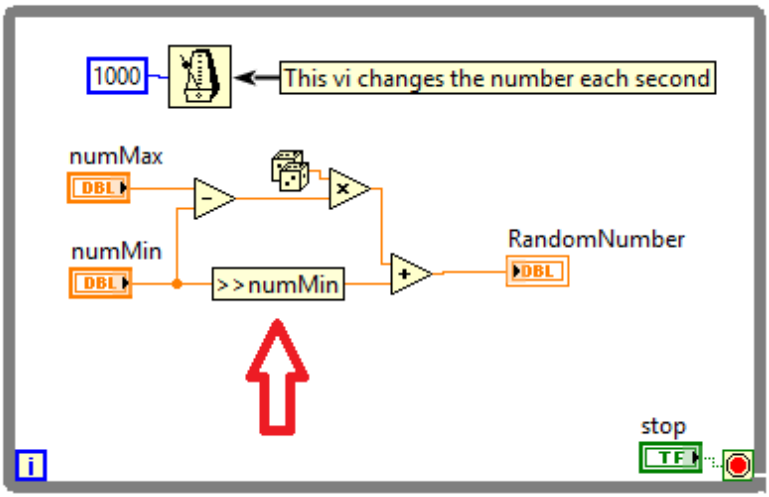


17. En algunas ocasiones cuando los códigos son tan grandes que ocupan varias pantallas, resulta útil sobre los cables escribir el tipo de dato que va por dicho cable, como ejemplo observa como antes de la operación suma agregue una nota usando también la opción label del paso 15 de esta guía y posicionándola sobre el cable.

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

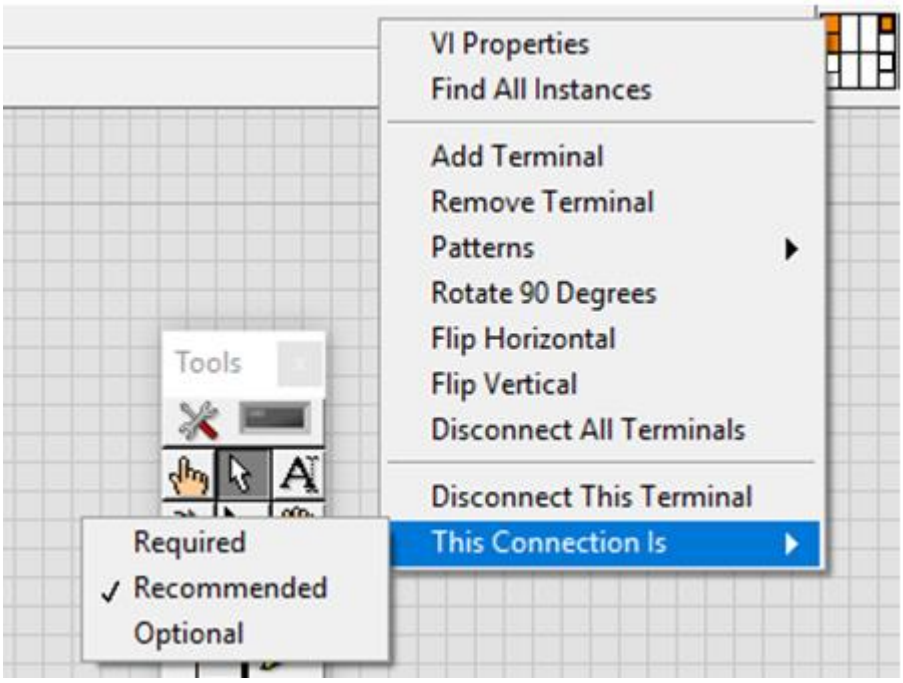
This equation is used to calculate a number between an upper and a lower limit, in this VI the number changes each second due the wait funcion 1000mS = 1seg

$$RandomNumber = (numMax - numMin) * (randomNumbert(0-1)) + numMin \quad \text{(Eq.1)}$$



18. Para culminar esta primera sección solo hace falta determinar si las entradas y salidas del VI son requeridas, recomendadas u opcionales, por default todas son recomendadas y las dejaremos así.

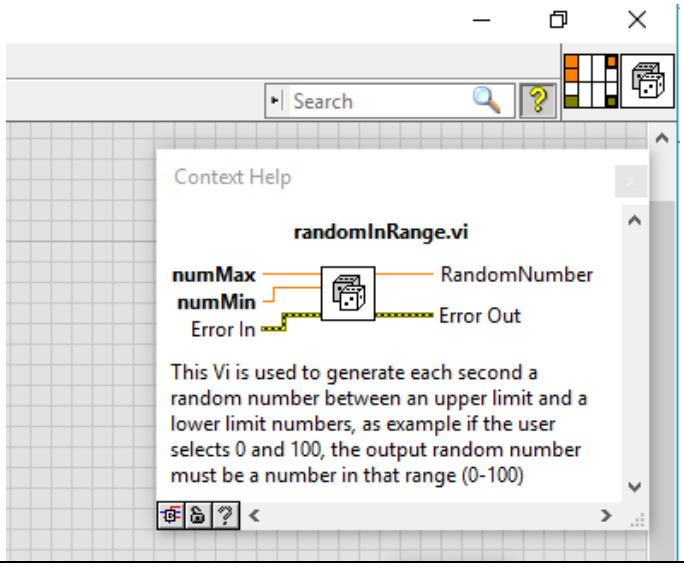
Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



Ten en cuenta que si la entrada o salida es requerida y no se conecta en el vi de destino, el SubVi arrojará un error (Conocer esto es importante, esto es a veces un punto de examen CLAD).

Tú como desarrollador y de acuerdo a la aplicación, eres quien determina los tipos de entrada o salida que deben tener los subVi (Requerida, recomendada u opcional).

Cuando las entradas son requeridas, el nombre de la entrada o salida se ve en negrilla dentro del panel de ayuda, como ejemplo mira cómo se vería si las entradas numMin y numMax fueran requeridas.



Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

RESUMEN PARTE I DOCUMENTACIÓN

Documenta muy bien tus Vis, no importa que tan chico o grande sea, uno de los obstáculos más grandes que he encontrado en mi experiencia trabajando con Labview como desarrollador y arquitecto, se encuentra en la pobre documentación que algunos desarrolladores llevan a cabo con sus programas… aveces es más fácil rehacer todo que entender todo lo que otro desarrollador hizo y no documento, no caigas en esta pésima practica de programación, por esa razón documenta muy bien tus programas☹.

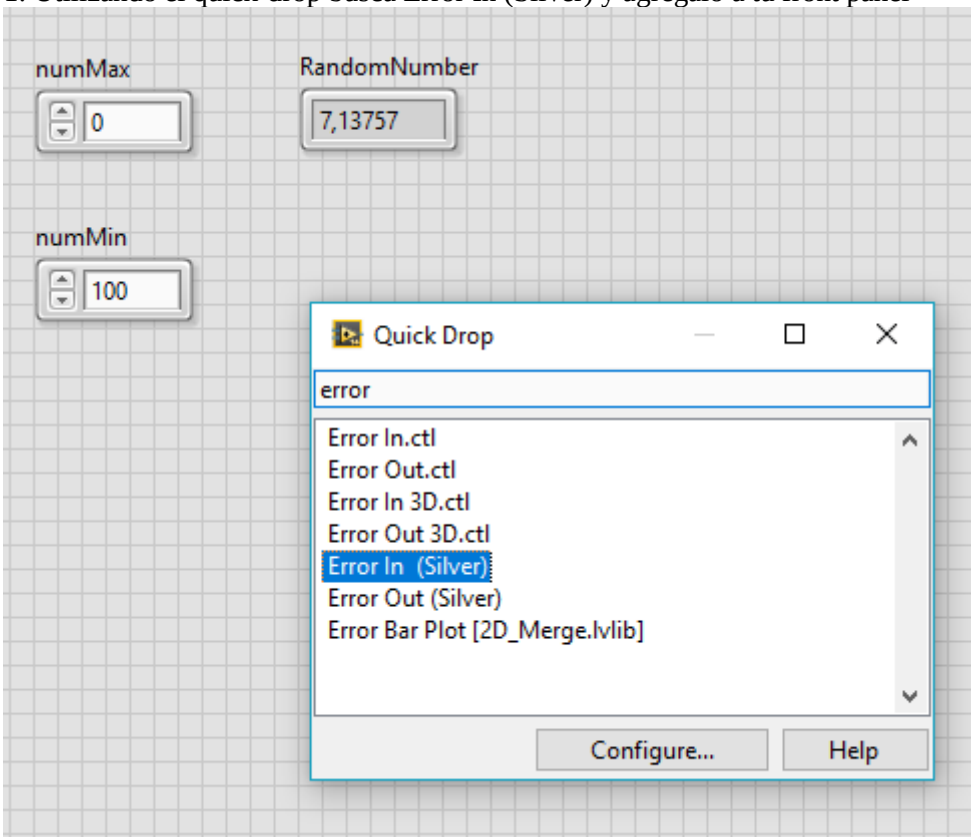
Desarrollo Parte II

Ahora que ya aprendiste de documentación de front-panels y block diagrams, es hora de introducir un nuevo elemento a tus competencias de programador Labview… la gestión de errores.

Vamos a agregar agregar dos importantes ítems que todo VI y Subvi debe tener: cluster de errores.

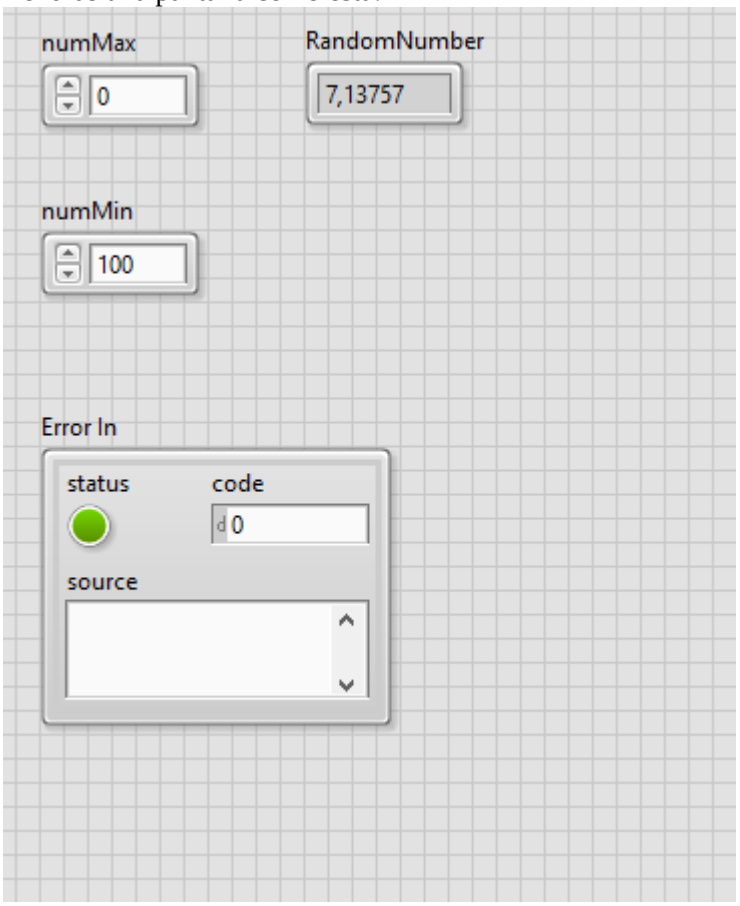
Vamos.

1. Utilizando el quick-drop busca Error In (Silver) y agrégalo a tu front panel



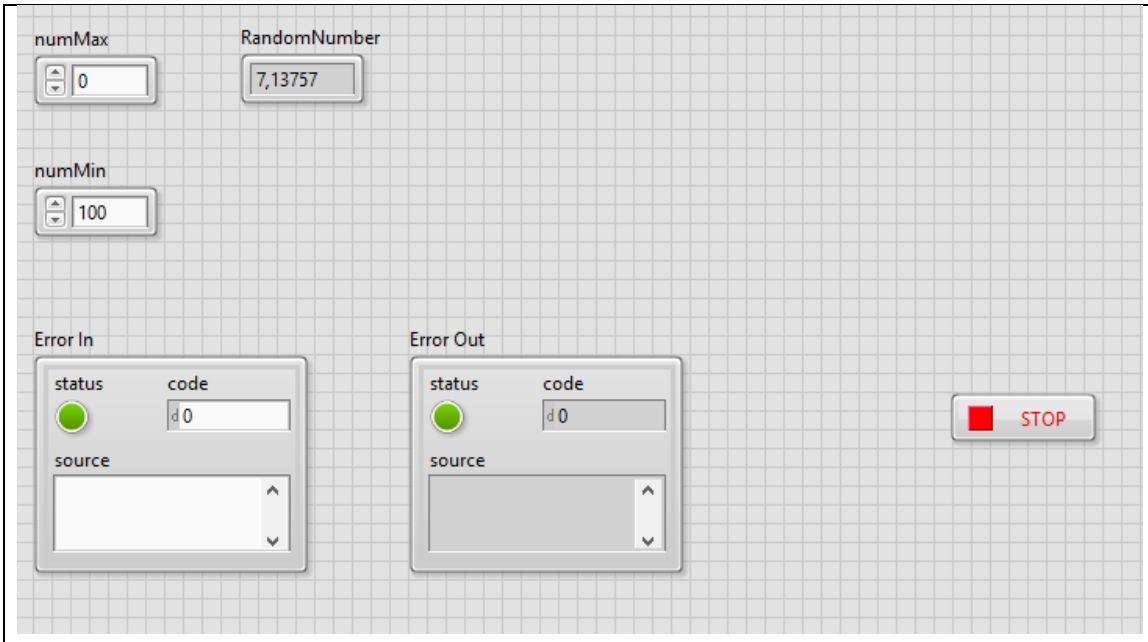
Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

Tendras una pantalla como esta:



2. Realiza una nueva búsqueda con Quick-Drop y ahora agrega Error Out (Silver), a la final tu panel se va convirtiendo en algo como lo siguiente:

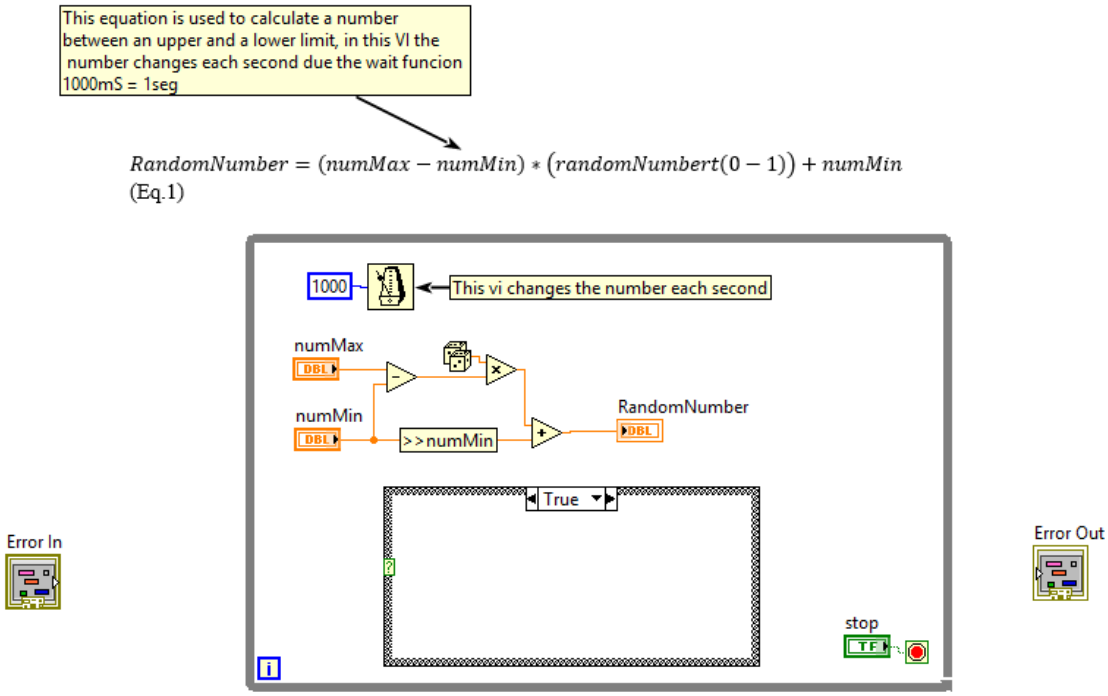
Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



Los cluster de error son una importante herramienta que los desarrolladores de software como tu y yo utilizamos para gestionar errores e identificar cuando las cosas no van bien con el código, ahora solo basta con modificar el block diagram agregando la gestión, para ello haremos uso de una estructura Case con dos casos (Cuando hay Error y Cuando no hay Error), ve al block diagram.

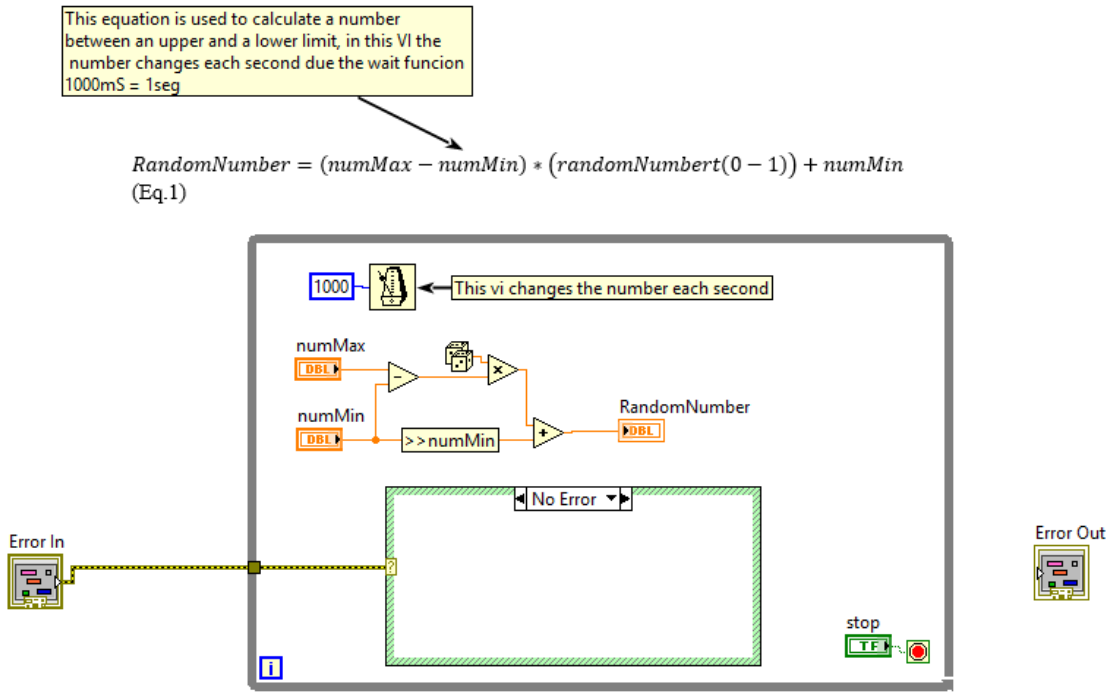
3. Agrega una estructura Case al block diagram

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



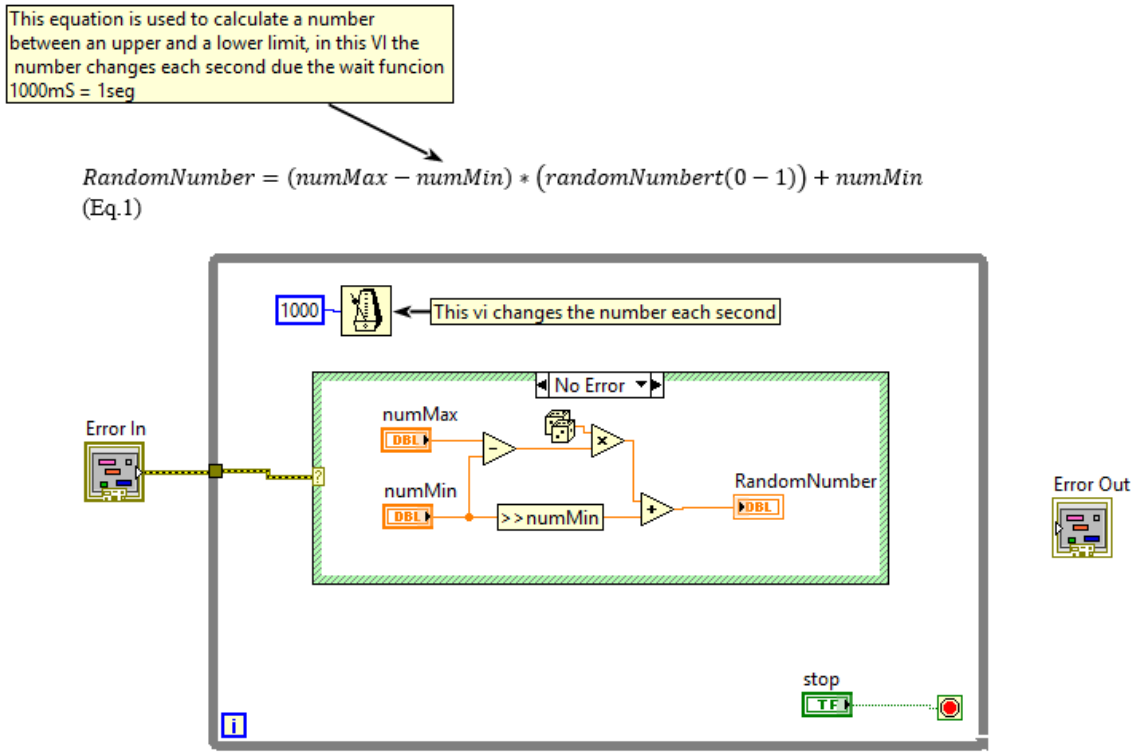
4. Ahora cablea la entrada del error in al selector del case, automáticamente se generan los casos error and No error, asi:

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



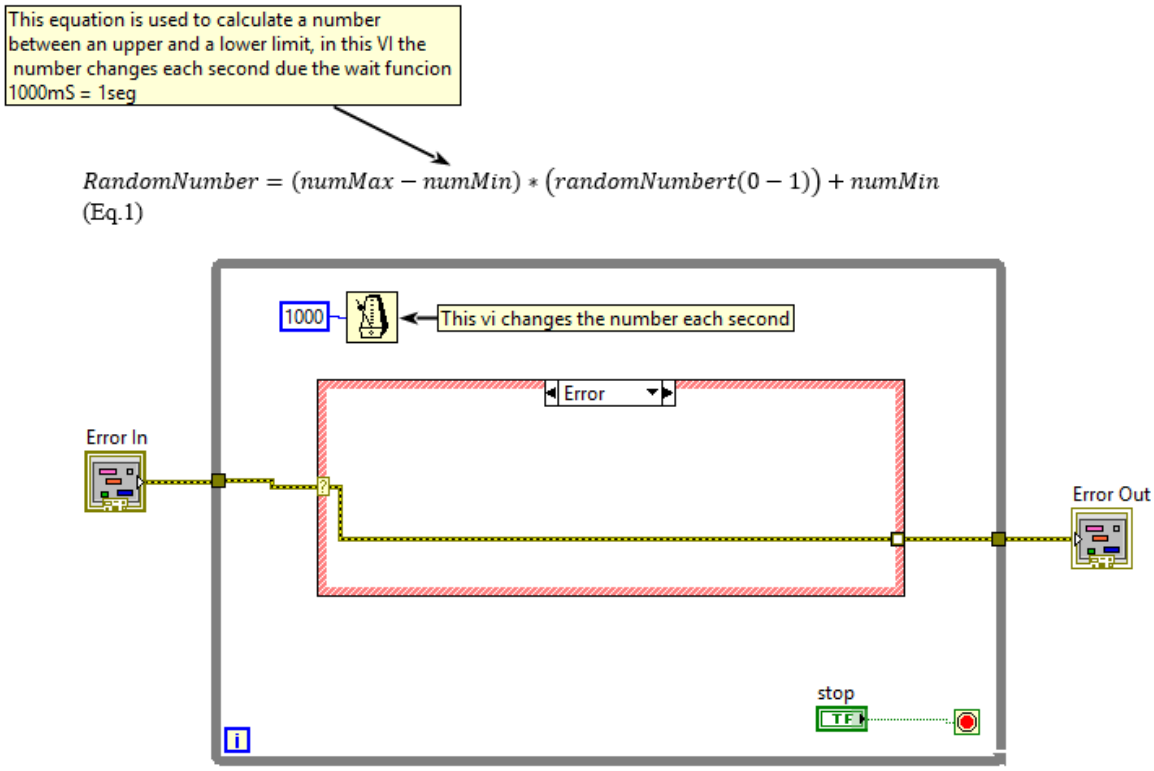
5. Incorpora todo tu código dentro del caso “No Error” asi:

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



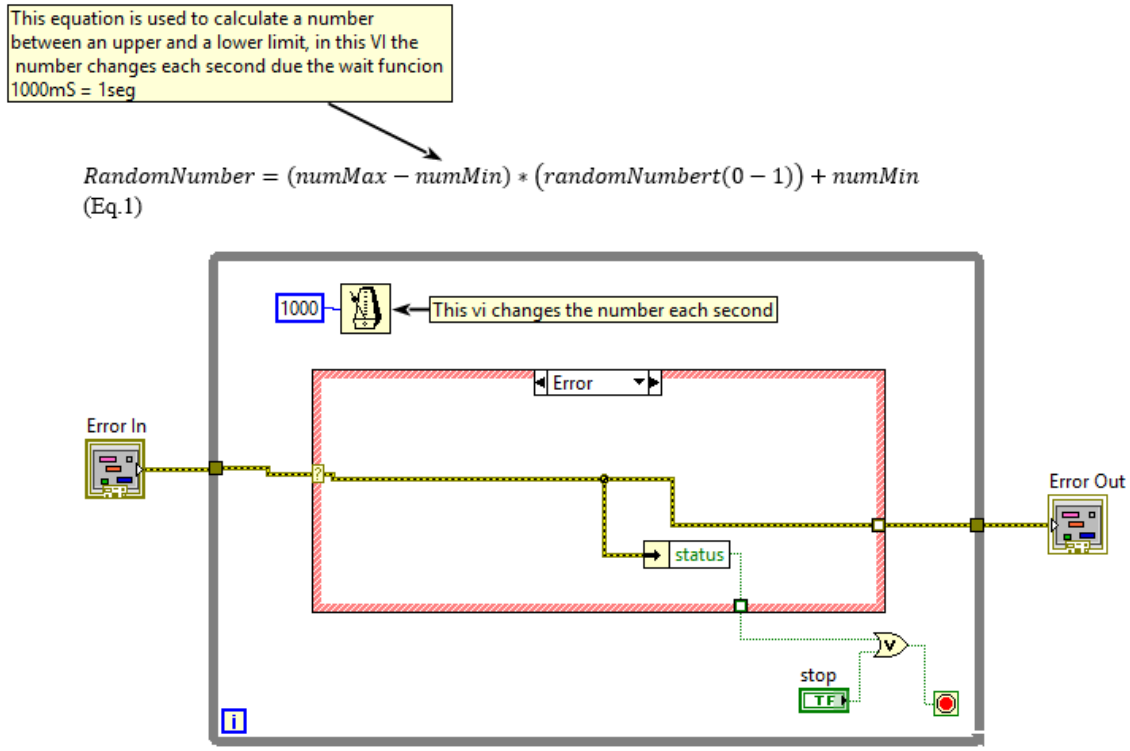
6. Ahora ve al siguiente caso (Error), cablea el error in al error out asi:

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



7. De tus clases teóricas, recuerda que el cluster de error tiene un valor booleano… aquí lo usaremos para parar en caso de error el vi y no quede en un loop infinito, para ello usa unbundle by name y realiza una OR con la salida de stop asi:

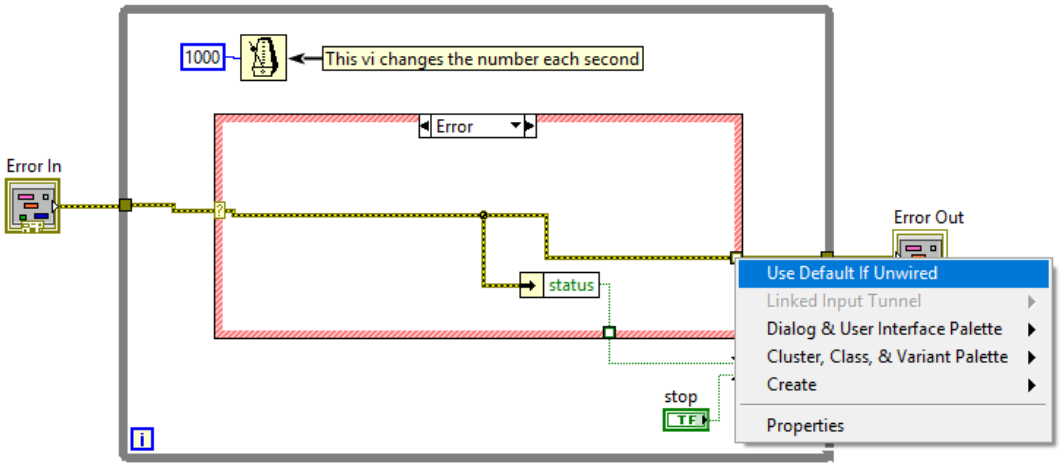
Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



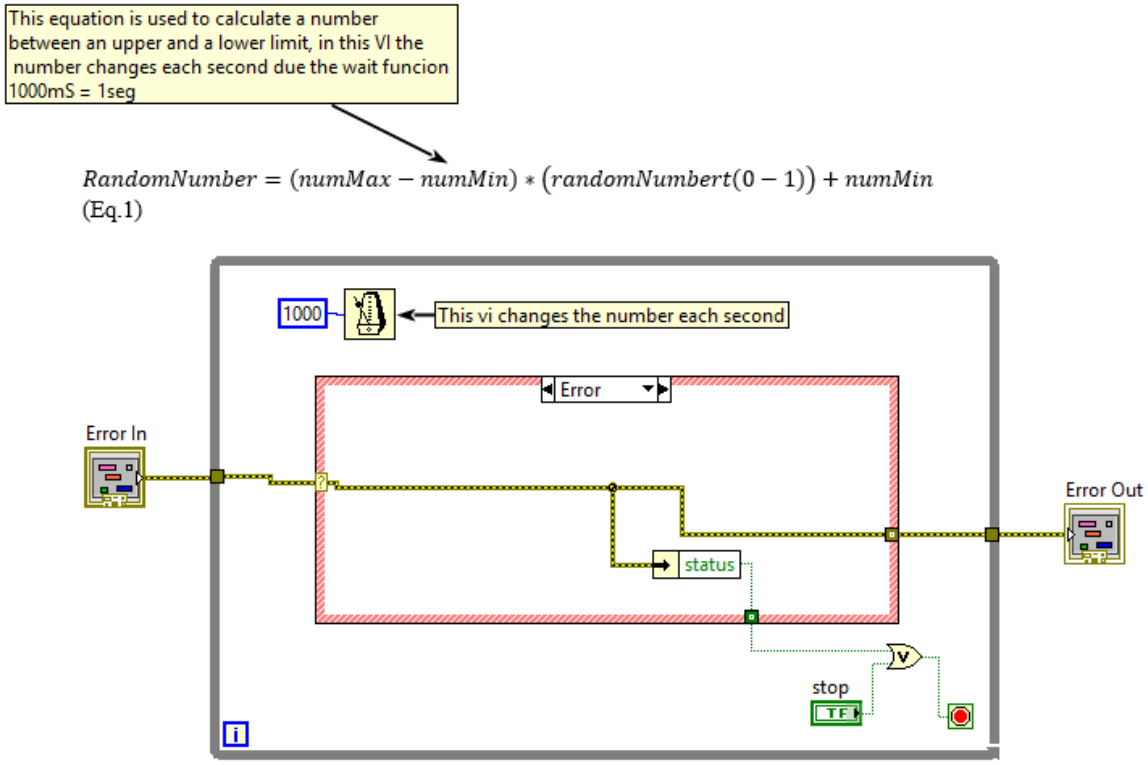
Con este código paras tu vi en caso de un error inesperado.

8. Dale click derecho en las terminales de túnel que aparecen en cuadro rojo y selecciona la opción>> Use default if unwired

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)



9. Repite y asegúrate que los dos cuadros rojos queden con default if unwired asi:



10 . Has culminado la gestión de errores de Vis, TODO vi Debe contener este tipo de gestión, recuerda tu trabajaras en grandes proyectos y equipos de ingeniería y de investigación, el trabajar

Desarrollado por: Msc. Carlos Moreno – Freiburg Mechatronik (freiburg.mechatronik@gmail.com)

con esta técnica es parte de la certificación CLAD y todo Clad mínimo debe conocer como se trabaja y para que sirven este tipo de herramientas para manejo de errores.

11.Cablea en el icon conector pane los cluster de entrada y salida de errores, los veras de color marrón por ser clusters, asi:

12. Por ultimo para convertir tu código en un subvi elimina el botón de stop y conecta al terminal de parada una contante true… recuerda que el while se ejecuta al menos una vez…

13. Haz culminado tu práctica, ya tienes creado tu subVi!!!!

Fin de la lección.