CAPÍTULO 4
DESARROLLO DEL SISTEMA DE MEDICIÓN

Los sistemas de tiempo real deben asegurar que sus respuestas ocurran instantáneamente. Con los sistemas operativos de propósito general no es posible asegurar que las respuestas ocurran dentro de un período de tiempo establecido, y el procesamiento puede terminar mucho más tarde de lo esperado.

Para realizar una programación adecuada de un sistema de tiempo real deben revisarse los siguientes conceptos esenciales:

1 Tiempo de ciclo

Muchas aplicaciones que requieren un funcionamiento en tiempo real son cíclicas. El tiempo transcurrido entre cada inicio de ciclo es el tiempo de ciclo o de lazo.

2 Fluctuación (*Jitter)*

Aún cuando se trabaja en sistemas operativos de tiempo real, puede existir una ligera variación entre el tiempo de un ciclo y otro. A la variación entre el tiempo de ciclo deseado y el tiempo de ciclo real se le conoce como fluctuación o *jitter.*

3 Determinismo

A la medida de la cantidad de fluctuación se le denomina determinismo. Éste indica qué tan confiable responde un sistema de tiempo real ante eventos externos o cuán confiable es para realizar operaciones dentro de un tiempo límite.

El determinismo es una característica fundamental de los sistemas de tiempo real, y garantiza que los cálculos y operaciones se realicen dentro de un tiempo límite, lo cual conlleva a que los resultados terminen siempre a tiempo.

4 Latencia

La latencia es el tiempo que se requiere para responder a un evento, o bien, el tiempo transcurrido de la entrada a la salida. Un sistema de tiempo real debe dar una respuesta en tiempo real que garantice el menor valor de latencia.

5 Prioridad

La prioridad es una característica que define cuándo un VI o ciclo debe ejecutarse con preferencia sobre otros. Esta prioridad puede ser configurada, y las acciones de control deben tener la mayor prioridad.

4.1 Consideraciones para programación en tiempo real

Para que un sistema pueda ser considerado de tiempo real, todos los elementos que participan en el sistema deben responder en tiempo real. Una aplicación que corre en un sistema operativo de tiempo real podría no tener un comportamiento característico de tiempo real si se encuentra asociada con alguna otra aplicación que no se comporte de esta manera.

Todo sistema de tiempo real que cumpla con determinismo debe encontrarse soportado por dispositivos físicos de respuesta inmediata. Cuando el hardware es capaz de realizar la adquisición y procesamiento en el tiempo deseado, se deben tomar en cuenta las siguientes consideraciones en la programación para conseguir un determinismo adecuado:

- Evitar conflictos de memoria
- Evitar el uso de los llamados VI Express
- Evitar la sobredemanda de subVIs
- Deshabilitar propiedades no esenciales de Vis.

Además, dado que el acceso a ciertas estructuras de datos, bibliotecas y variables sólo puede darse de manera serial, debe minimizarse el uso de recursos compartidos tales como variables globales, variables compartidas, códigos de comunicación en red, subVIs no reentrantes y la lectura/escritura de archivos, ya que pueden reducir el determinismo del sistema.

4.2 Arquitectura general del sistema de medición

La arquitectura general de los sistemas de tiempo real consta básicamente de tres componentes principales: un lazo o ciclo de alta prioridad o de tiempo crítico, un lazo de prioridad normal y una PC servidora de red, como se observa en la Figura 4.1.

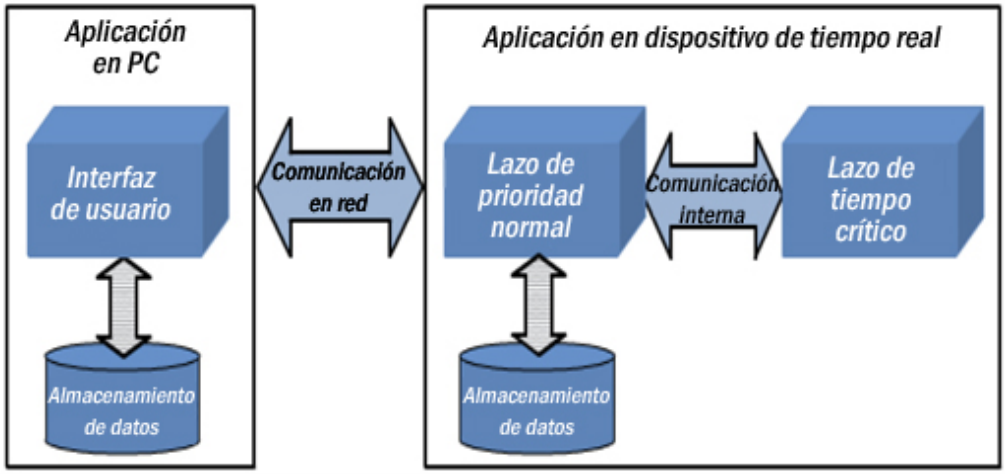


Figura 4.1 Arquitectura general de un sistema de tiempo real.

- El lazo de tiempo crítico contiene las funciones de alta prioridad como son la lectura y escritura de entradas y salidas, el procesamiento de señales, el control y la toma de decisiones punto a punto.

- En el lazo de prioridad normal se incluyen los procesos de almacenamiento de datos y la comunicación en red.

- La PC servidora en red permite la interfaz gráfica con el usuario de modo remoto, así como el desarrollo de una etapa de postprocesamiento y almacenamiento de datos adicional.

Dado que el sistema de adquisición, medición, y análisis presentado se desarrolla en un CompactRIO, es posible realizar la programación de manera que el lazo de tiempo crítico sea ejecutado en el FPGA, el cual tiene una mayor velocidad de procesamiento que el controlador, mientras que el lazo de prioridad normal se ejecuta desde el controlador. Esta separación favorece de manera importante el determinismo del sistema. La arquitectura del sistema desarrollado se ilustra en la Figura 4.2.

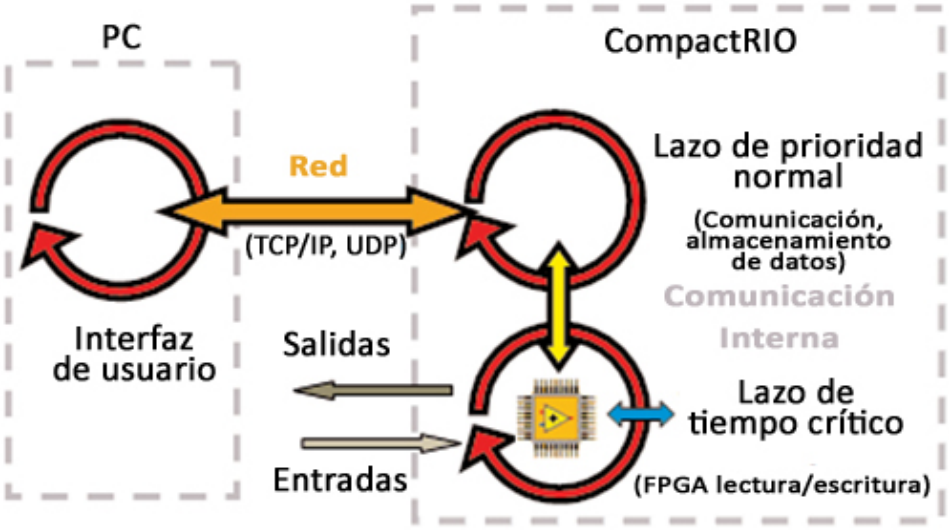


Figura 4.2 Arquitectura del sistema de adquisición, medición y análisis.

Para la construcción de las diferentes partes del sistema bajo la arquitectura mencionada se generó un proyecto en LabVIEW con el nombre de *STC-VITE* (*Sistema de Transporte Colectivo – Vibraciones y Tensiones*), y puesto que la adquisición pertenece al lazo de tiempo crítico, los módulos de adquisición que se utilizan en el sistema se configuraron bajo la modalidad de Interfaz FPGA de LabVIEW (*LabVIEW FPGA Interface*).

En la Figura 4.3 se muestra el árbol del proyecto. En éste se consideran dos módulos NI-9233 con los cuales se pueden conectar hasta ocho acelerómetros. Las entradas del módulo NI-9219 se configuran de manera que los cuatro canales realicen mediciones de tensión con un rango de ±60 V de CD.

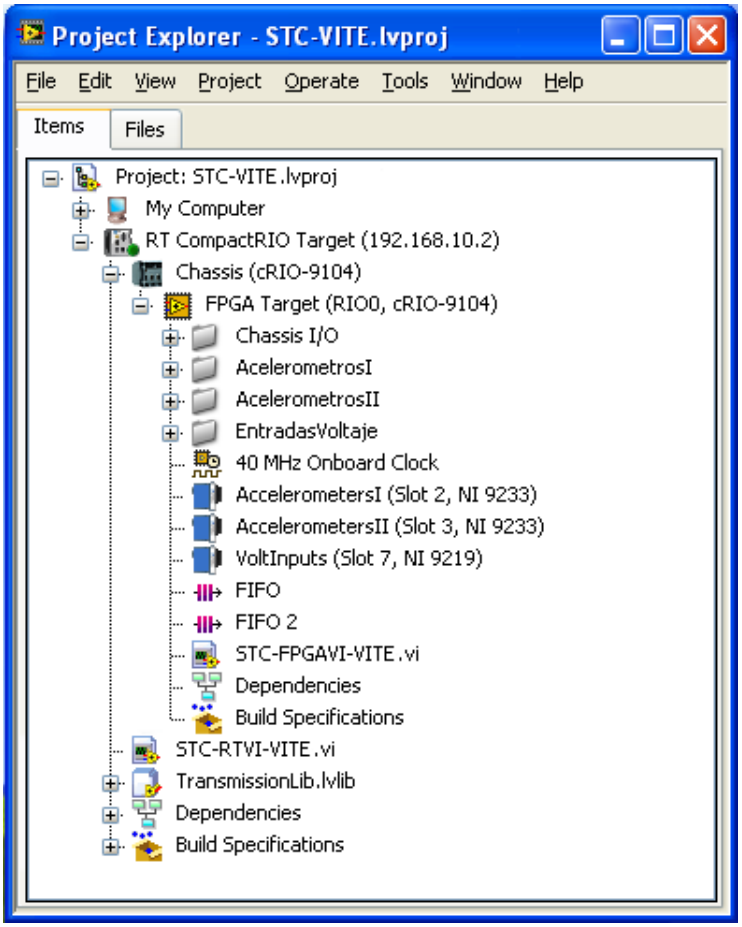


Figura 4.3 Árbol del proyecto en LabVIEW.

Al configurar los módulos bajo el modo *LabVIEW FPGA Interface,* estos aparecen desplegados en el cuerpo del proyecto dentro de la rama *FPGA Target*. Para una fácil identificación fueron renombrados como *AccelerometersI, AccelerometersII* y *VoltInputs.*

De manera automática se crean los nodos pertenecientes a cada módulo para que puedan utilizarse en el código del programa. En la Figura 4.4 se muestran los nodos desglosados

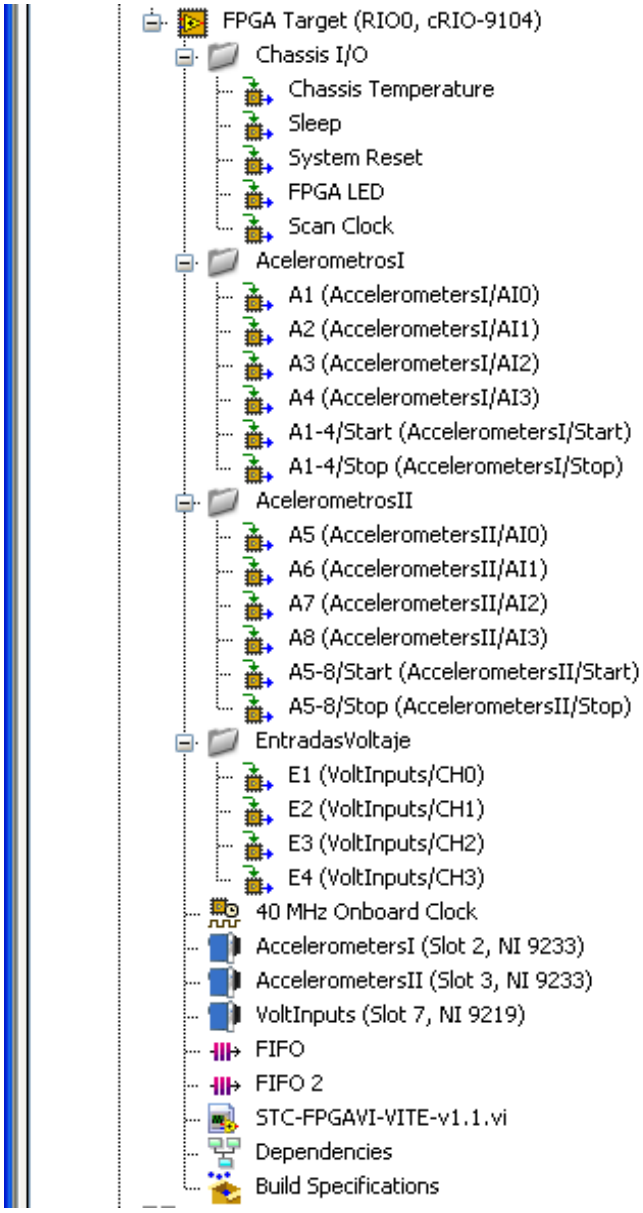


Figura 4.4 FPGA con desglose de nodos.

Se nombraron los nodos de los módulos NI-9233 como *AcelerómetrosI* y *AcelerómetrosII*, y a las variables de cada uno de ellos como A1, A2, A3 y A4 para el primero, y A5, A6, A7 y A8 para el segundo. El nodo del módulo NI-9219 se nombró como *EntradasVoltaje* y sus variables con los nombres desde E1 hasta E4.

El nodo *Chassis I/O* contiene variables para el manejo del FPGA. En este desarrollo la variable que resultó de utilidad fue *FPGA LED*, con la cual se puede manipular el led de la cara frontal del controlador.

El *FPGA Target* también crea una base de tiempo de 40 MHz, con la cual se consigue procesamiento de gran velocidad.

Para transferir al controlador los datos adquiridos por los módulos, dentro del *FPGA Target* se crearon dos arreglos de acceso dinámico de memoria (DMA) de tipo primera entrada primera salida (FIFO) con modalidad *Target to Host*. Uno de ellos transfiere la información recabada por los módulos NI-9233 y el otro transfiere la información recabada por el módulo NI-9219.

Estos DMA FIFOs reservan localidades de memoria del FPGA para realizar la transferencia de datos hacia el controlador. Con ello se evita el empleo de recursos compartidos para esta tarea y se favorece el determinismo en la etapa de adquisición.

La velocidad de adquisición de datos y llenado de los DMA FIFOs por parte del FPGA está en función de la máxima tasa de muestreo de los módulos. Por otra parte, la extracción de los datos de los DMA FIFOs por parte del controlador no depende de la tasa de muestreo de los módulos, y sí de la carga de trabajo que presente el controlador.

Por lo anterior, un ciclo en el FPGA puede ejecutarse varias veces antes de que culmine un ciclo del controlador. Si los DMA FIFOs no cuentan con capacidad suficiente para almacenar todos los datos adquiridos antes de que sean extraídos por el controlador, el sistema registrará pérdidas importantes de datos. Así, el tamaño de los DMA FIFOs debe ser configurado de manera que sea lo suficientemente grande para almacenar todos los datos previos a la extracción por parte del controlador, pero debe ser lo suficientemente pequeño para usar el mínimo número de recursos del FPGA.

La siguiente expresión muestra la relación entre la tasa de muestreo y el tiempo de descarga, para evitar que se sature el DMA FIFO.

$$t_{max} = \frac{D}{R \times N}$$

Donde

- t_{max} es el tiempo máximo para vaciar el DMA FIFO
- D es el tamaño del DMA FIFO
- R es la taza de muestreo
- N es el número de canales

La Tabla 4.1 muestra el tiempo que tardan en llenarse diferentes tamaños de DMA FIFOs cuando se realiza una adquisición con los 16 canales disponibles de los módulos NI-9233 con la tasa de muestreo máxima permitida por este módulo.

Tabla 4.1 Tiempo de llenado de DMA FIFO con la tasa de muestreo máxima.

Muestreo por segundo por canal	*Muestreo por segundo por 16 canales*	*Tamaño de DMA FIFO*	*Tiempo máximo en milisegundos para vaciar DMA FIFO*
50 000	800 000	15	0.018750
50 000	800 000	31	0.038750
50 000	800 000	63	0.078750
50 000	800 000	127	0.158750
50 000	800 000	255	0.318750
50 000	800 000	511	0.638750
50 000	800 000	1023	1.278750
50 000	800 000	2047	2.558750
50 000	800 000	4095	5.118750
50 000	800 000	8191	10.238750
50 000	800 000	16 383	20.478750
50 000	800 000	32 767	40.958750
50 000	800 000	65 535	81.918750
50 000	800 000	131 071	163.838750
50 000	800 000	262 143	327.678750
50 000	800 000	524 287	655.358750
50 000	800 000	1 048 575	1310.718750
50 000	800 000	2 097 151	2621.438750
50 000	800 000	4 194 303	5242.878750
50 000	800 000	8 388 607	10 485.758750

El tiempo de ciclo mínimo de un lazo en un VI que corre en el controlador es de 1 ms; sin embargo, ya que el tiempo real depende del código contenido en el lazo, usualmente la duración del lazo que incluye tareas como las que se van a realizar no es menor a 5 ms.

Para evitar una saturación de los DMA FIFOs que pueda llevar a pérdidas de información, en el sistema presentado al DMA FIFO que almacena los datos adquiridos por los módulos NI-9233 se le asignó un tamaño de 32 767.

Con estos valores, el controlador tiene hasta 40 ms para extraer la totalidad de los datos del DMA FIFO con una tasa de muestreo máxima.

Al DMA FIFO que almacena los datos recabados por el módulo NI-9219 se le respetó el valor por defecto de 1023, ya que por la baja tasa fija de muestreo del módulo no presenta el mismo riesgo de saturación.

El código para la adquisición de datos y su transferencia al controlador es programado en un VI específico para el *FPGA Target*, al cual se le dio el nombre de *STC-FPGAVI-VITE*.

La etapa de registro y análisis se monta directamente sobre el controlador, fuera del *FPGA Target*. Al VI que se encarga de estas tareas se le nombró *STC-RTVI-VITE* y funge como programa principal, ya que recibe la información proveniente del FPGA, la almacena ya sea en la memoria interna del mismo CompactRIO o bien en un dispositivo USB externo, y la publica para que pueda ser monitoreada y analizada en tiempo real.

Para que sea posible el monitoreo, fue necesario incorporar al proyecto una biblioteca de variables compartidas. Esta biblioteca se instala también en el controlador, ya que de hacerlo en el chasis se vería afectado el determinismo del sistema. Se le asignó el nombre de *TransmissionLib.*

4.3 Descripción del programa

Como se ya se mencionó, el programa principal encargado de almacenar, analizar y publicar los datos para su monitoreo es *STC-RTVI-VITE*. Sin embargo, se explicará inicialmente el programa contenido en el FPGA para llevar la secuencia del flujo de las señales adquiridas, y posteriormente se detallará el funcionamiento del programa principal.

4.3.1 STC-FPGAVI-VITE

El *STC-FPGAVI-VITE* es el VI que se encarga de la adquisición de las señales de vibraciones y tensiones. Consta de 6 etapas que se ejecutan de manera secuencial.

En la primera etapa se lee la información de calibración contenida en cada uno de los módulos NI-9233. Los parámetros que se registran son la ponderación del bit menos significativo (*LSB Weight*) y el *Offset*.

Los valores registrados se almacenan en arreglos. A los valores de calibración del primer módulo se le llamó Calibración 1, y los valores de calibración del segundo módulo se almacenaron bajo el nombre de Calibración 2. Esto se observa en el código de la Figura 4.5.

Como posteriormente se verá, estos valores de calibración se utilizan para conseguir una correcta reconstrucción de las señales adquiridas, y con ello tener un monitoreo y análisis confiable.

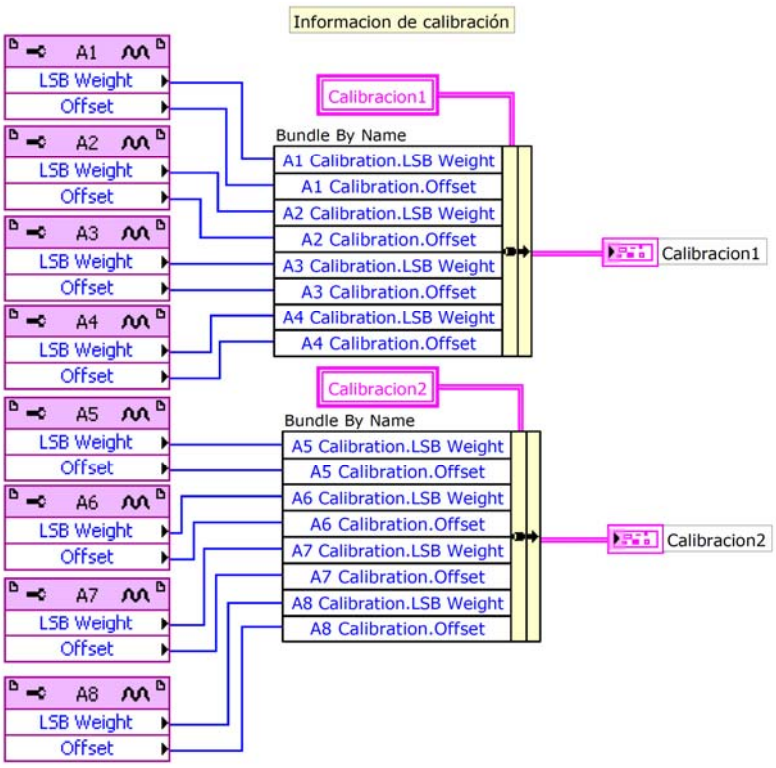


Figura 4.5 Información de calibración.

En la segunda etapa se genera una interrupción con el fin de sincronizar la ejecución de los VIs *STC-RTVI-VITE* y *STC-FPGAVI-VITE*. En este paso el programa se detiene hasta que el *STC-RTVI-VITE* informe al FPGA que está listo para continuar. La instrucción en el código que inserta dicha interrupción se muestra en la Figura 4.6.

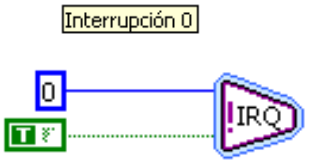


Figura 4.6 Inserción de interrupción.

En la tercera etapa se asigna la configuración programática para los módulos. A cada canal del módulo NI-9219 se le asigna el rango de valores de tensión que pueden medir. A cada módulo NI-9233 se le asigna la máxima tasa de muestreo con la que pueden adquirir las señales de vibración. La reducción de esta taza de muestreo podrá ser modificada por el usuario desde el *STC-RTVI-VITE.*

El código en LabVIEW para la tercera etapa se muestra en la Figura 4.7.

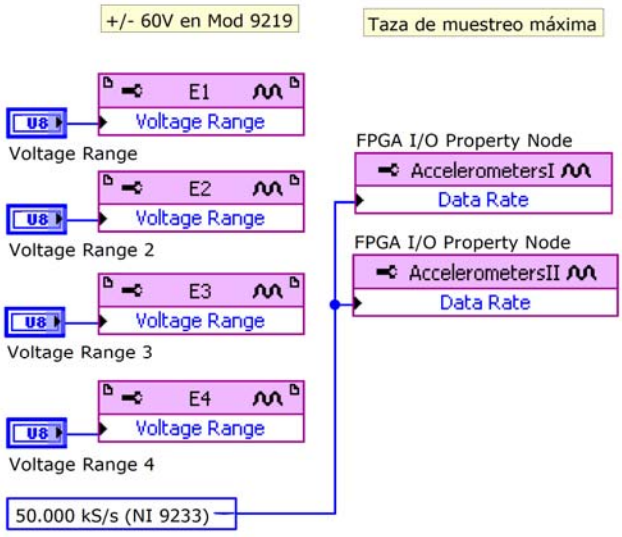


Figura 4.7 Asignación de propiedades para los módulos NI-9219 y NI-9233.

El código para la cuarta etapa se muestra en la Figura 4.8. En esta etapa se habilita la adquisición de los módulos NI-9233 y se inicializa la variable e indicadores booleanos *Exit, FPGA Error V* y *FPGA Error T*.

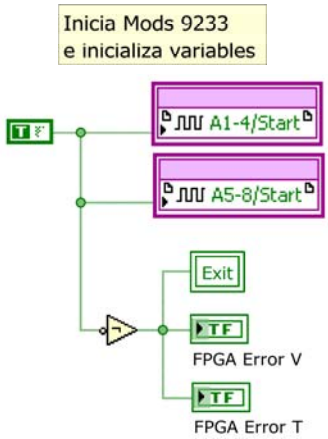


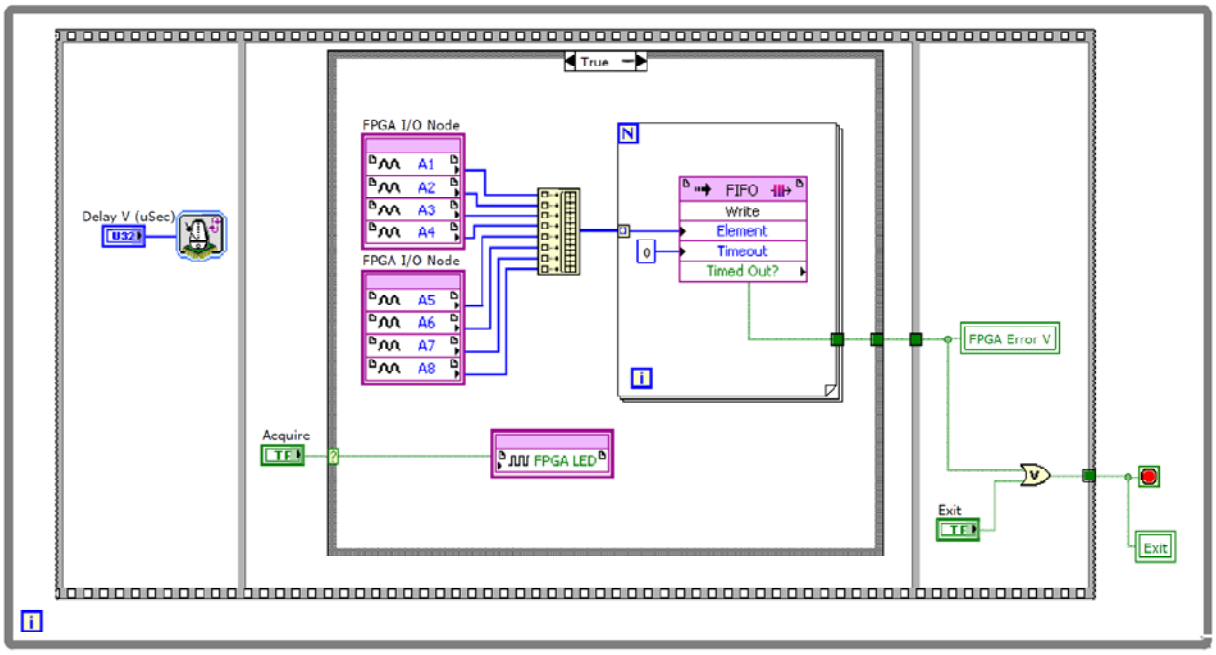
Figura 4.8 Habilitación de módulos NI-9233 e inicialización de variables.

En la quinta etapa es donde se lleva a cabo la adquisición. Consta de un par de lazos que se ejecutan en paralelo. El primero de ellos se encarga de realizar la adquisición de vibraciones de los dos módulos NI-9233 activados. El segundo se encarga de la adquisición de tensiones del módulo NI-9211.

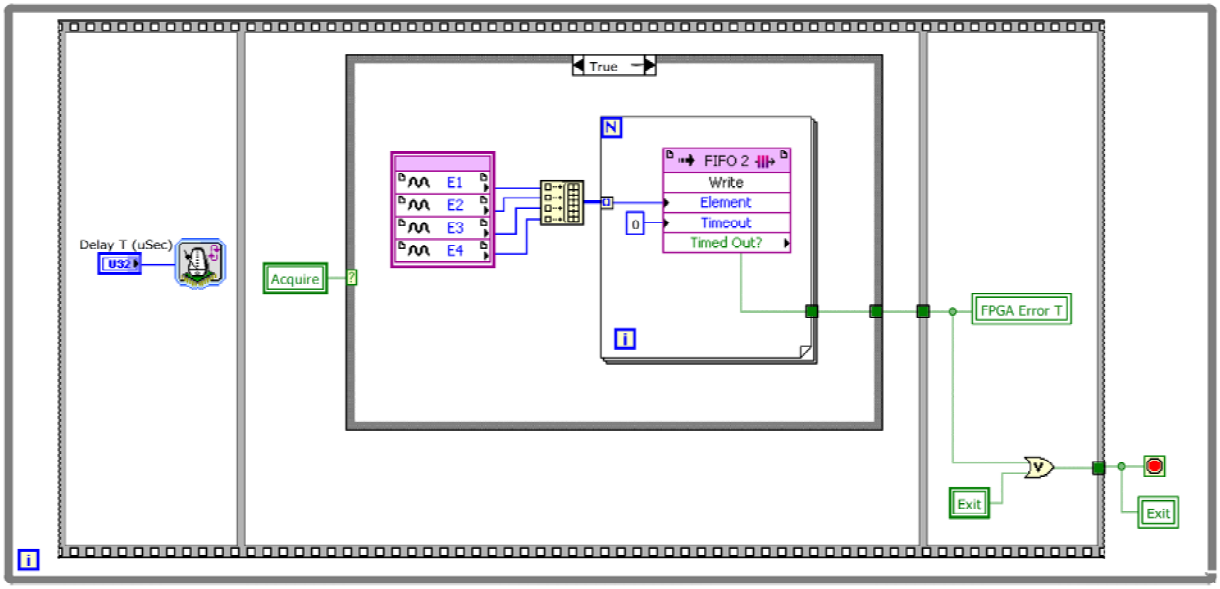
Ambos lazos contienen los mismos elementos y presentan el mismo funcionamiento. La información de las señales adquiridas se envía al DMA FIFO para que el *STC-RTVI-VITE* pueda acceder a ella sin afectar el determinismo del sistema.

El código consta de dos estructuras secuenciales de tres etapas, o secuencias, que se ejecutan cíclicamente como se ilustra en la Figura 4.9, y se explican a continuación.

a)



b)



c)

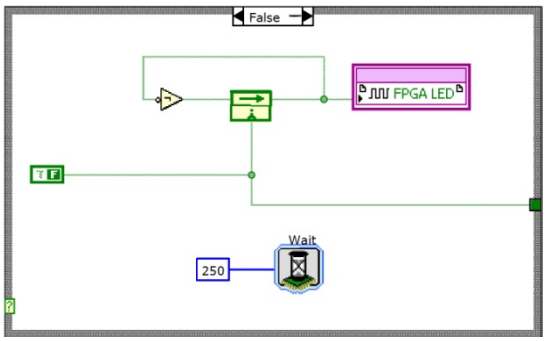


Figura 4.9 Lazos de adquisición de a) vibraciones, b) tensiones, c) caso falso.

- En la primera secuencia se asigna el tiempo de ciclo deseado en microsegundos.
- La segunda secuencia está compuesta por una estructura de toma de decisión con dos posibles casos de adquisición: verdadero y falso. En el caso falso, se hace parpadear el led FPGA por medio de la variable FPGA LED. Esto significará que el sistema se encuentra listo para realizar la adquisición de datos. En el caso verdadero se realiza la adquisición de las señales, se juntan en un arreglo y se envían al DMA FIFO. La elección del caso a ejecutar será determinada por el *STC-RTVI-VITE*.
- En la tercera secuencia se actualiza el valor del indicador *FPGA Error*. Este indicador se enciende cuando el DMA FIFO se satura y detiene la ejecución. La ejecución de ambos lazos también se detiene si la variable booleana *Exit* es verdadera. Cuando se detiene la ejecución de los lazos finaliza la etapa 5.

La sexta y última etapa deshabilita los módulos NI-9233, y finaliza la adquisición de vibraciones. El código para esta etapa es el mostrado en la Figura 4.10.

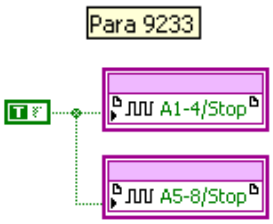


Figura 4.10 Deshabilitación de módulos NI-9233.

Este código debe ser compilado antes de ser instalado y ejecutado en el *FPGA Target* del CompactRIO. LabVIEW incorpora el compilador que traduce el código en lenguaje G a código VHDL. Para realizar la compilación se debe seleccionar la opción *Compile* al dar clic derecho sobre el *STC-FPGAVI-VITE* en el árbol del proyecto.

Una vez que concluye la compilación, se genera un archivo de bits (*bitfile*) en la carpeta donde se encuentra el programa *STC-FPGAVI-VITE*. Para instalarlo en el CompactRIO se da clic derecho sobre el *FPGA Target* en el árbol del proyecto y se selecciona *RIO Device Setup…*

En la ventana emergente, que se muestra en la Figura 4.11, se presiona el botón *Erase Bitfile on Flash* para limpiar la memoria del FPGA. Posteriormente en la sección *Bitfile to Download* se selecciona el archivo de bits creado en la compilación y se instala en el CompactRIO al presionar el botón *Download Bitfile*.

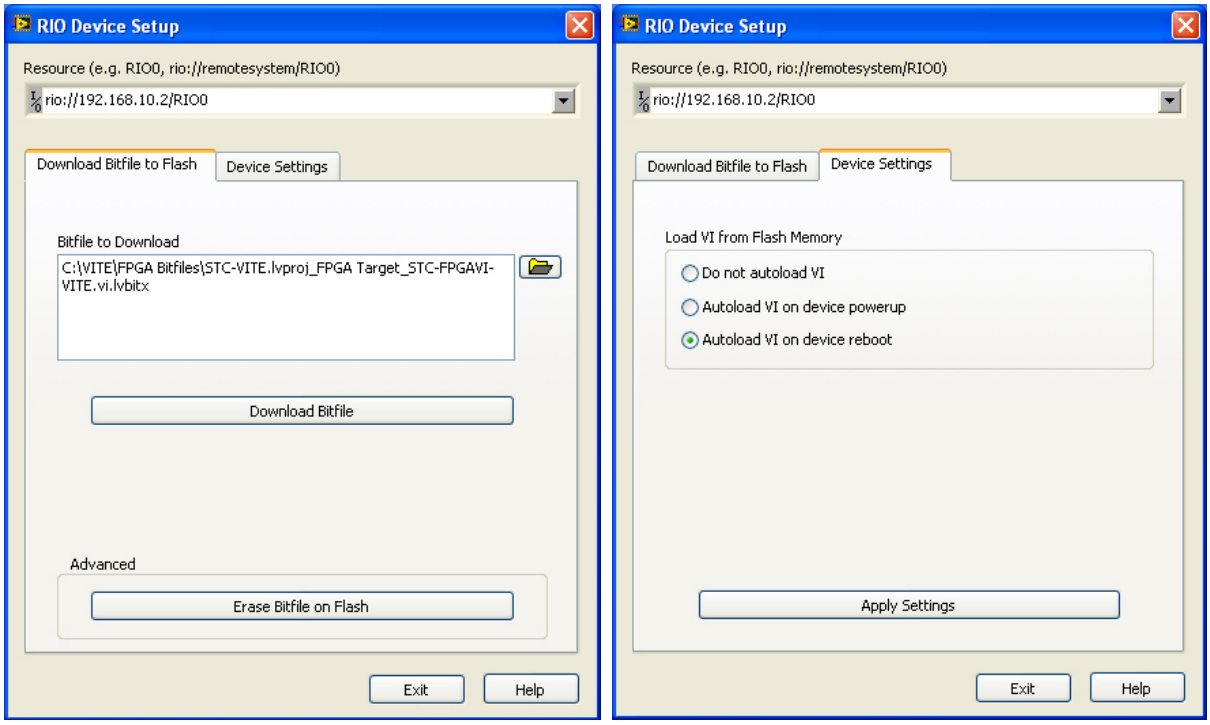


Figura 4.11 Pestañas de la ventana RIO Device Setup.

En la pestaña *Device Settings* se debe seleccionar la opción *Autoload VI on device reboot* y presionar el botón *Apply Settings*.

Cualquier cambio en la configuración de los elementos dentro del *FPGA Target*, como el tipo de señal que adquiere el módulo NI-9219, o cualquier cambio en el código del programa *STC-FPGAVI-VITE*, por mínimo que este sea, y que no se dé de manera programática, exige una re-compilación.

4.3.2 STC-RTVI-VITE

La arquitectura del *STC-RTVI-VITE* consta de tres etapas secuenciales:

1 Inicialización
2 Operación
3 Finalización

4.3.2.1 Inicialización

En esta etapa se ejecuta una rutina de inicialización con la cual prepara al controlador para su operación. Como se muestra en el código de la Figura 4.12, se abre una referencia al *FPGA Target* y envía al *STC-FPGAVI-VITE* una confirmación de que el *STC-RTVI-VITE* se encuentra listo para la operación, con lo que la interrupción en el FPGA se limpia y ambos programas se ejecutan paralela y sincronizadamente. En esta parte se establecen los parámetros para la adquisición (como tasas de muestreo y rangos de tensión) mediante los nodos que comunican a estos dos programas.

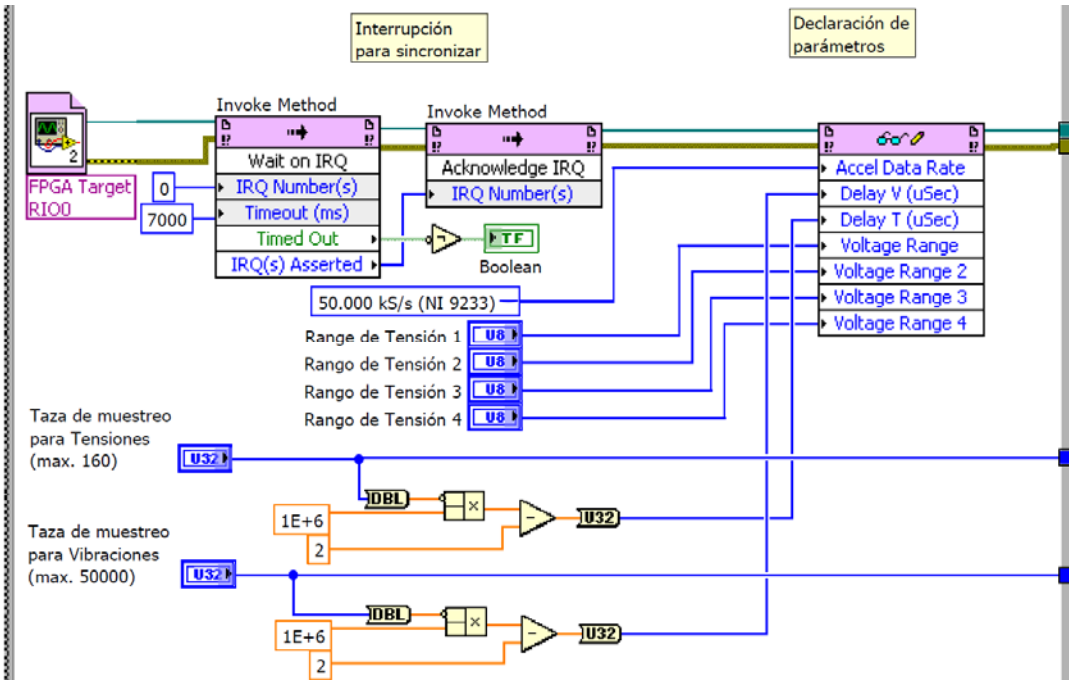


Figura 4.12 Sincronización y declaración de parámetros iniciales.

Adicionalmente, se leen los valores de calibración que se extrajeron en la primera etapa del *STC-FPGAVI-VITE* y se almacenan en un archivo binario con el nombre de *Calibracion.bin.* Ya que el archivo de calibración sólo se guarda una sola vez y no sufre modificaciones, este procedimiento se encuentra dentro de la etapa de inicialización. El código correspondiente se muestra en la Figura 4.13.

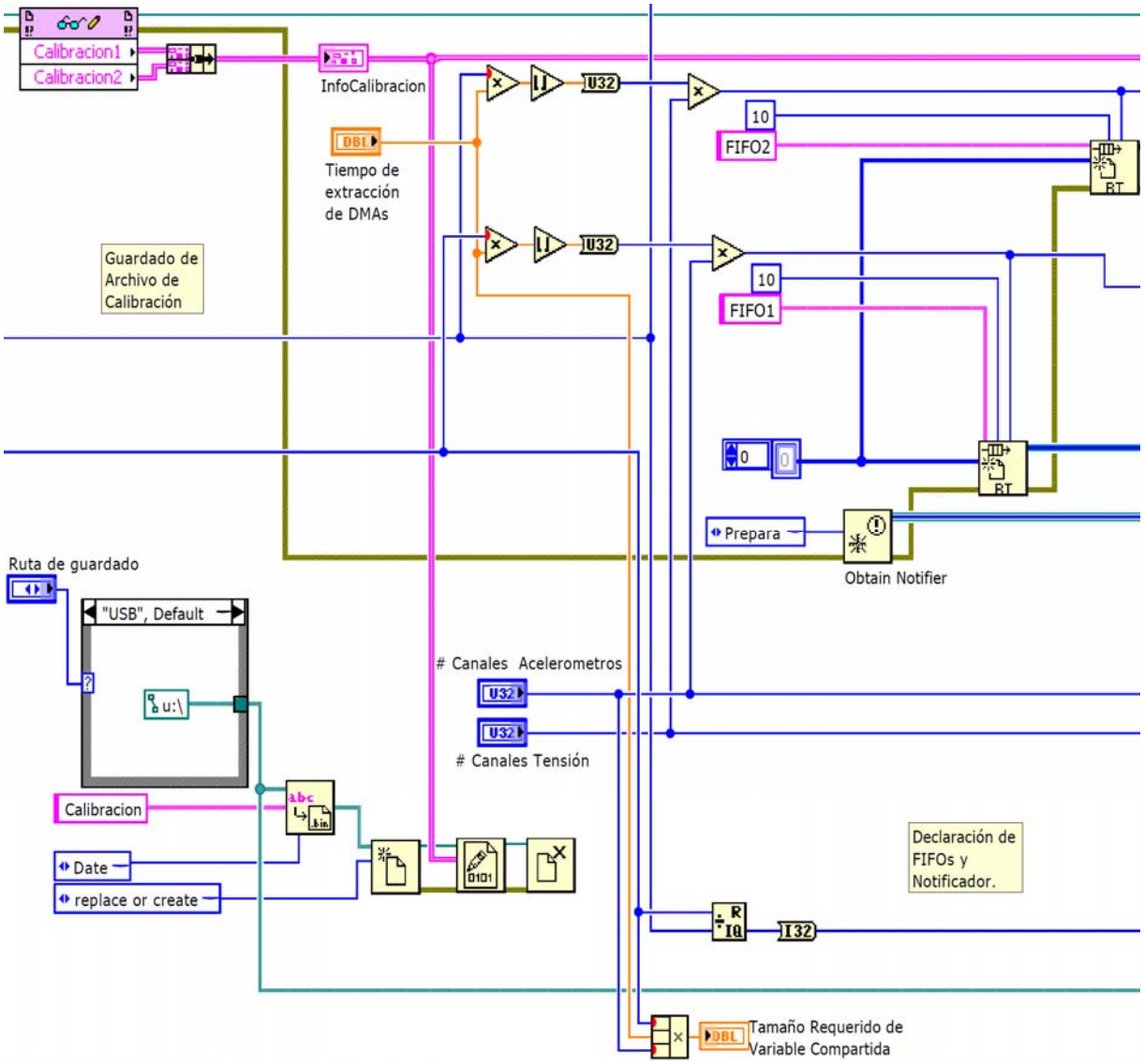


Figura 4.13 Guardado de archivo de calibración y creación de *buffers* FIFO y notificador.

Además se crean los *buffers* FIFO de tiempo real para la transferencia de datos desde los DMA FIFOs y el notificador para la logística de creación, guardado y cierre de los registros de vibraciones y tensiones.

Ya anteriormente se seleccionó el tamaño de los DMA FIFOs. Con el fin de descargar cada DMA FIFO en el momento que el número de datos que contengan represente 40 ms, el tamaño de los *buffers* FIFO dependerá de la tasa de muestreo, además de que debe ser múltiplo del número de canales disponibles.

Dado que para el registro de tensiones se cuenta con cuatro canales habilitados a una tasa de R muestras/s, entonces

$$Tamaño\ de\ buffer = \left[\mathrm{R}\ \frac{\text{muestras}}{\mathrm{s}} \times \frac{40}{1000}\ \mathrm{s} \right]_{\text{redondeado}} \times \ 4\ \text{canales.}$$

De manera análoga, para el registro de vibraciones de 8 canales habilitados a una tasa de R muestras/ s, se tiene que

$$Tamaño\ de\ buffer = \left[\mathrm{R}\ \frac{\text{muestras}}{\mathrm{s}} \times \frac{40}{1000}\ \mathrm{s} \right]_{\text{redondeado}} \times \ 8\ \text{canales.}$$

4.3.2.2 Operación

En la etapa de operación se ejecutan dos lazos paralelos sincronizadamente que realizarán la extracción, guardado y publicación de los datos para su monitoreo.

El lazo principal se encarga de la extracción de datos de los DMA FIFOs y su publicación en red para monitoreo; además de la toma de decisiones para el correcto flujo del programa desde el inicio hasta el cierre de la etapa de adquisición. El lazo secundario se encarga de crear y guardar los registros.

La ejecución de los lazos tiene una configuración de máquina de estado como se muestra en la Figura 4.14.

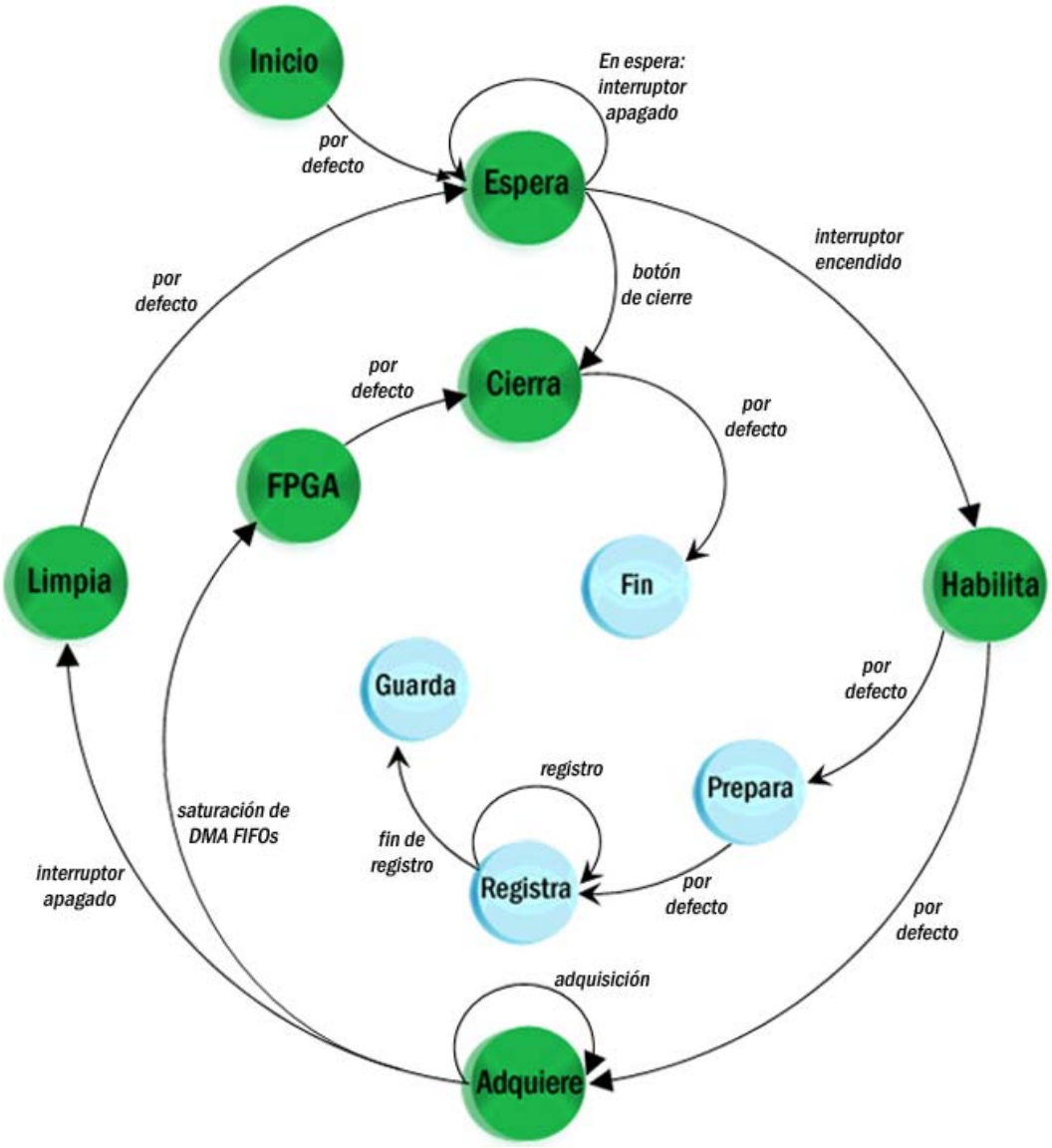


Figura 4.14 Diagrama de estados.

Al lazo central corresponden los estados: *espera*, *habilita*, *adquiere*, *FPGA*, *limpia* y *cierra*. El lazo secundario está conformado por los estados: *prepara*, *registra*, *guarda* y *fin*. Estos últimos se ejecutan dependiendo de la notificación que publique el estado en el que se encuentre el lazo central.

Para sincronizar la operación del lazo principal con el lazo secundario se emplea un notificador; de esta manera, al iniciar la adquisición en el lazo principal (*Habilita*) el notificador activa el estado de preparación en el secundario (*Prepara*), y una vez que termina la adquisición, el registro se guarda. A continuación se explica a detalle el funcionamiento del *STC-RTVI-VITE.*

El primer estado en ejecutarse es *Espera*. El programa se mantiene en este estado hasta que se active el interruptor *User1*, o se solicite finalizar el programa. El código del estado *Espera* se muestra en la Figura 4.15.

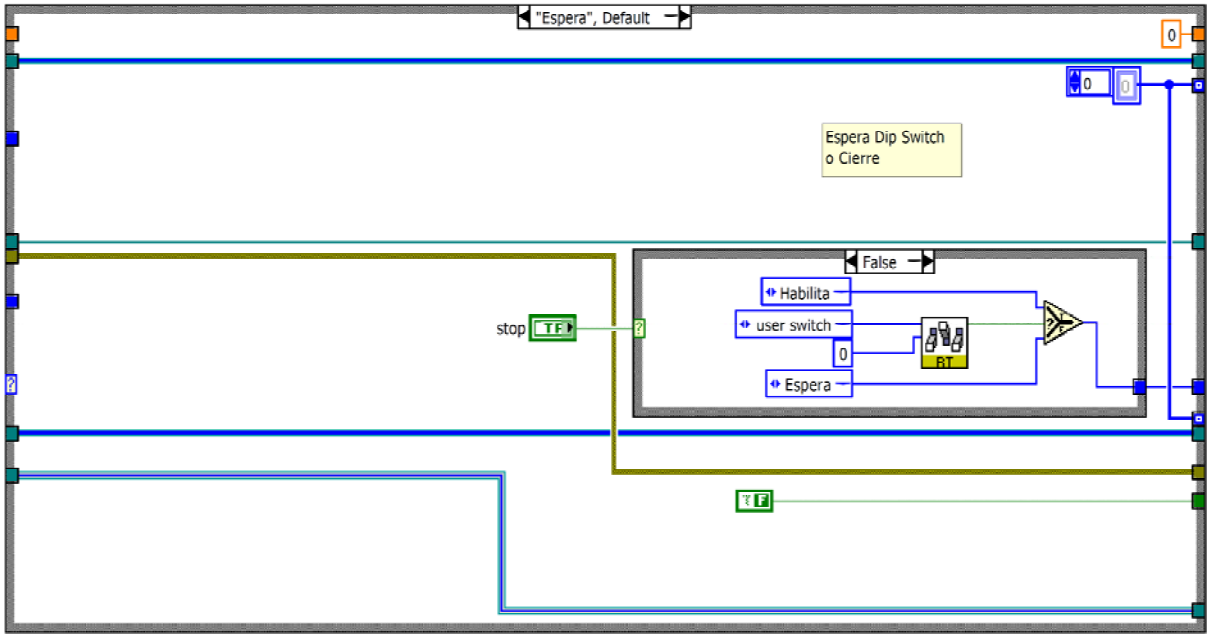


Figura 4.15 Estado *Espera*.

Cuando se activa el interruptor *User1* la ejecución pasa al estado *Habilita*, el cual se muestra en la Figura 4.16.

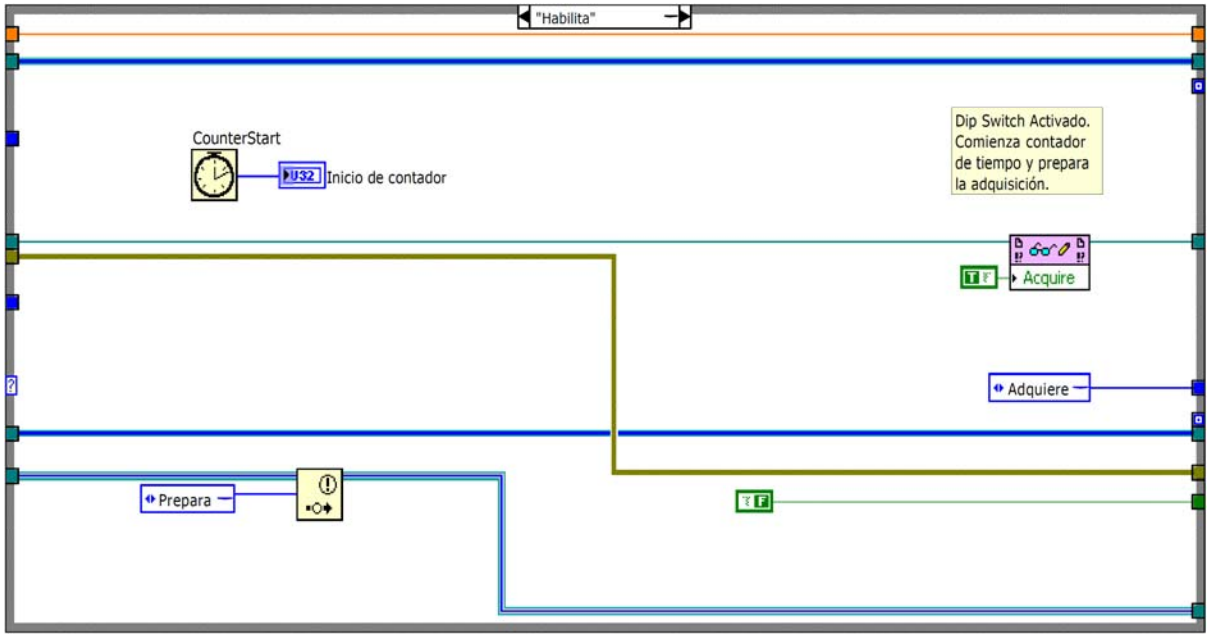


Figura 4.16 Estado *Habilita*.

Habilita activa el caso de adquisición en el *STC-FPGAVI-VITE*, con lo cual se mantiene encendido el *FPGA LED* y da inicio la adquisición y el almacenamiento de datos en los DMA FIFOs.

Además, en este estado se inicializa un contador de milisegundos y se publica la notificación *Prepara*.

Con la notificación *Prepara* el lazo secundario se encarga de la creación de tres archivos: *Vibraciones.bin*, *Tensiones.bin* y *Tiempo.bin.* El código del estado *Prepara* se muestra en la Figura 4.17.

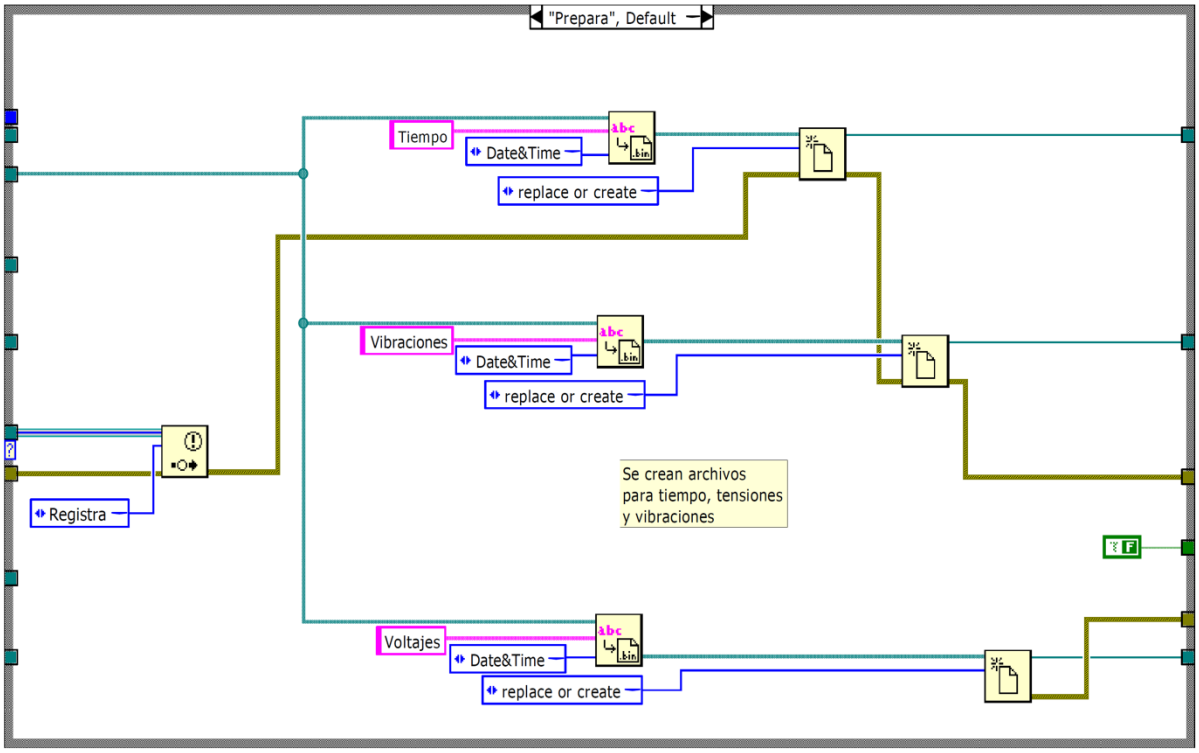


Figura 4.17 Estado *Prepara*.

De esta forma, ya se han creado los archivos para el registro y se da inicio a la extracción de los datos adquiridos por el *FPGA Target*. Al termino del estado *Habilita* se ejecuta el estado *Adquiere*, que se muestra en la Figura 4.18, mientras el estado *Prepara* publica la notificación *Registra*, que se muestra en la Figura 4.19.

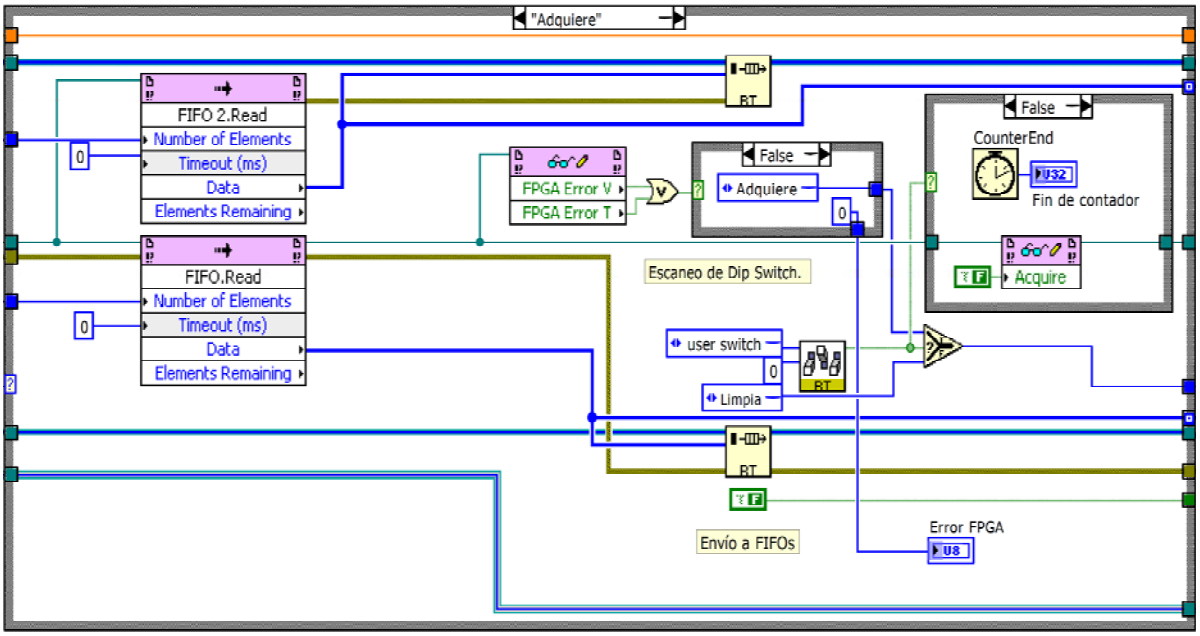


Figura 4.18 Estado *Adquiere*.

El estado *Adquiere* es el encargado de realizar la extracción de los datos almacenados en los DMA FIFOs. Para ello contiene dos bloques de lectura llamados *FIFO.Read* y *FIFO 2.Read.* Se encuentran configurados para extraer la misma cantidad de datos suficiente para llenar los *buffers* FIFO creados en la etapa de inicialización; es decir, en cada ciclo se extrae la cantidad de datos que representen 40 ms de medición. Los datos de vibración y tensión extraídos se almacenan en los *buffers* FIFO1 y FIFO2 respectivamente.

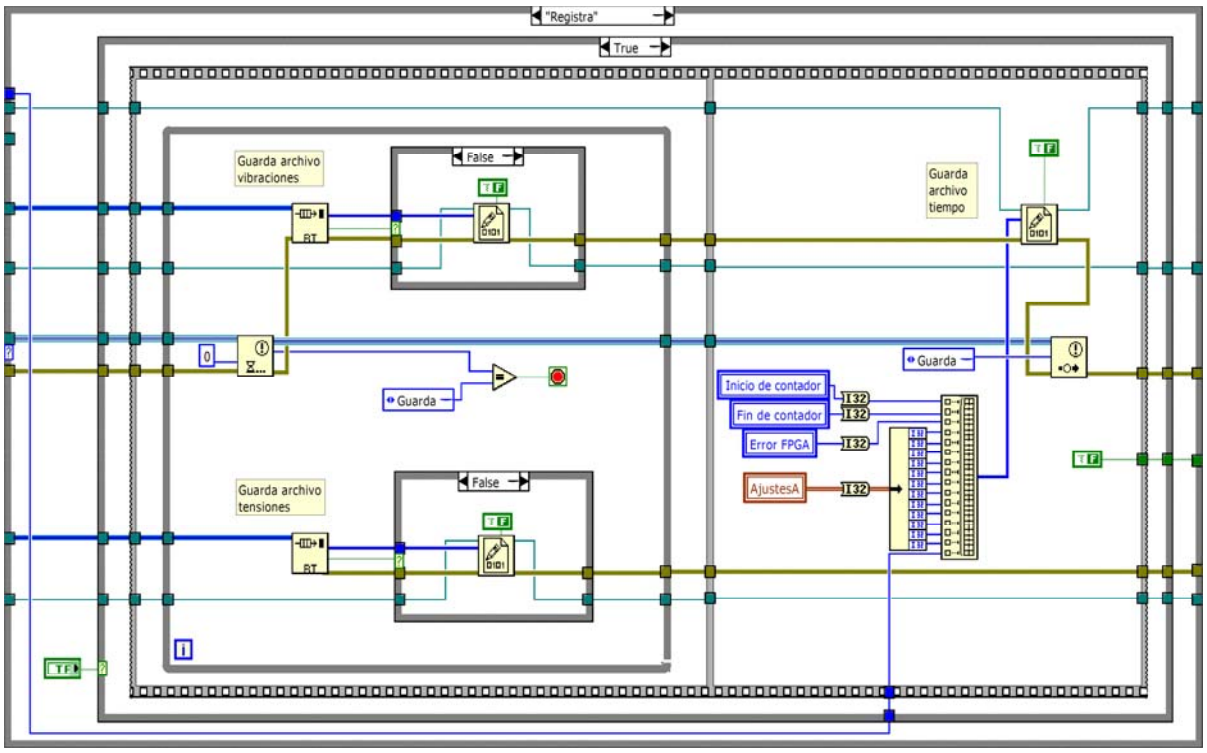


Figura 4.19 Estado *Registra*.

Adquiere verifica que no exista saturación en los DMA FIFOs. Si se da el caso, se ejecuta el estado *FPGA,* el cual vacía la totalidad de los DMA FIFOs, se publica la notificación *Guarda* y la ejecución pasa directamente al estado *Cierra*.

El código del estado FPGA se muestra en la Figura 4.20.

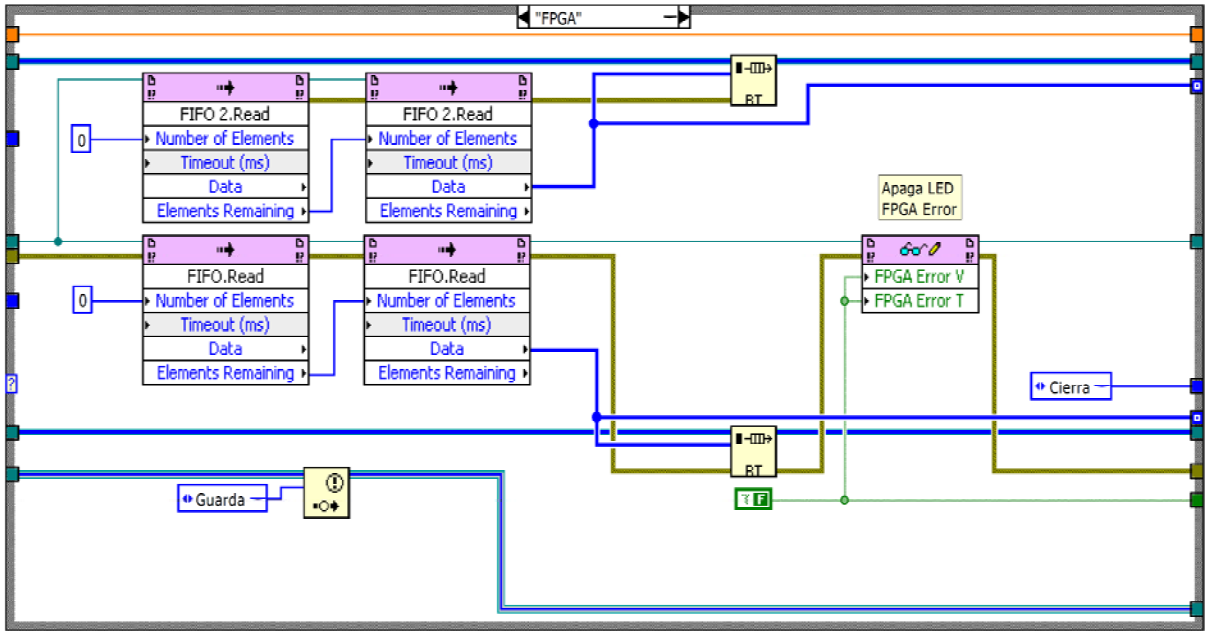


Figura 4.20 Estado *FPGA*.

Cuando se desactiva el interruptor *User1* se finaliza el contador en milisegundos que fue inicializado en el estado *Habilita*, se deshabilita el caso de adquisición en el *STC-FPGAVI-VITE* con lo que el FPGA LED reinicia su parpadeo, y la ejecución pasa del estado *Adquiere* al estado *Limpia*.

El código del estado *Limpia* se muestra en la Figura 4.21. En él se extraen los elementos de los DMA FIFOs que no alcanzaron a ser extraídos en el estado *Adquiere* y se almacenan en los *buffers*.

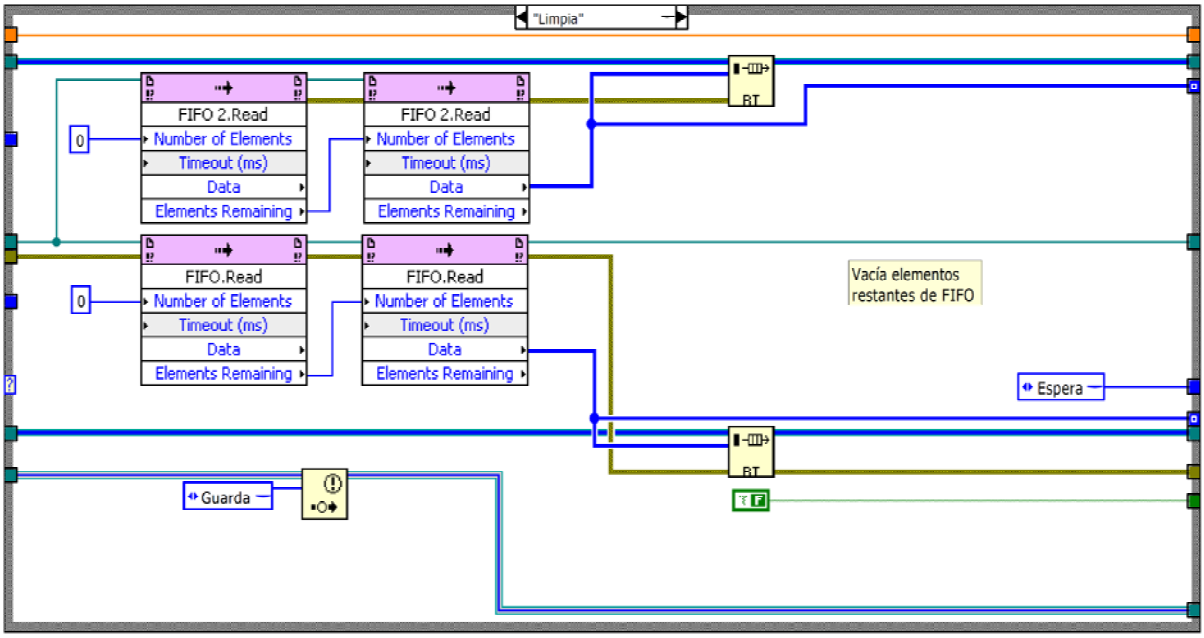


Figura 4.21 Estado *Limpia.*

El estado *Limpia* envía la notificación *Guarda*, y da lugar a que se ejecute nuevamente el estado *Espera* para iniciar así un nuevo ciclo de adquisición.

Cuando se publica la notificación *Guarda*, el lazo secundario detiene el registro de vibraciones y tensiones y guarda un archivo llamado *Tiempo.bin*. Este archivo contiene:

- El valor inicial del contador de milisegundos
- El valor final del contador de milisegundos
- Un 0 si el registro fue finalizado por el usuario, o un 1 si el registro terminó abruptamente por saturación de los DMA FIFOs en el FPGA
- Los ajustes que se hayan realizado para corregir desplazamientos de la gráfica desplegada, durante el monitoreo mediante PC de desarrollo.
- El factor entre la tasa de muestreo para vibraciones y la tasa de muestreo para tensiones.

Cuando la notificación ejecuta el estado *Guarda* en el lazo secundario, se cierran los archivos *Vibraciones.bin*, *Tensiones.bin* y *Tiempo.bin*, como se muestra en el código de la Figura 4.22. Además se limpian las referencias a estos archivos para poder crear unos nuevos en caso de que se inicie un nuevo ciclo de adquisición.

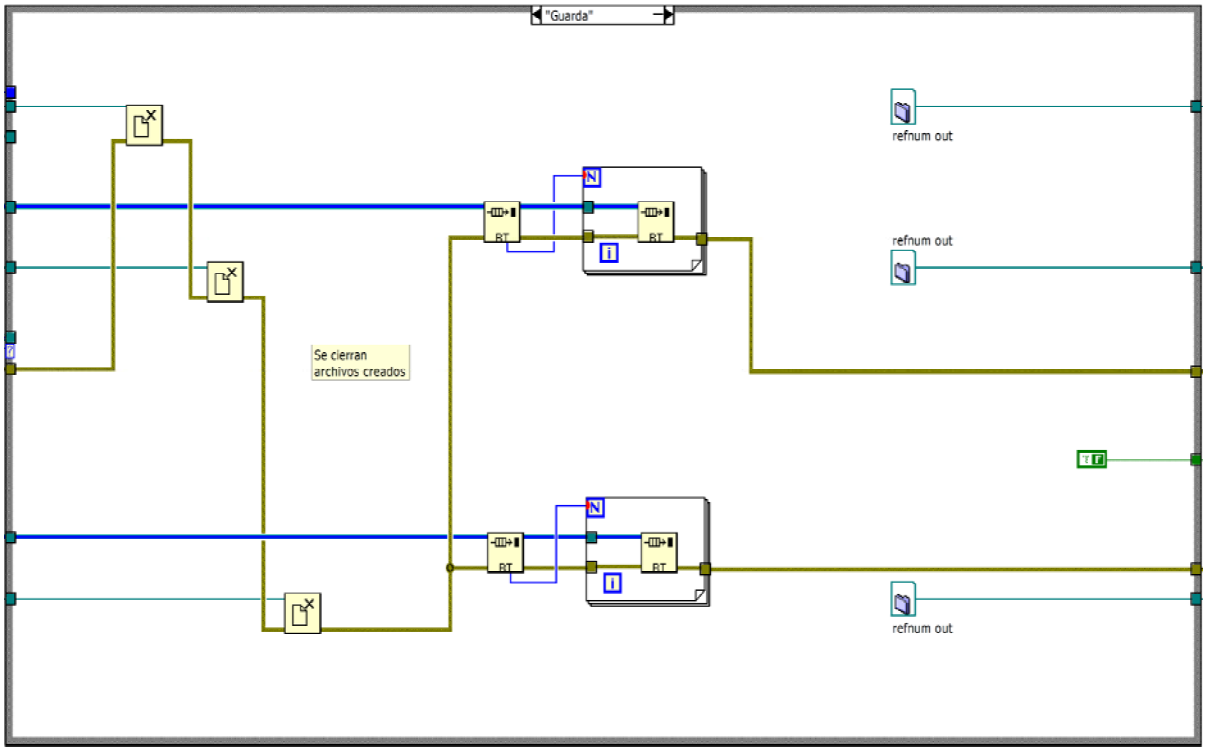


Figura 4.22 Estado *Guarda.*

Siempre que se presione el botón *CERRAR* en el estado de *Espera* la ejecución pasa al estado *Cierra*. En este estado se publica la notificación *Fin* y se detiene la ejecución del lazo principal. El código del estado *Cierra* se muestra en la Figura 4.23.

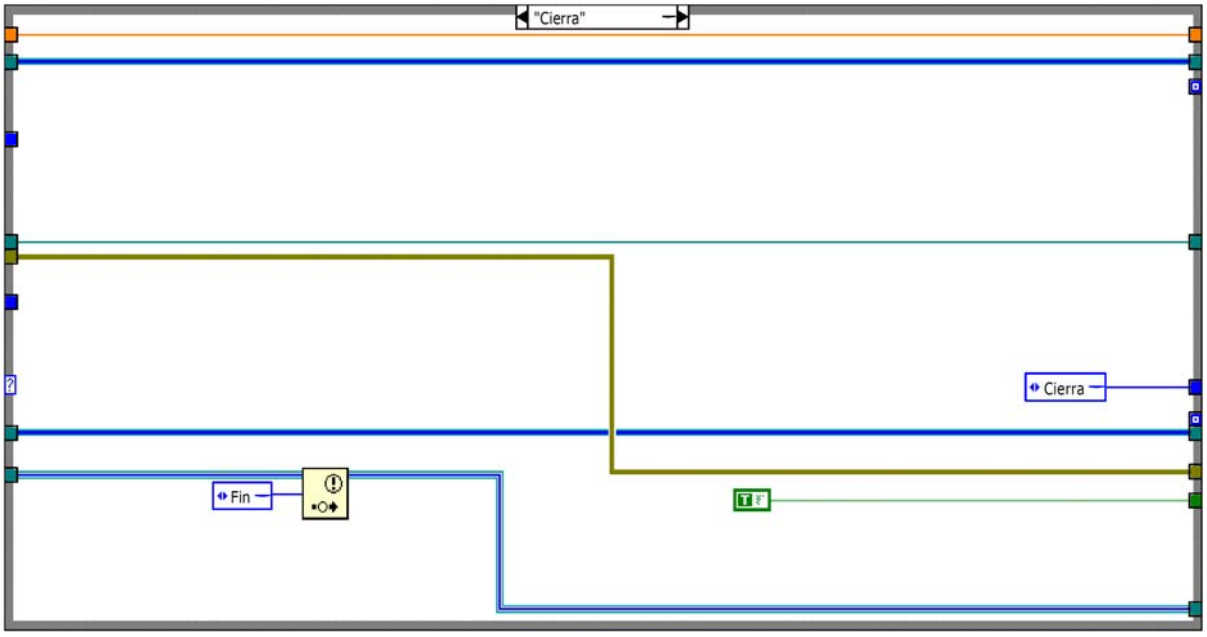


Figura 4.23 Estado *Cierra.*

Por su parte, la notificación *Fin* detiene la ejecución del lazo secundario. El código del estado *Fin* es el mostrado en la Figura 4.24.

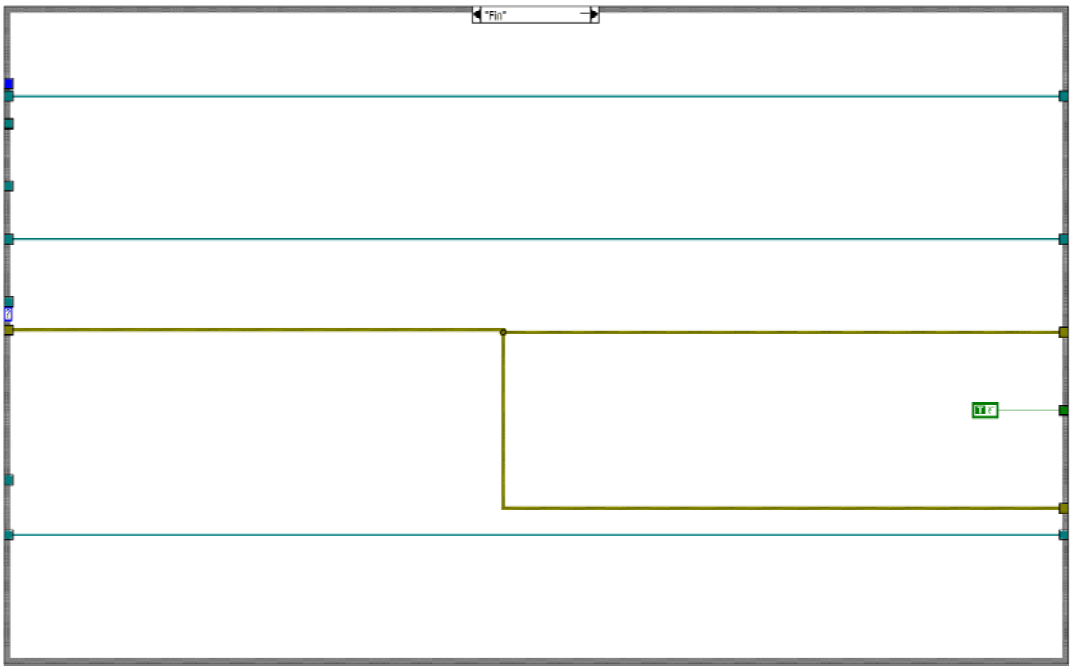


Figura 4.24 Estado *Fin.*

El lazo principal cuenta con una sección adicional orientada al monitoreo de los datos. Para ello lee los datos extraídos de los DMA FIFOs, les aplica ajustes para poder analizarlos y graficarlos, y los publica como variables compartidas en red. Esta sección orientada al monitoreo y a la publicación de variables compartidas en red se muestra en el código de la Figura 4.25.

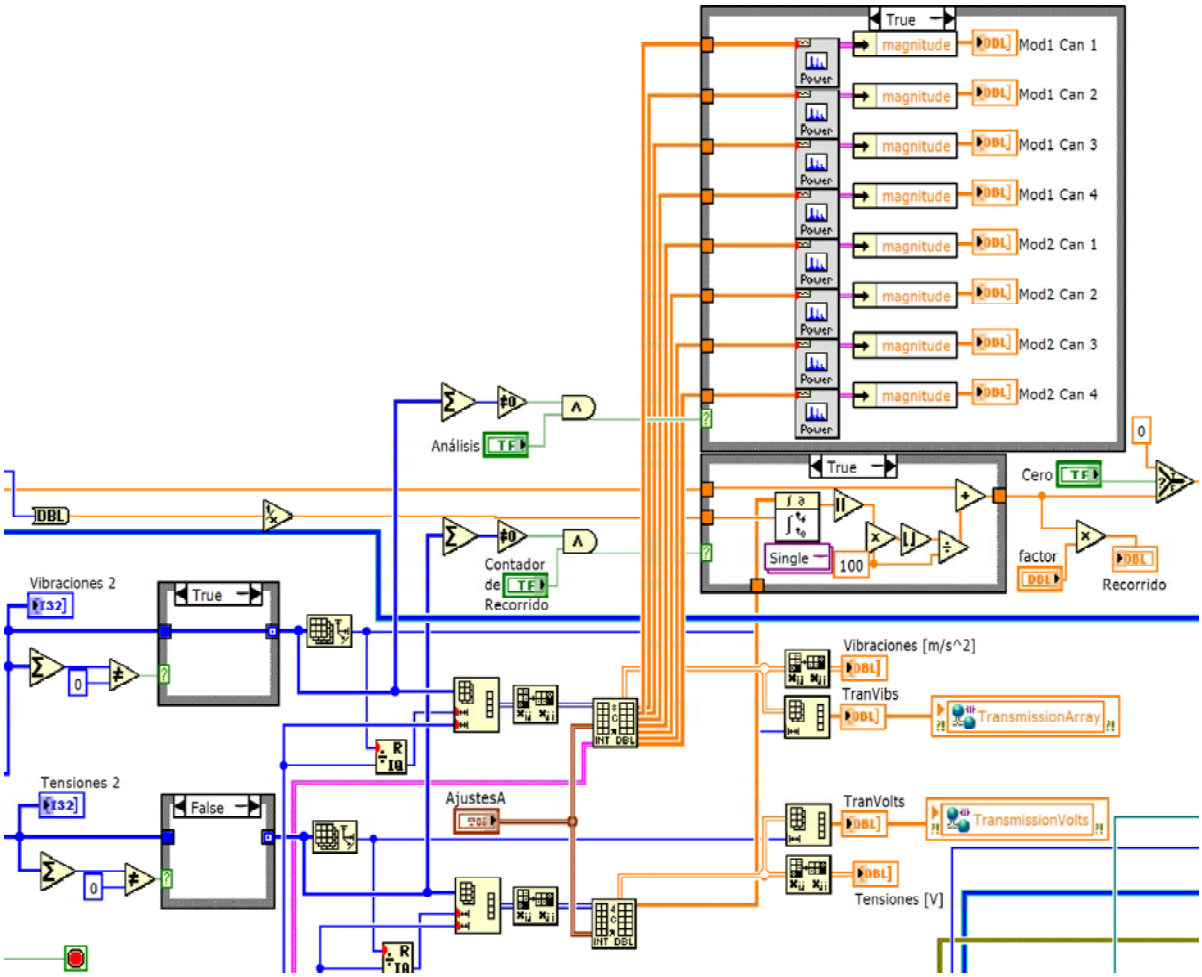


Figura 4.25 Sección orientada a monitoreo.

4.3.2.3 Finalización

Por último se ejecuta la etapa de finalización, que se muestra en la Figura 4.26.

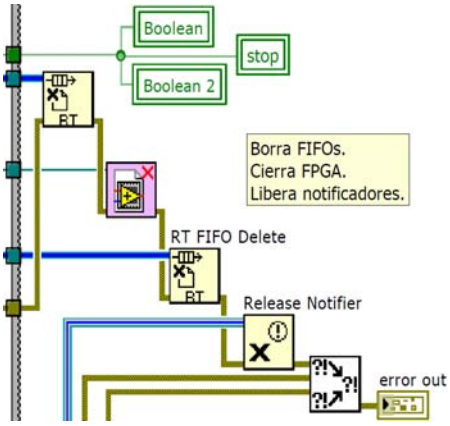


Figura 4.26 Etapa de finalización.

En esta etapa se eliminan los *buffers* FIFO, se libera el notificador, se limpian los indicadores booleanos y se finaliza la etapa de adquisición en el *STC-FPGAVI-VITE* para posteriormente borrar toda referencia al *FPGA Target*.

4.3 Modalidades de funcionamiento

Se contemplaron tres modalidades de funcionamiento: CompactRIO independiente sin monitoreo, CompactRIO independiente con monitoreo y CompactRIO con PC de desarrollo.

4.3.1 CompactRIO independiente sin monitoreo

En esta modalidad se utiliza el CompactRIO como registrador sin necesidad de conectar algún instrumento de monitoreo. Para realizar la adquisición se efectúa lo siguiente:

1 Insertar una unidad de almacenamiento en el puerto USB con una capacidad mínima de 1 GB.
2 Verificar que todos los interruptores de la cara frontal del controlador se encuentren en la posición de apagado (OFF)
3 Alimentar el CompactRIO.
4 Esperar a que el *FPGA LED* parpadee, lo cual indica que el CompactRIO se encuentra listo para iniciar el registro.

5 Mover el interruptor *User1* a la posición de encendido para iniciar el registro.
6 Para finalizar el registro colocar el interruptor *User1* de vuelta en posición de apagado.

Para realizar un nuevo registro, basta con volver a colocar el interruptor *User1* en posición de encendido al inicio de la medición y en posición de apagado al final de la misma. Esto se repite tantas veces como registros se deseen.

Cada vez que se coloca el interruptor *User1* en posición de apagado, el *FPGA LED* vuelve a parpadear indicando que se encuentra listo para un nuevo registro.

Para finalizar la adquisición basta con desconectar el CompactRIO de la alimentación. Dado que ya han sido cerrados los archivos *Vibraciones.bin*, *Tensiones.bin* y *Tiempo.bin*, así como el de *Calibración.bin* y se ha limpiado la referencia a estos archivos programáticamente, el retiro de la alimentación al CompactRIO no perjudica ni al sistema ni a los registros.

Para realizar algún cambio en las propiedades de adquisición, como tasa de muestreo y el rango de tensión, se deberán realizar los ajustes en el *STC-RTVI-VITE* desde la PC de desarrollo e instalarlos en el CompactRIO a través del árbol del proyecto.

4.3.2 CompactRIO independiente con monitoreo

Bajo esta modalidad, el CompactRIO opera en conjunto con un instrumento de monitoreo. Esta unidad de monitoreo debe tener cargado el software LabVIEW, LabVIEW Embedded o el LabVIEW Run-Time Engine 2009.

Para realizar la adquisición se lleva a cabo lo siguiente.

1 Insertar una unidad de almacenamiento en el puerto USB con una capacidad mínima de 1 GB.
2 Verificar que todos los interruptores de la cara frontal del controlador se encuentren en la posición de apagado (OFF)
3 Conectar la unidad de monitoreo (pantalla táctil, laptop, etc.) al CompactRIO mediante cable Ethernet cruzado. Verificar que la dirección IP sea correcta.
4 Alimentar el CompactRIO.
5 Alimentar y encender el dispositivo de monitoreo.
6 Abrir y ejecutar el programa de monitoreo en el dispositivo de monitoreo.
7 Esperar a que el *FPGA LED* parpadee, lo cual indica que el CompactRIO se encuentra listo para iniciar el registro.

8 Mover el interruptor *User1* a la posición de encendido para iniciar el registro.
9 Para finalizar el registro colocar el interruptor *User1* en posición de apagado.

El dispositivo de monitoreo no es indispensable para poder realizar la adquisición; sin embargo, permite visualizar en tiempo real las señales adquiridas. En cualquier momento puede ejecutarse el programa de monitoreo, incluso si ya inició la adquisición de vibraciones. Asimismo, puede finalizarse el programa de monitoreo en cualquier momento sin afectar la adquisición de datos.

Al igual que en la modalidad sin monitoreo, para realizar algún cambio en las propiedades de adquisición se deberán realizar los ajustes en el *STC-RTVI-VITE* desde la PC de desarrollo e instalarlos en el CompactRIO a través del árbol del proyecto.

4.3.3. CompactRIO con PC de desarrollo

Para poder realizar los registros bajo esta modalidad, es necesaria una PC con el software LabVIEW instalado incluyendo los módulos FPGA y Real-Time. Además debe conocerse la ubicación del programa principal dentro del proyecto y ejecutarse directamente desde ahí.

La adquisición se lleva a cabo de la siguiente manera:

1 Insertar una unidad de almacenamiento en el puerto USB con una capacidad mínima de 1 GB.
2 Colocar el interruptor *No App* en la posición de encendido (ON) y el resto en la posición de apagado (OFF).
3 Alimentar el CompactRIO.
4 Alimentar y encender la PC de desarrollo (puede ser una laptop)
5 Conectar la PC al CompactRIO mediante cable Ethernet cruzado. Verificar que las direcciones IP sean correctas.
6 Abrir el proyecto *STC-VITE*.lvproj en LabVIEW.
7 En el árbol del proyecto abrir el programa principal *STC-RTVI-VITE*.
8 Pulsar el botón *Run* en la barra de tareas para ejecutar el programa.
9 Esperar a que el *FPGA LED* parpadee, lo cual indica que el CompactRIO se encuentra listo para iniciar el registro.
10 Mover el interruptor *User1* a la posición de encendido para iniciar el registro.
11 Para finalizar el registro colocar el interruptor *User1* en posición de apagado.

Una vez realizados los registros necesarios, para finalizar se debe pulsar el botón CERRAR en el *STC-RTVI-VITE* mientras el *FPGA LED* se encuentre parpadeando. Al hacer esto se apaga el *FPGA LED* por completo.

En esta modalidad pueden realizarse modificaciones al código del programa; sin embargo, para ello debe conocerse bien la estructura y el funcionamiento del mismo.

Esta modalidad permite modificar la tasa de muestreo y el rango de tensión de los módulos, sin tener que realizar algún cambio al código del programa.

Para que esta modalidad pueda funcionar, es necesario que todos los subVIs que interactúan con el *STC-RTVI-VITE*, así como la carpeta que contiene el archivo de bits (bitfile) que se creó de la compilación del *STC-FPGAVI-VITE,* se encuentren en una misma carpeta.

Nota: En cualquier modalidad, si se llega a presentar un error de FPGA, la ejecución del programa se detiene y el *FPGA LED* se apaga por completo. Para que sea posible realizar un nuevo registro cuando no se emplea PC de desarrollo, debe presionarse el botón *Reset* en el CompactRIO. Empleando la PC de desarrollo, basta con presionar nuevamente el botón *Run*; sin embargo, en ambos casos, dado que este fenómeno ocurre debido a un error en la configuración del DMA, la falla volverá a presentarse. Para resolver el problema será necesario revisar la programación.