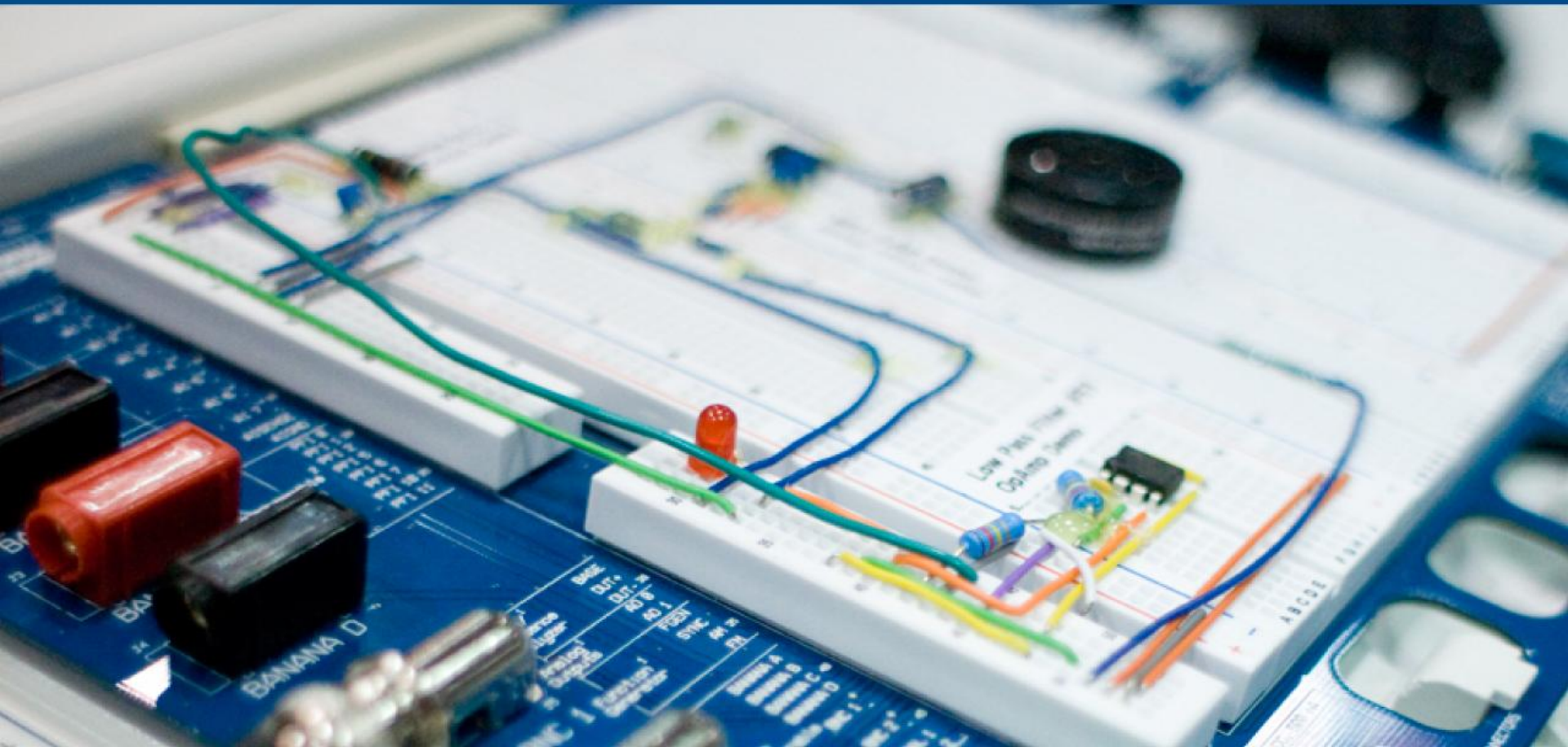


# NI Academic Days 2011

6 de Mayo ▪ Bogotá, Colombia

[ni.com/colombia](http://ni.com/colombia)

01 8000 513680 o (1) 482.4888



# **Técnicas de Programación para Obtener el Máximo Provecho de LabVIEW**

Felipe Rincón

Field Sales Engineer

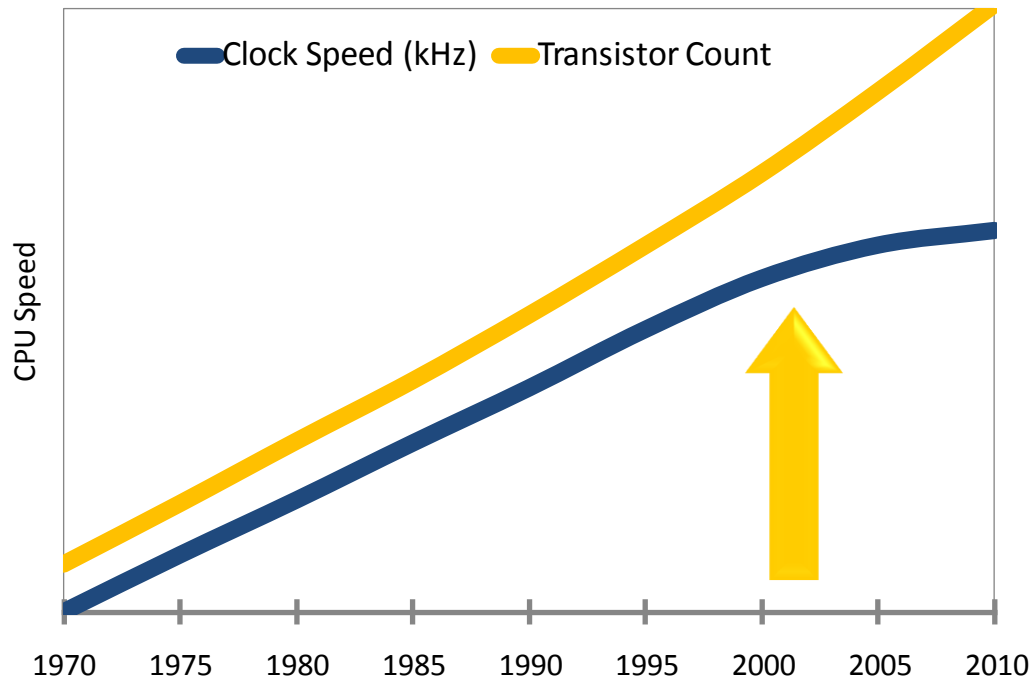
National Instruments Andean & Caribbean

# Agenda

- Técnicas de Programación Multinúcleo
- Patrones de Diseño Avanzados
- Programación Orientada a Objetos
- Herramientas para Aplicaciones a Gran Escala

# Arquitecturas Paralelas Aumentan Desempeño

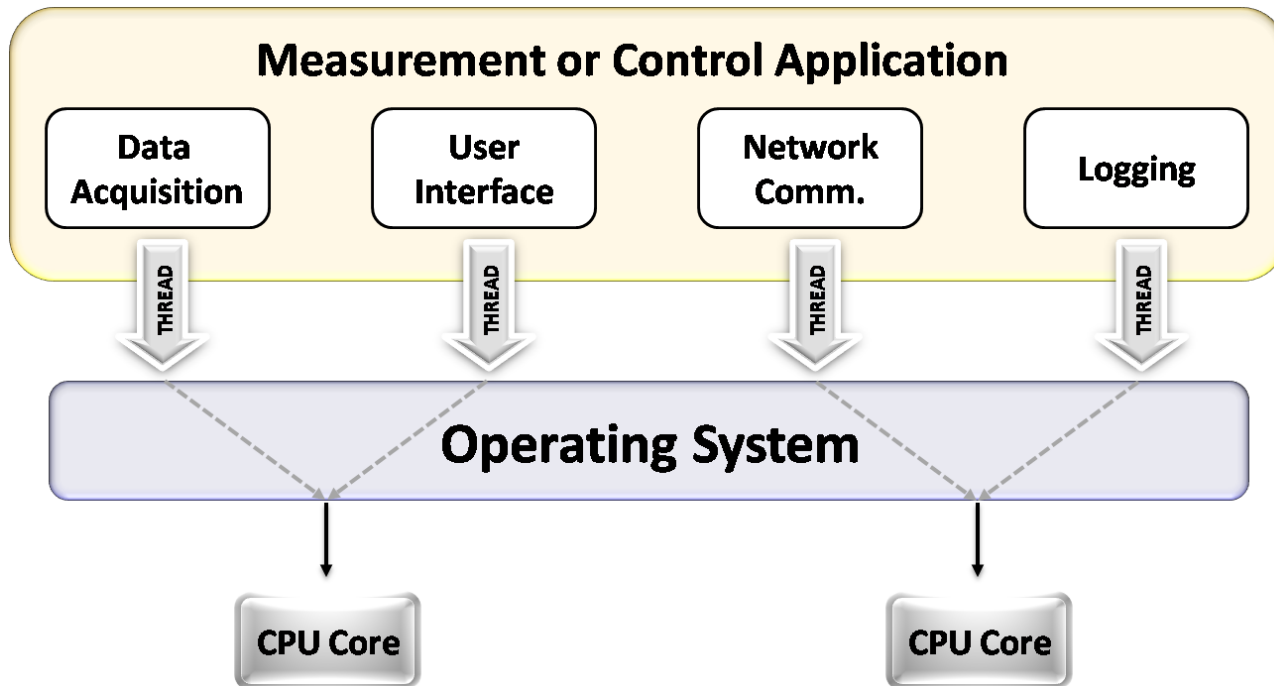
Procesadores rápidos → Procesadores Multinúcleo



Procesador Intel QX6700 Quad Core

- 4 procesadores (pare de Core 2)
- Velocidad de reloj 2.66 GHz

# Creando Aplicaciones Multitarea



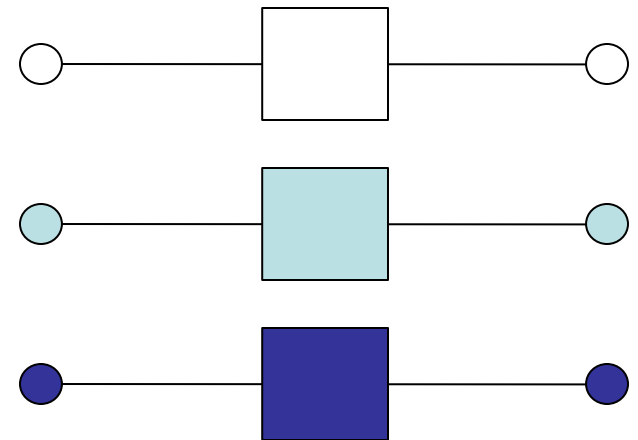
Aplicaciones deben hacer uso de las tareas para beneficiarse de los procesadores multinúcleo.

# Programas Multitarea en LabVIEW

- Multitarea automático
  - LabVIEW automáticamente asigna tareas basado en el paralelismo
  - La mayoría de los programas existentes correrán más rápido en un sistema Multinúcleo **sin alteración**
- Multitarea manual
  - Se limita la sección de código a ejecutarse como una tarea

# Técnicas para Paralelismo

- Paralelismo de Tareas y Multitareas Automático
- Paralelismo de Datos
- Pipelining y balanceo de etapas



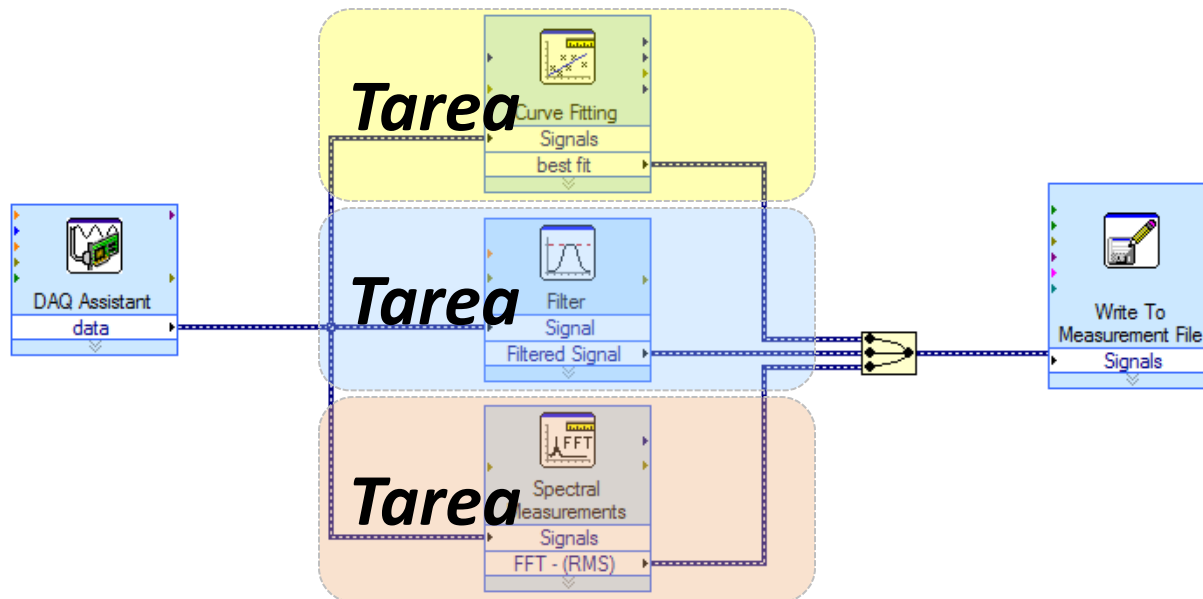
# Paralelismo Tareas

- 1) Buscar tareas que pueden correr en paralelo
- 2) La arquitectura del código refleja el paralelismo
  - Eliminar dependencias de datos
  - LabVIEW automáticamente identifica código paralelo y puede separarlo en múltiples tareas!



# Ejemplo: Paralelismo de Tareas

## Operaciones Paralelas

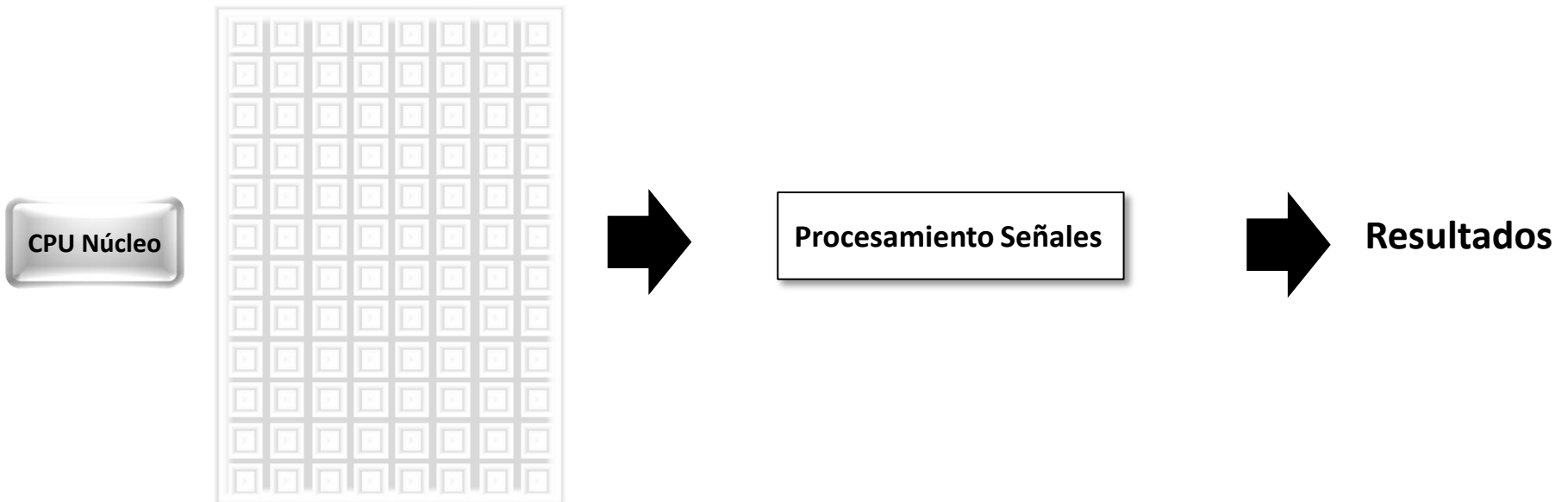


# Paralelismo de Datos

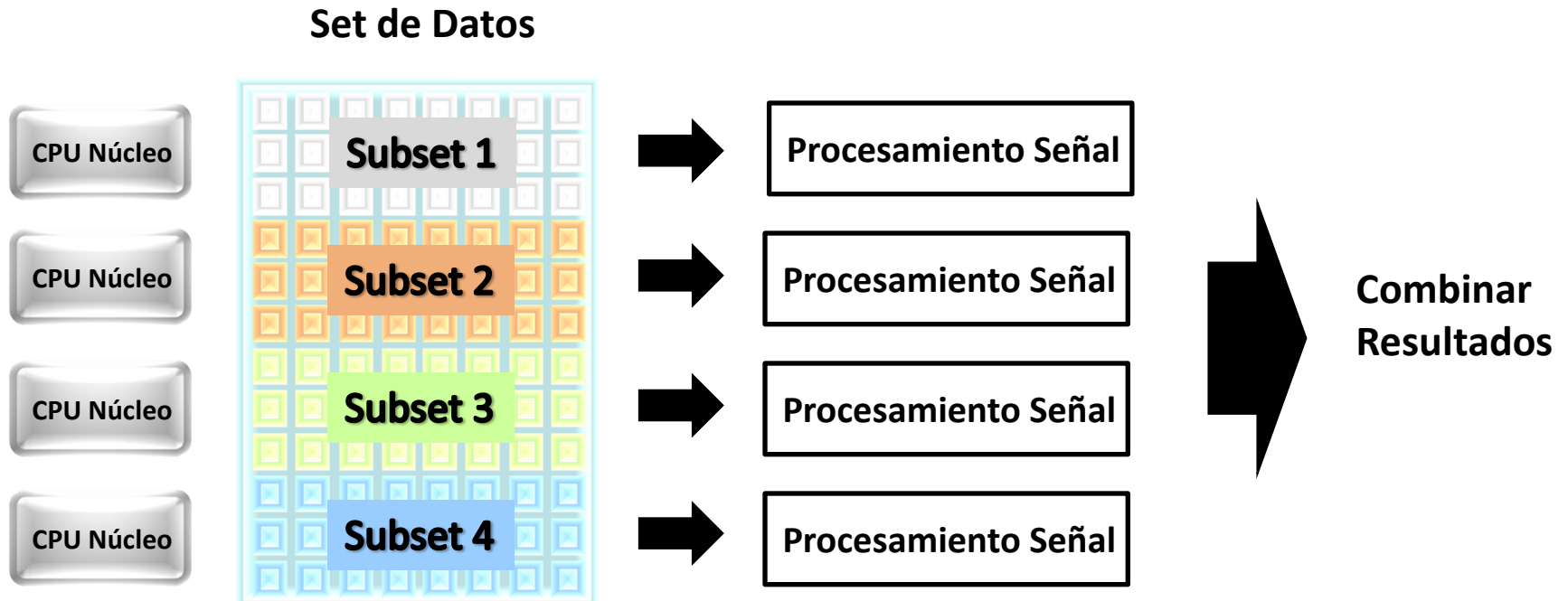
- 1) Buscar un set de datos grande que pueda ser procesado en dos o más pedazos independientes
- 2) Arquitectura de código:
  - Separar los datos
  - Procesar los datos en paralelos
  - Combinar los resultados individuales para obtener un resultado general

# Ejemplo: Paralelismo de Datos

Set de Datos

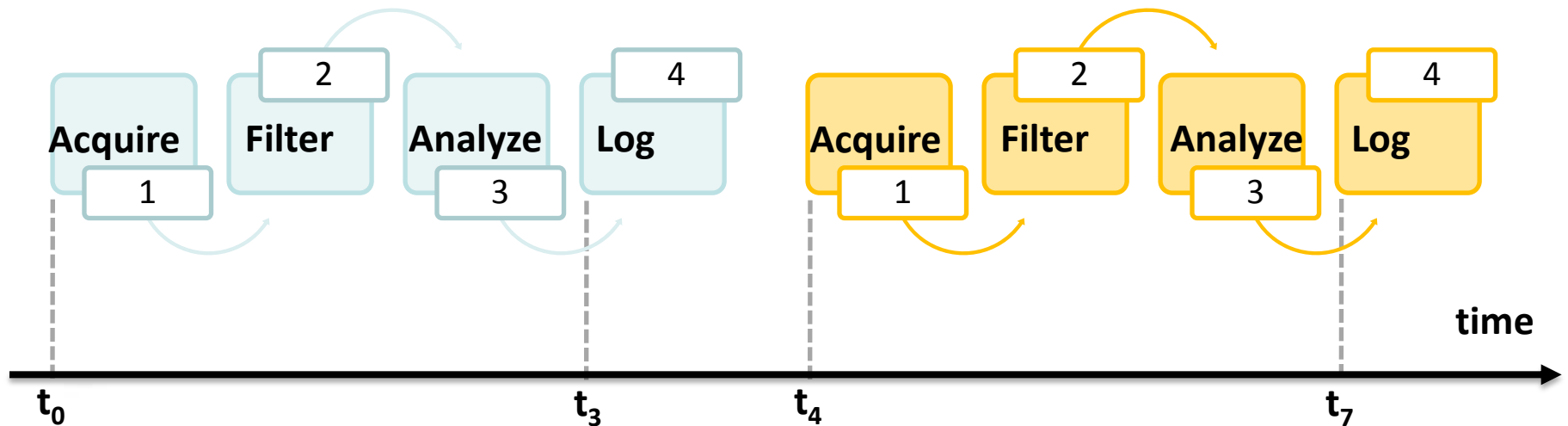


# Ejemplo: Paralelismo Datos

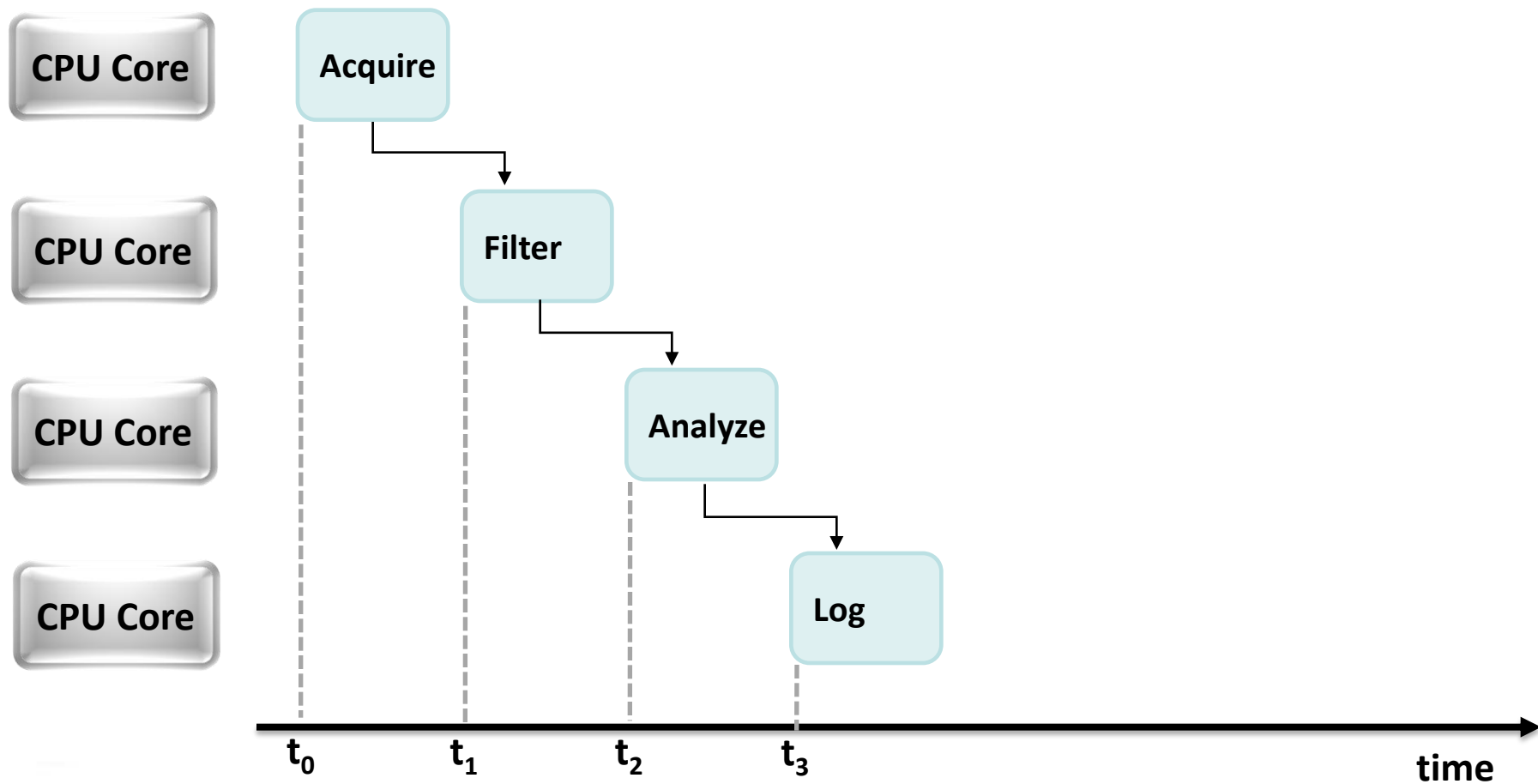


# Pipelining

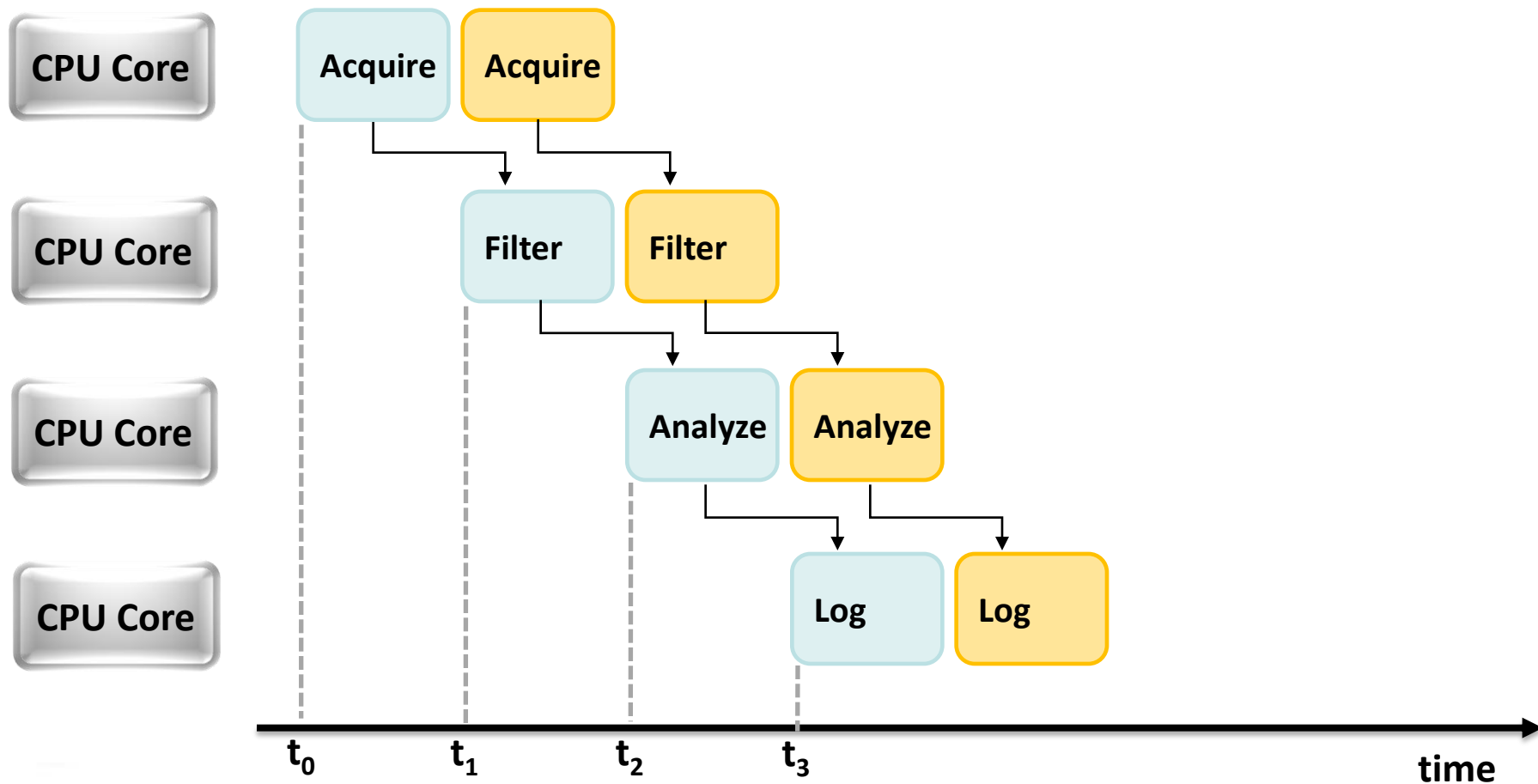
- Motivación: Muchos programas contienen secuencias, algoritmos de múltiples pasos
- Aplicando pipelining se puede incrementar la cantidad de datos procesados



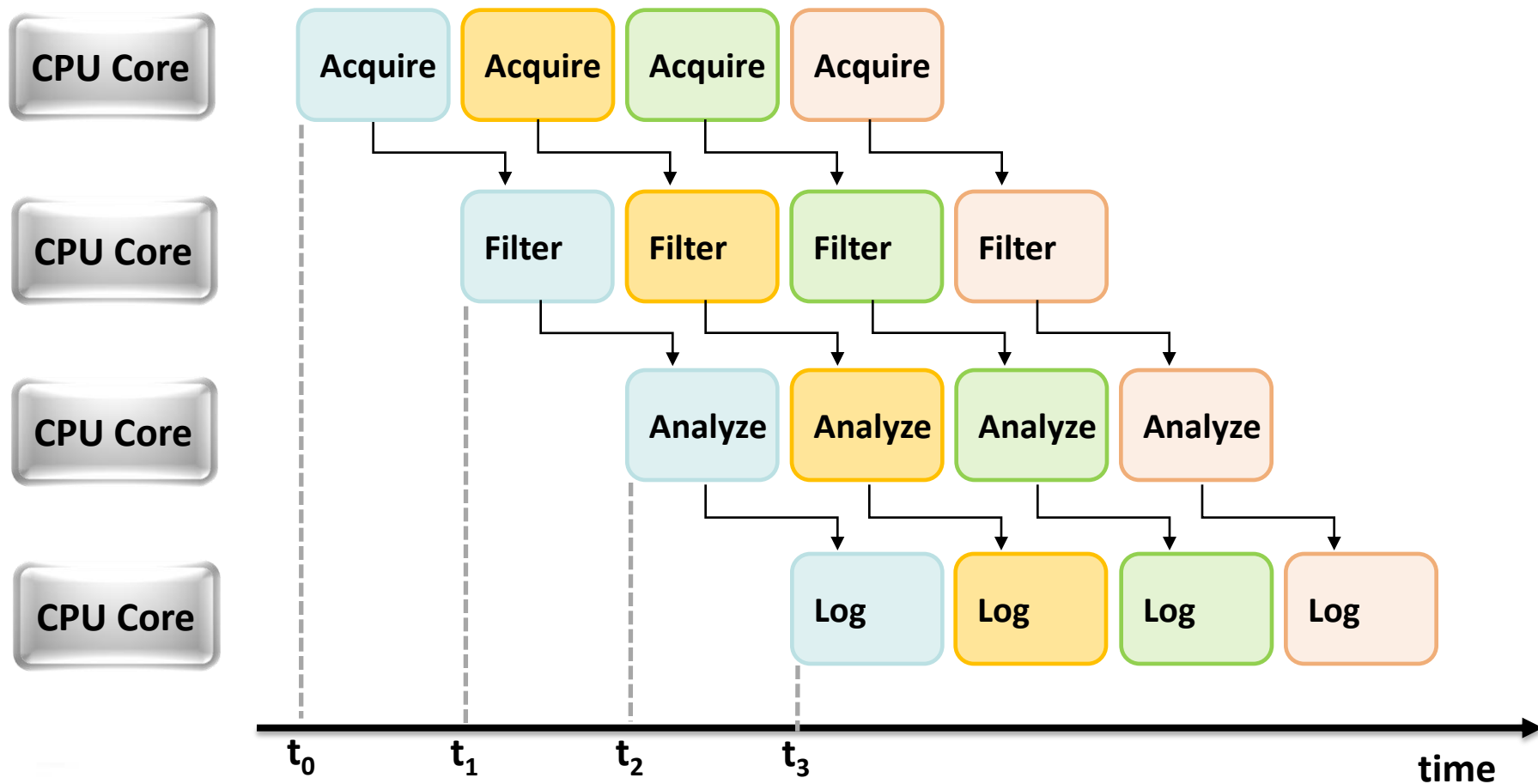
# Estrategia Pipelining



# Estrategia Pipelining



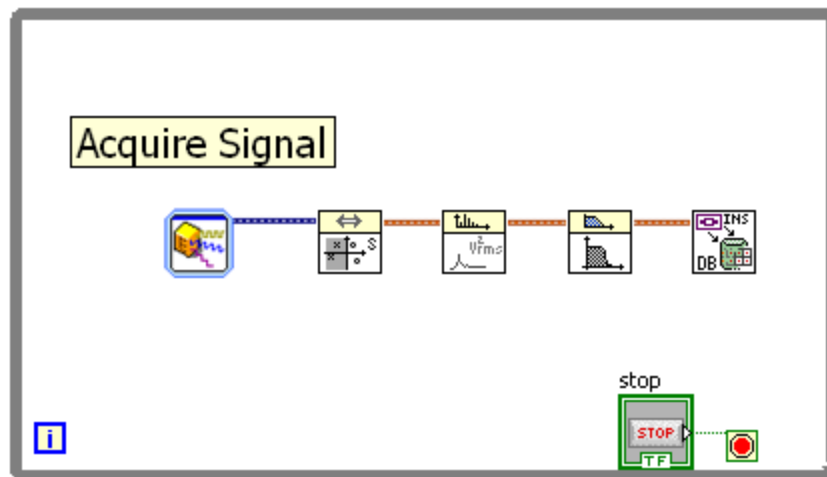
# Estrategia Pipelining



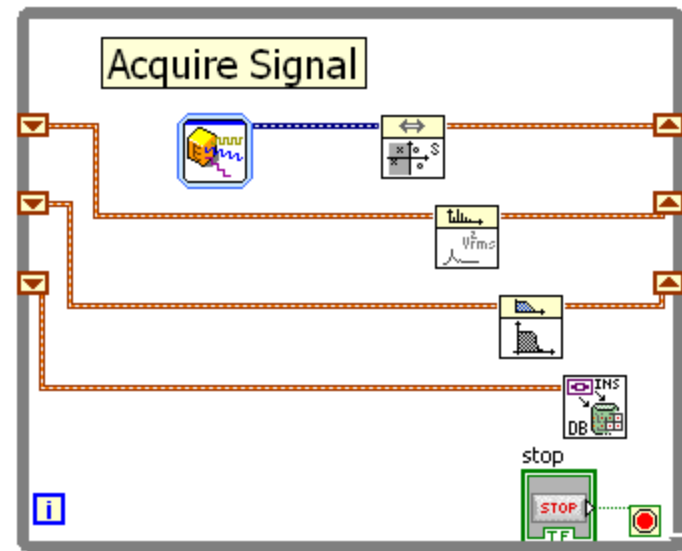


# Pipelining en LabVIEW

## Secuencial



## Pipelining

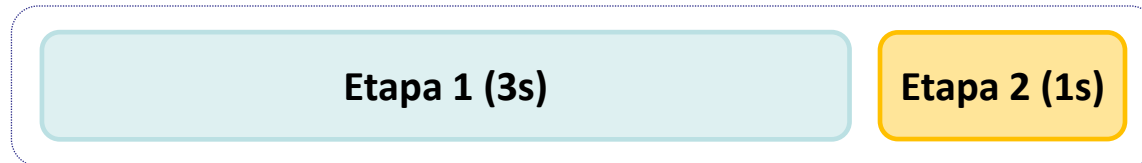


Nota: otras técnicas existen para pipelining como utilizar múltiples ciclos con buffers, y utilizar nodos de retroalimentación

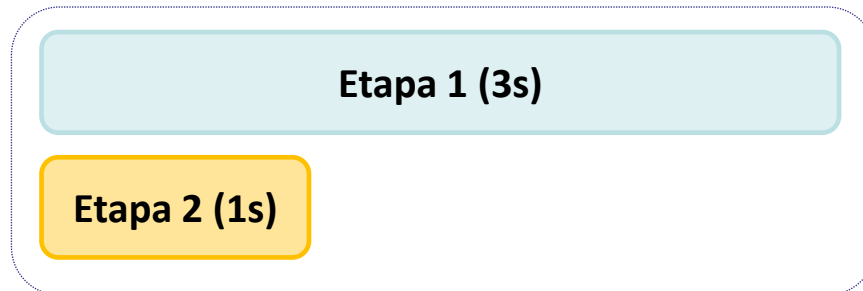
# Balanceo de Etapas

- La ruta crítica es la etapa más larga
- Pipelining con etapas desbalanceadas no necesariamente da una mejora en desempeño

Sin Pipelining (tiempo total= 4s)



Con Pipelining (tiempo total= 3s): aumento velocidad = 1.33X (no ideal para pipelining)

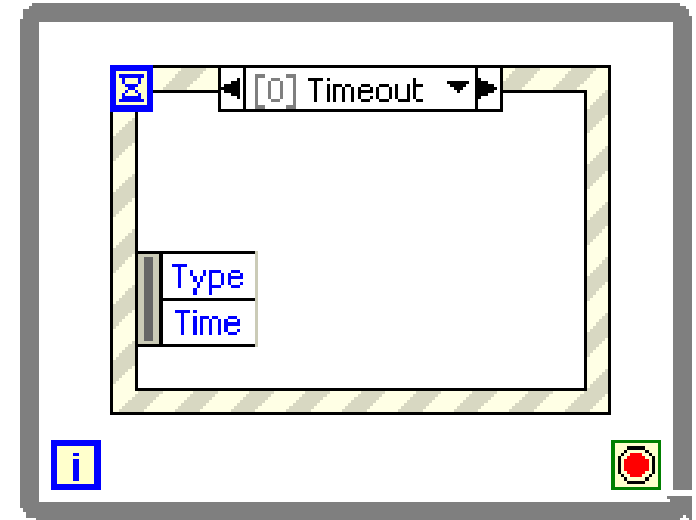
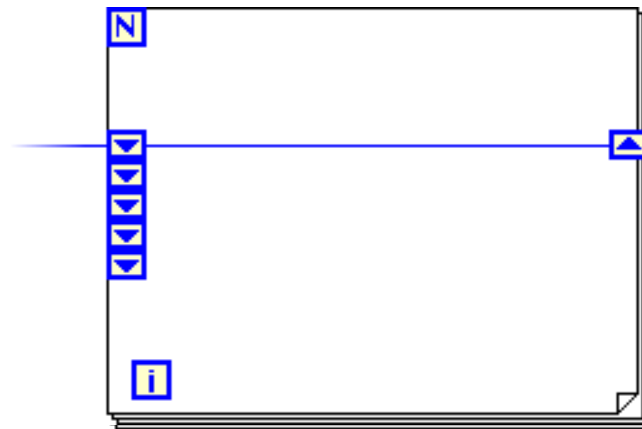
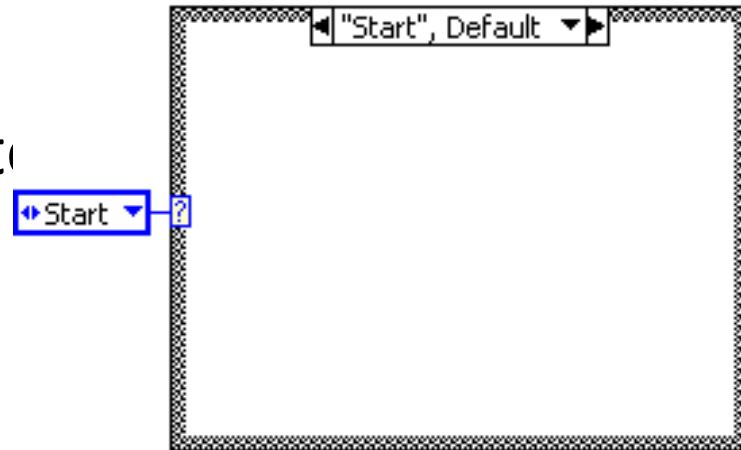


# ¿Qué Son los Patrones de Diseño?

- Una platilla o arquitectura para código de LabVIEW
- Ampliamente aceptado y bien conocido
- Fácilmente reconocible

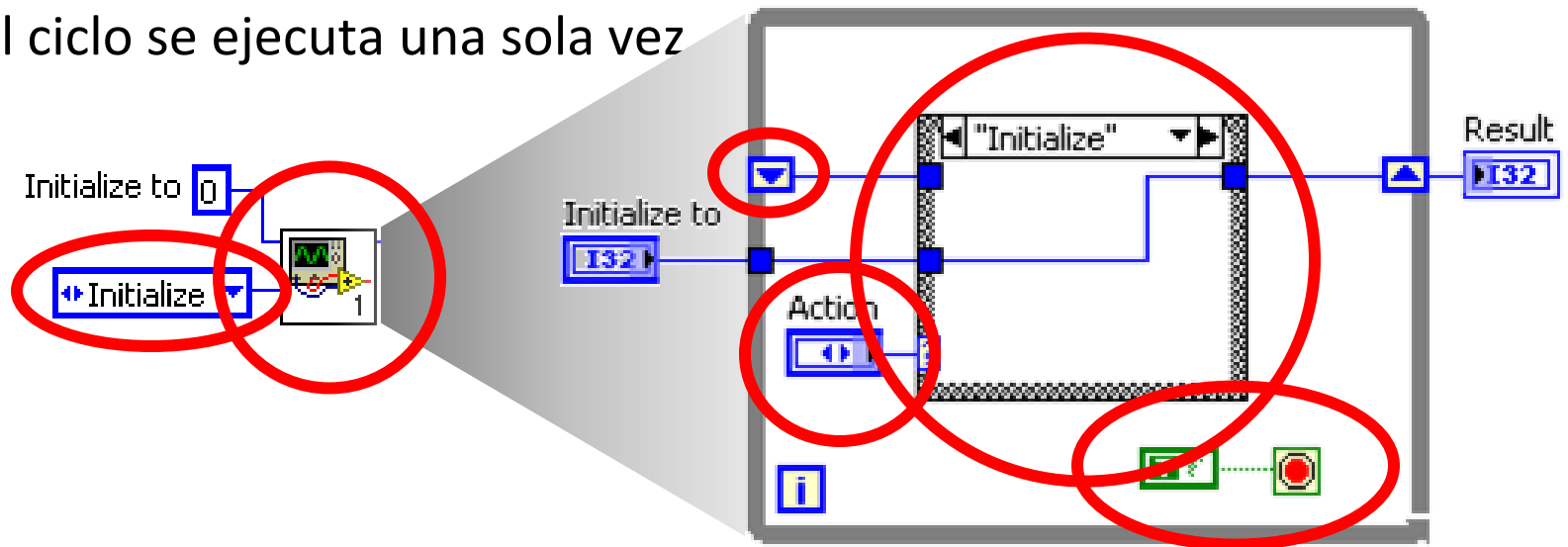
# Herramientas Básicas

- Ciclos
- Registros Corrimientos
- Estructura Casos
- Constantes Enum
- Estructura Eventos



# Variable Global Funcional

1. La Variable Global Funcional es un SubVI **No-Reentrante**
2. Se puede hacer acciones sobre los datos
3. La constante Enum selecciona la acción
4. Guarda el resultado en un registro de corrimiento sin inicializar
5. El ciclo se ejecuta una sola vez

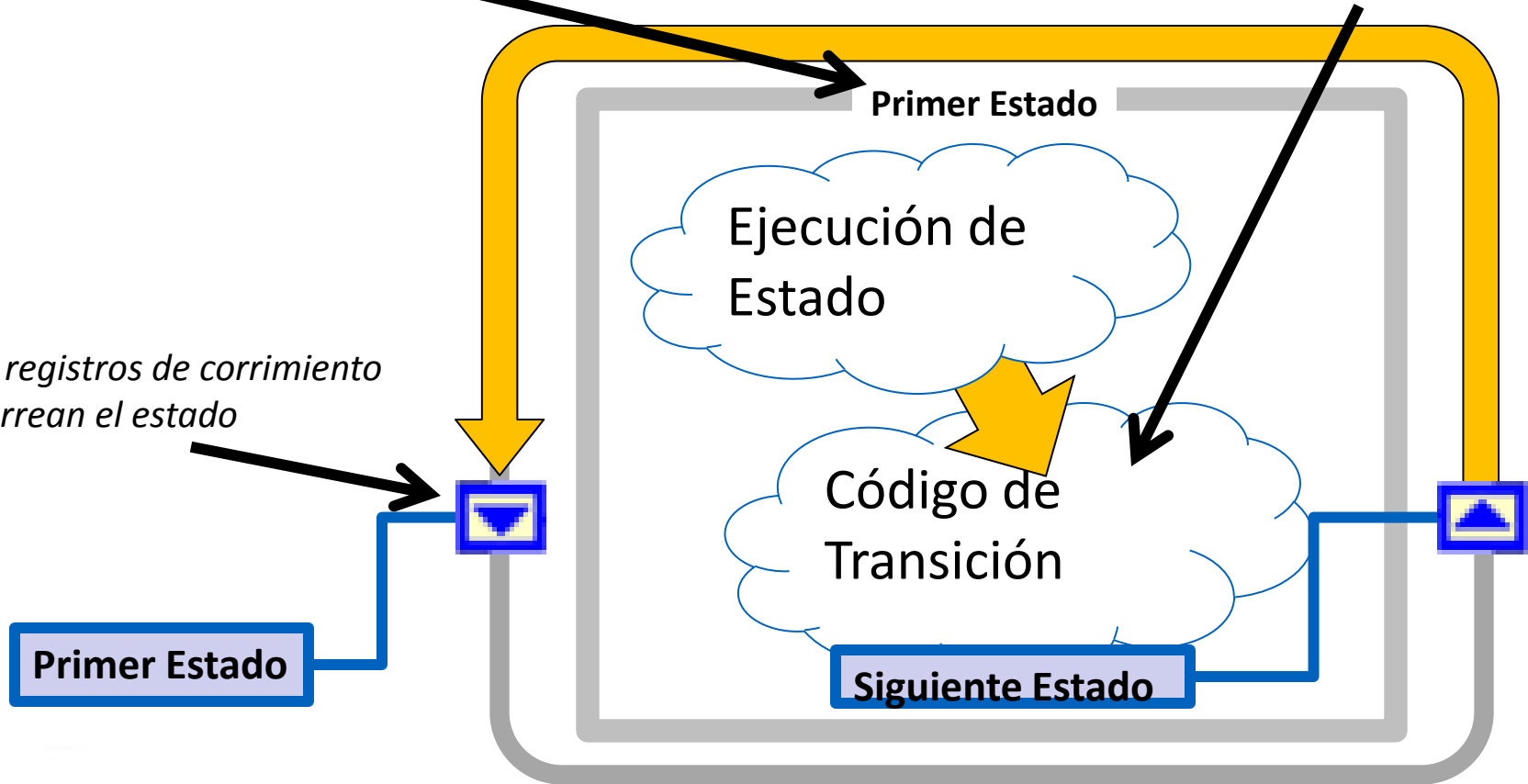


# Máquina de Estados

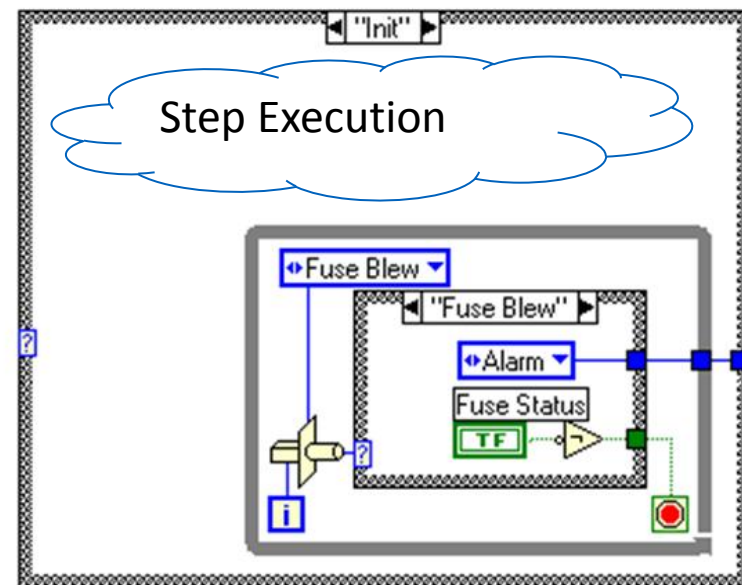
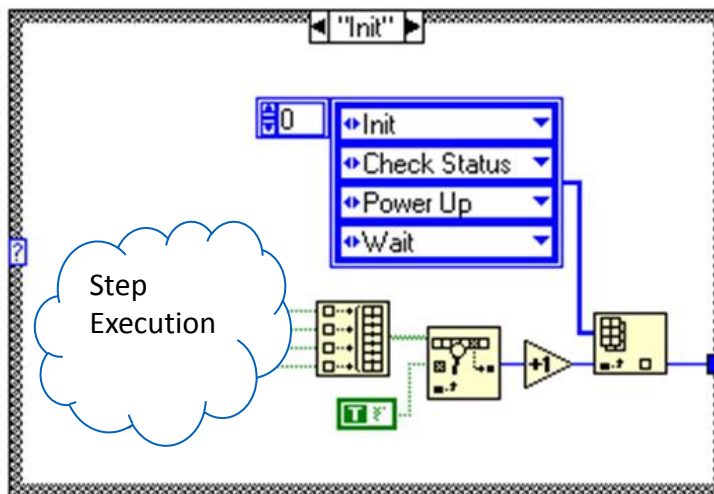
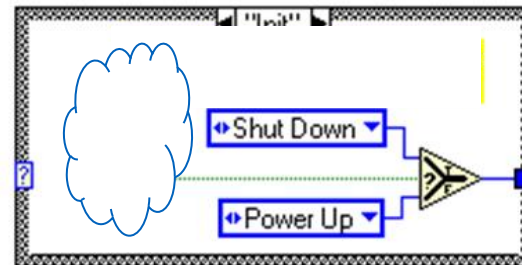
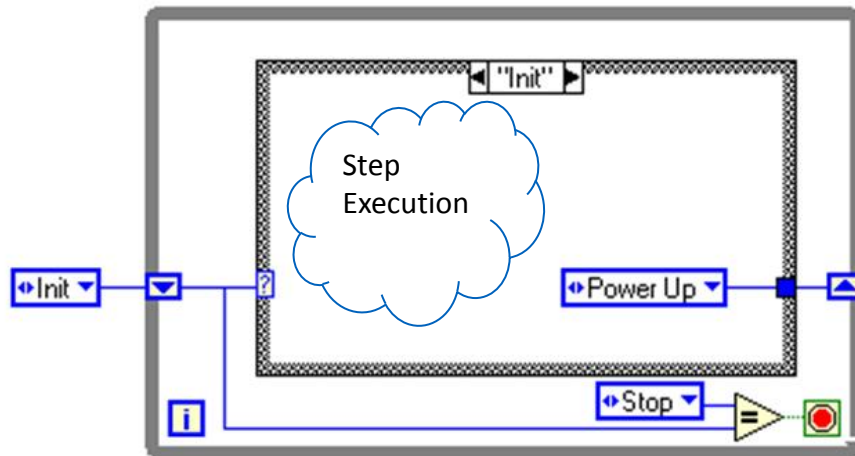
*Se tiene un caso por cada estado*

*El código de transición determina el siguiente caso según los resultados*

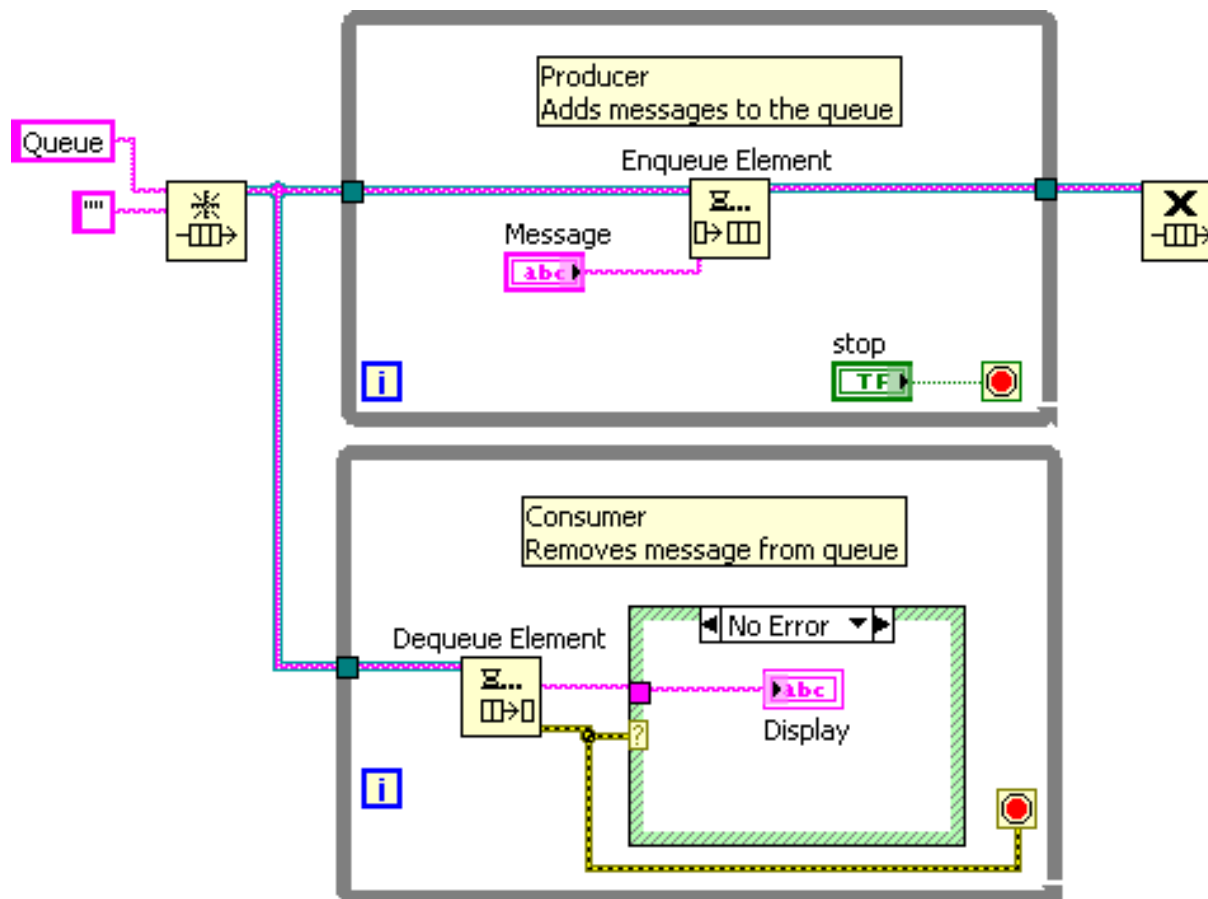
*Los registros de corrimiento acarrear el estado*



# Opciones de Código de Transición



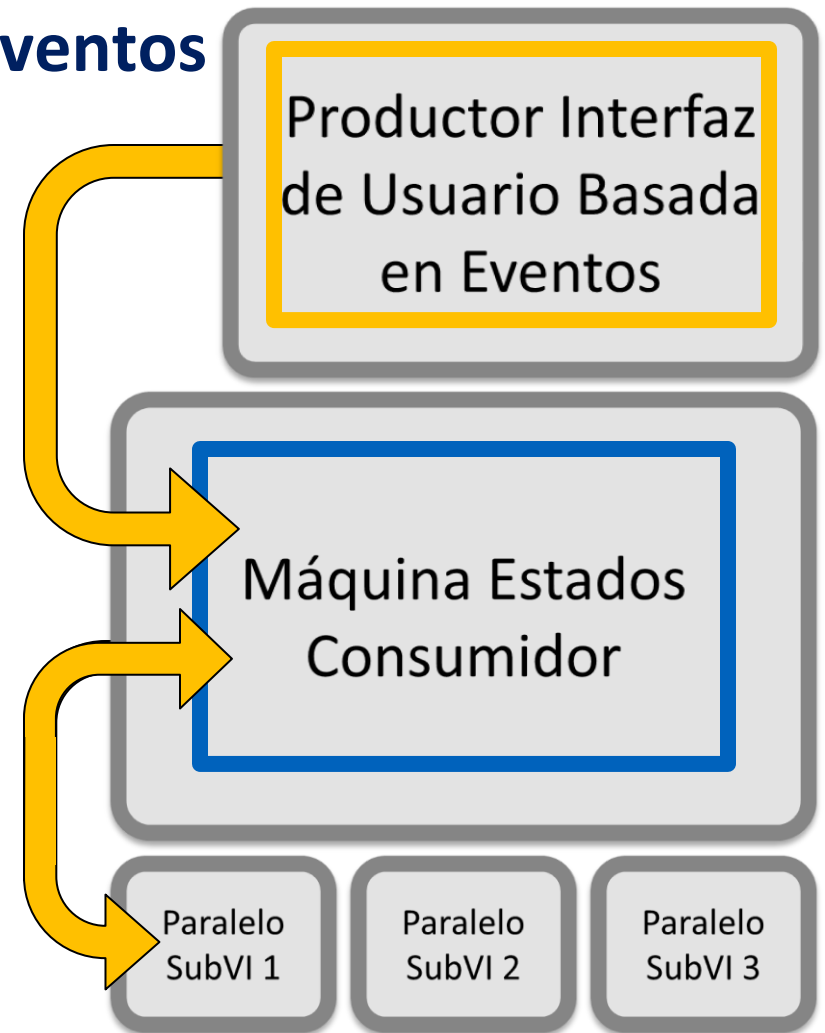
# Producer / Consumidor





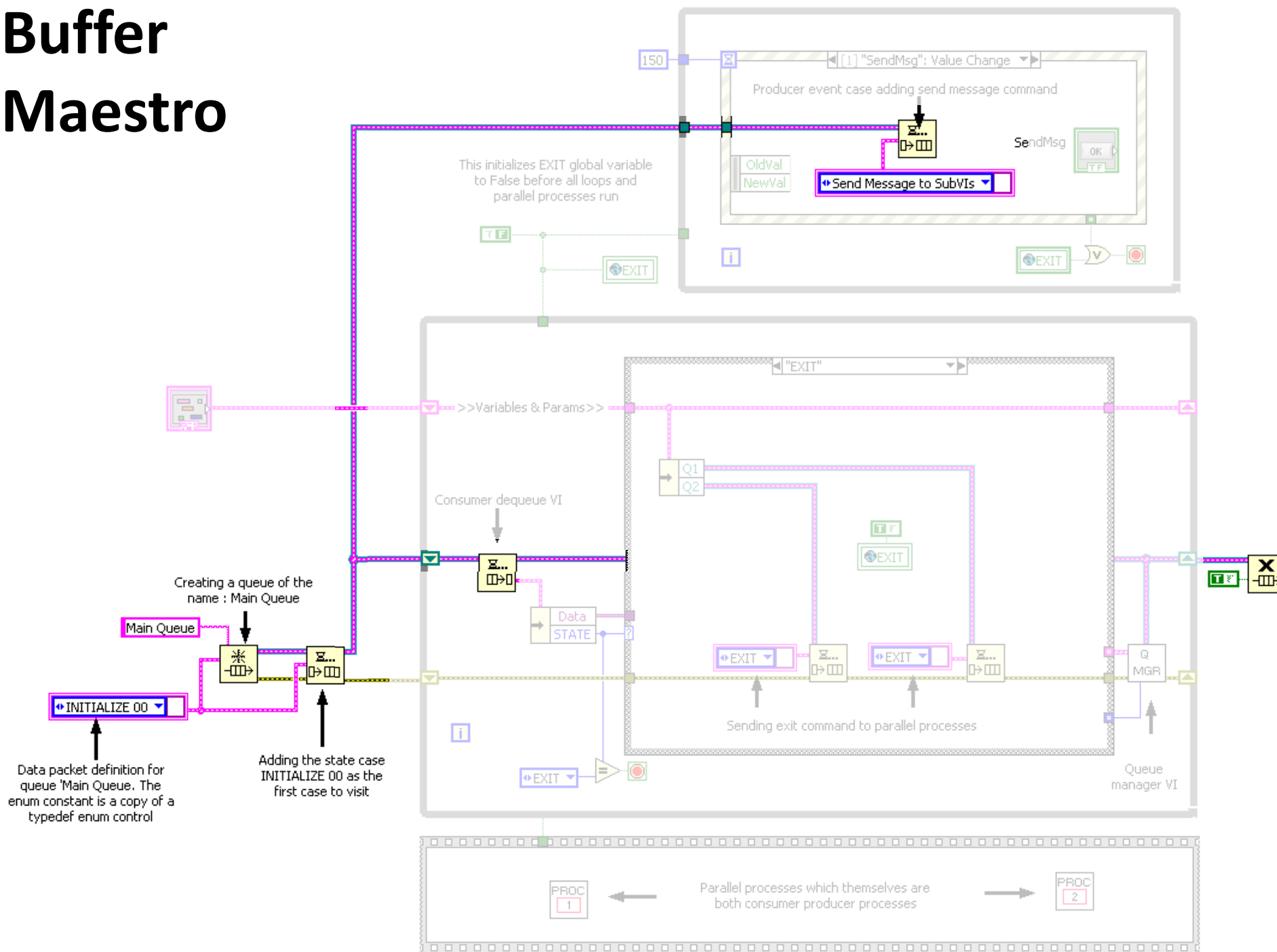
# Máquina Estados Buffer & Productor/Consumidor con Eventos

1. Eventos son capturados por el productor
2. Productor pone los datos en el buffer
3. La máquina de estados en el consumidor ejecuta las acciones sobre los datos
4. SubVIs paralelos se comunican usando referencias a los buffers



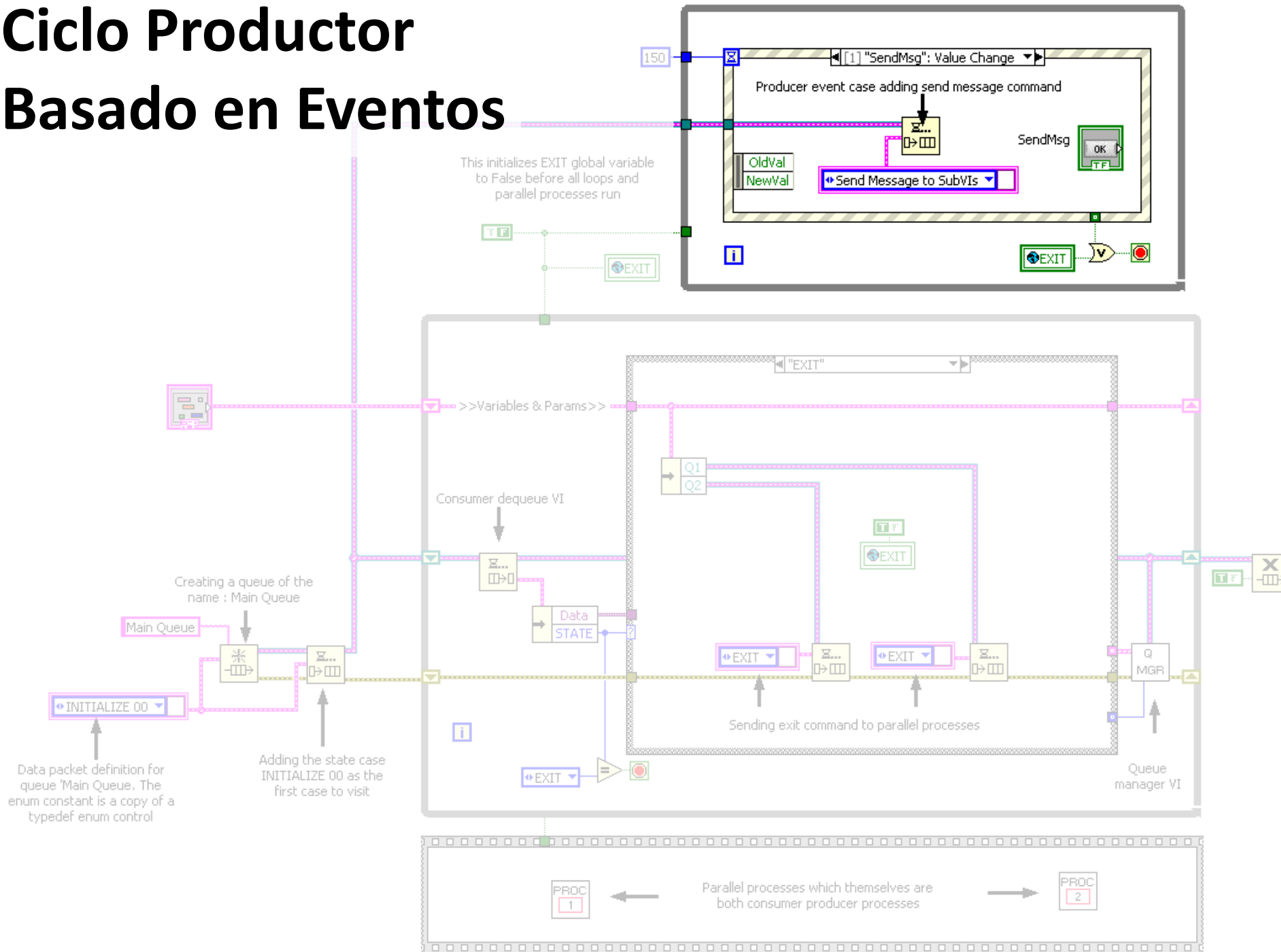


# Buffer Maestro

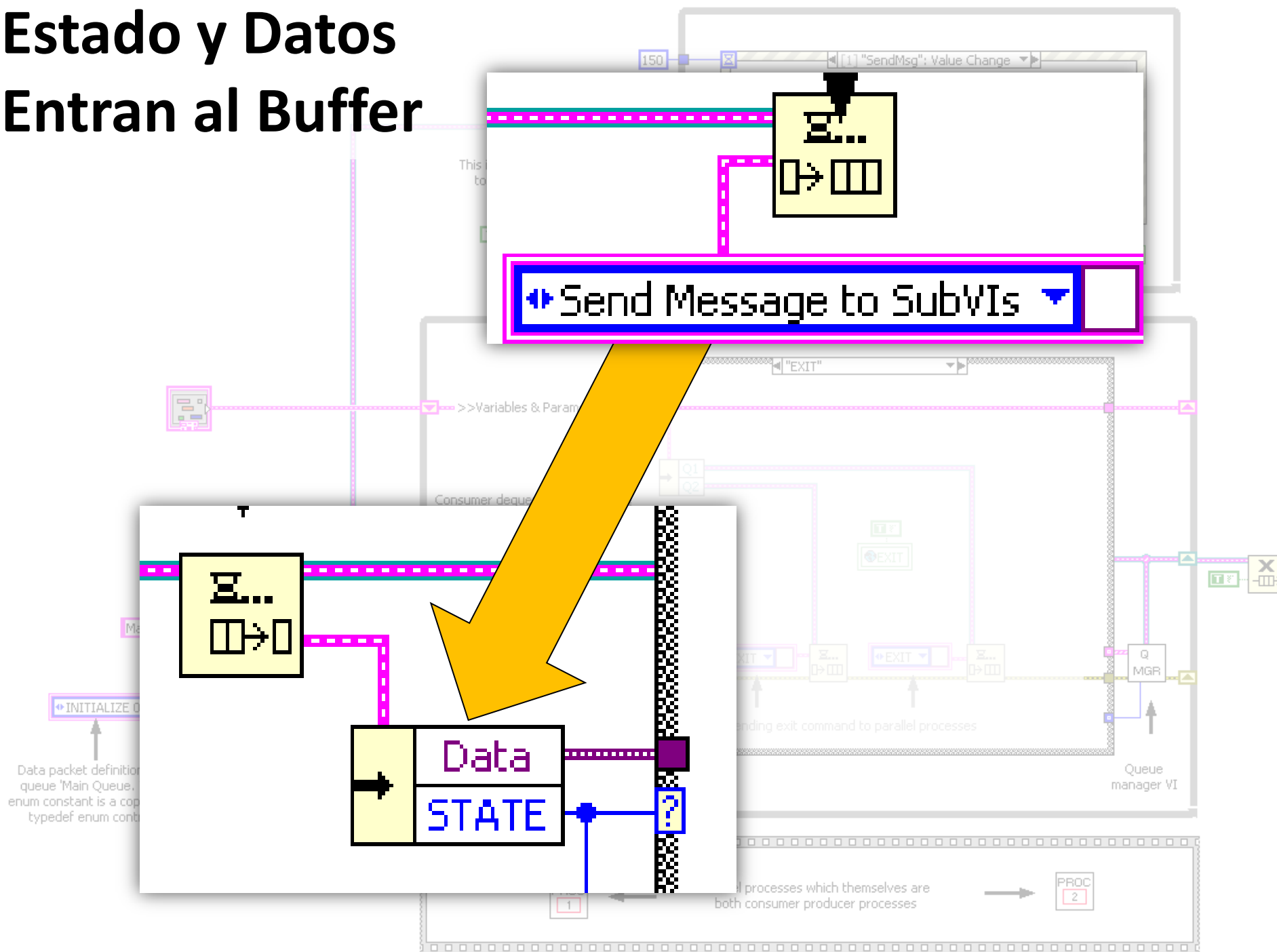


# Ciclo Productor

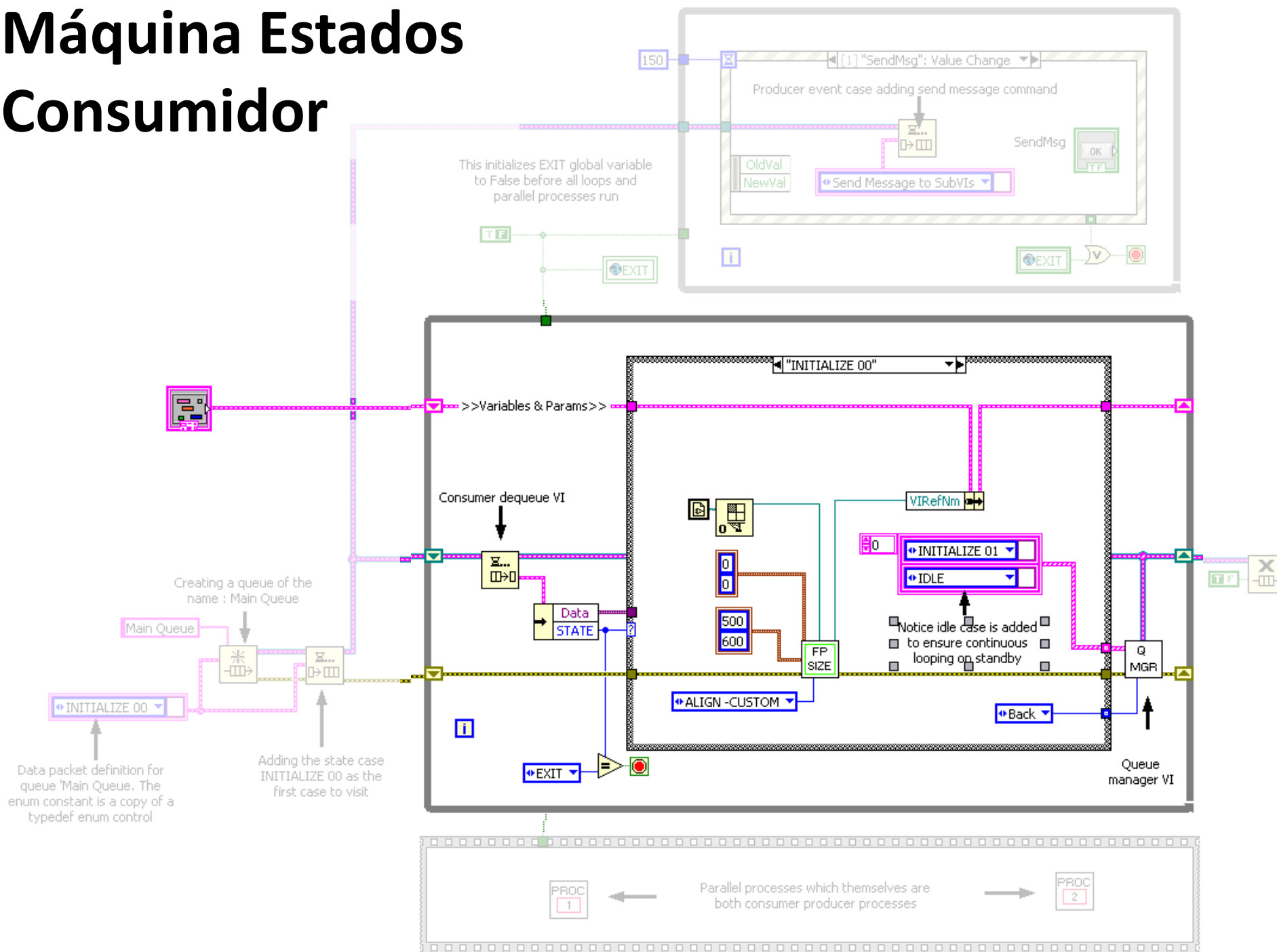
## Basado en Eventos



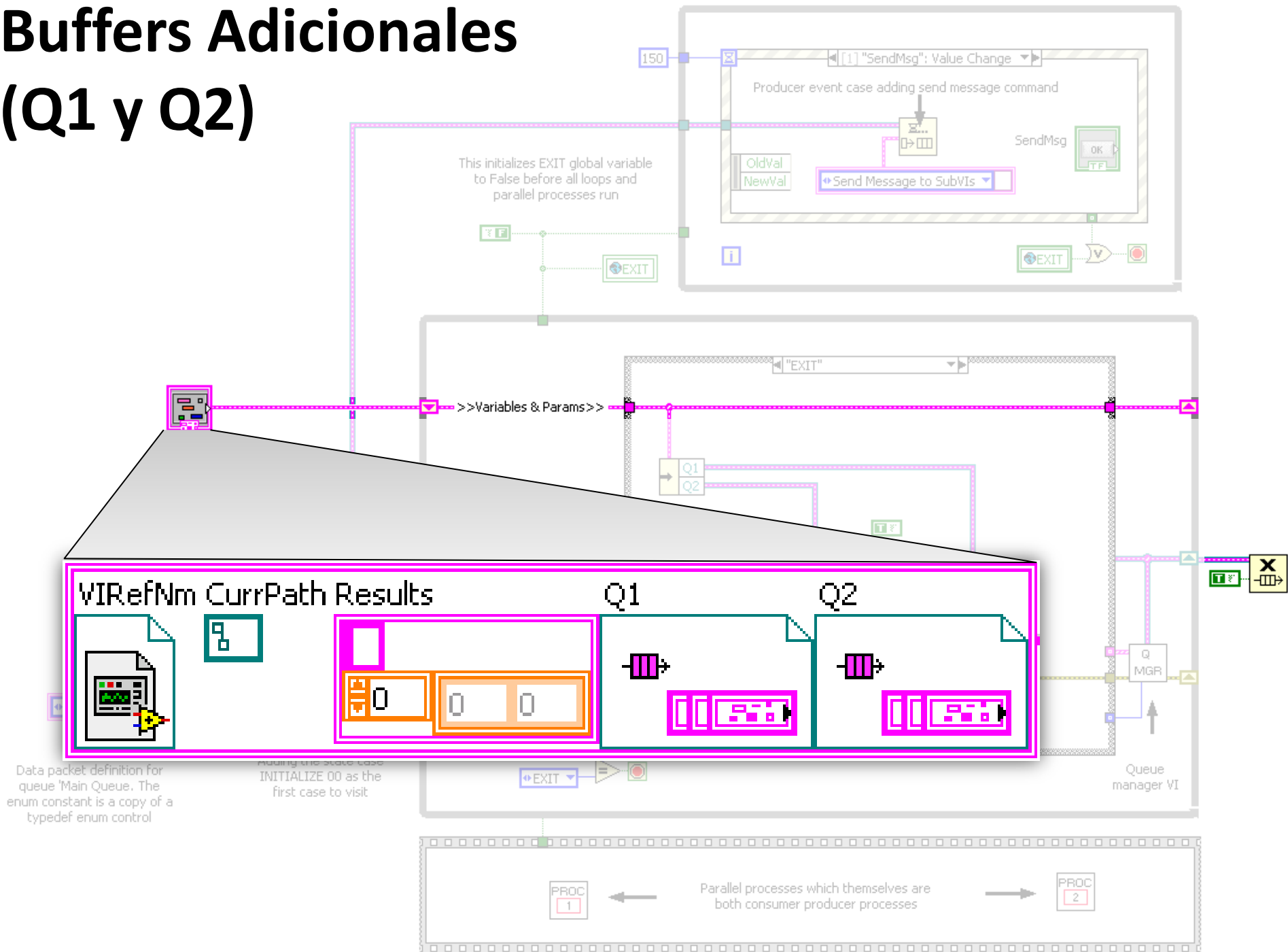
# Estado y Datos Entran al Buffer



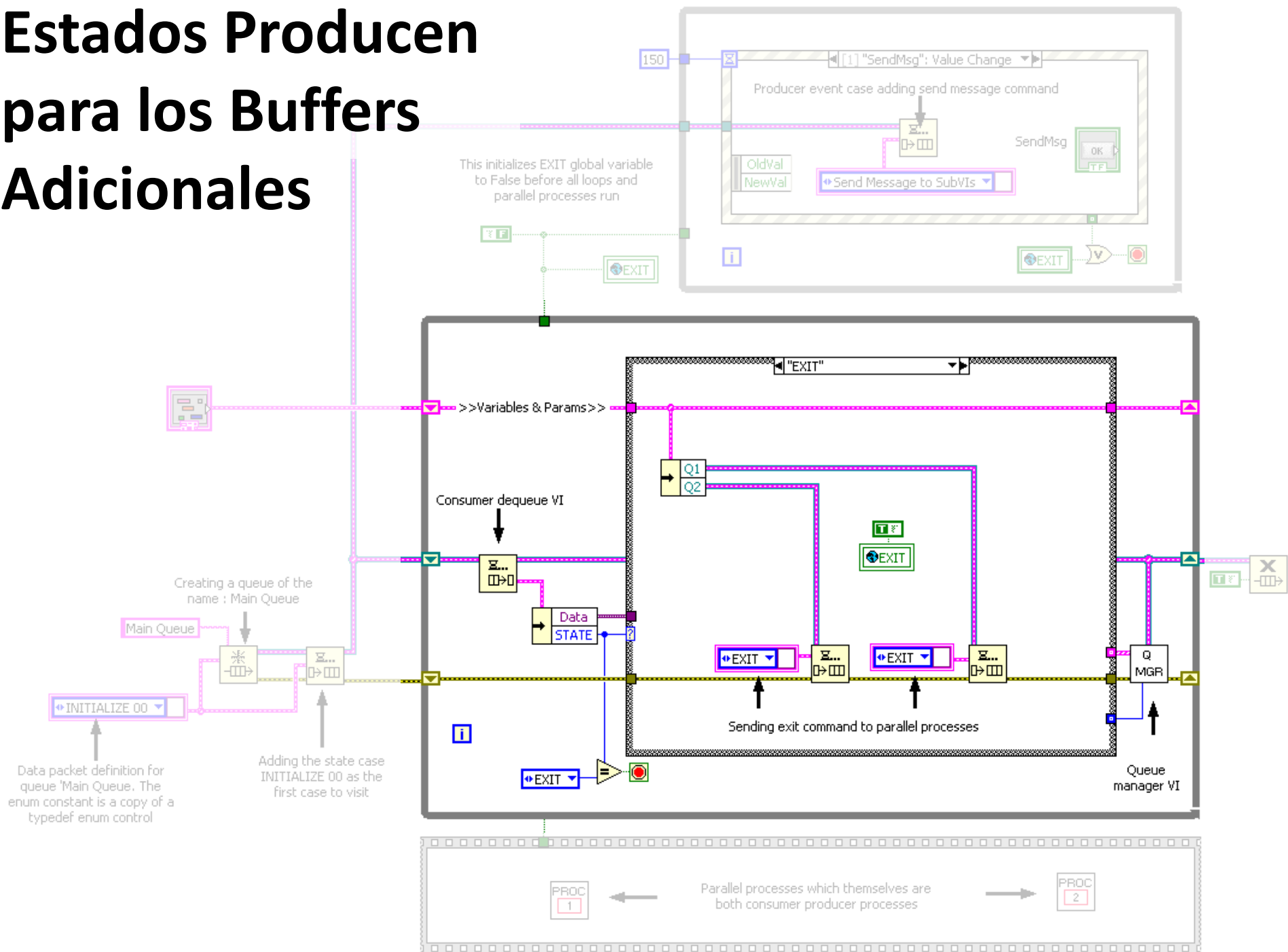
# Máquina Estados Consumidor



# Buffers Adicionales (Q1 y Q2)

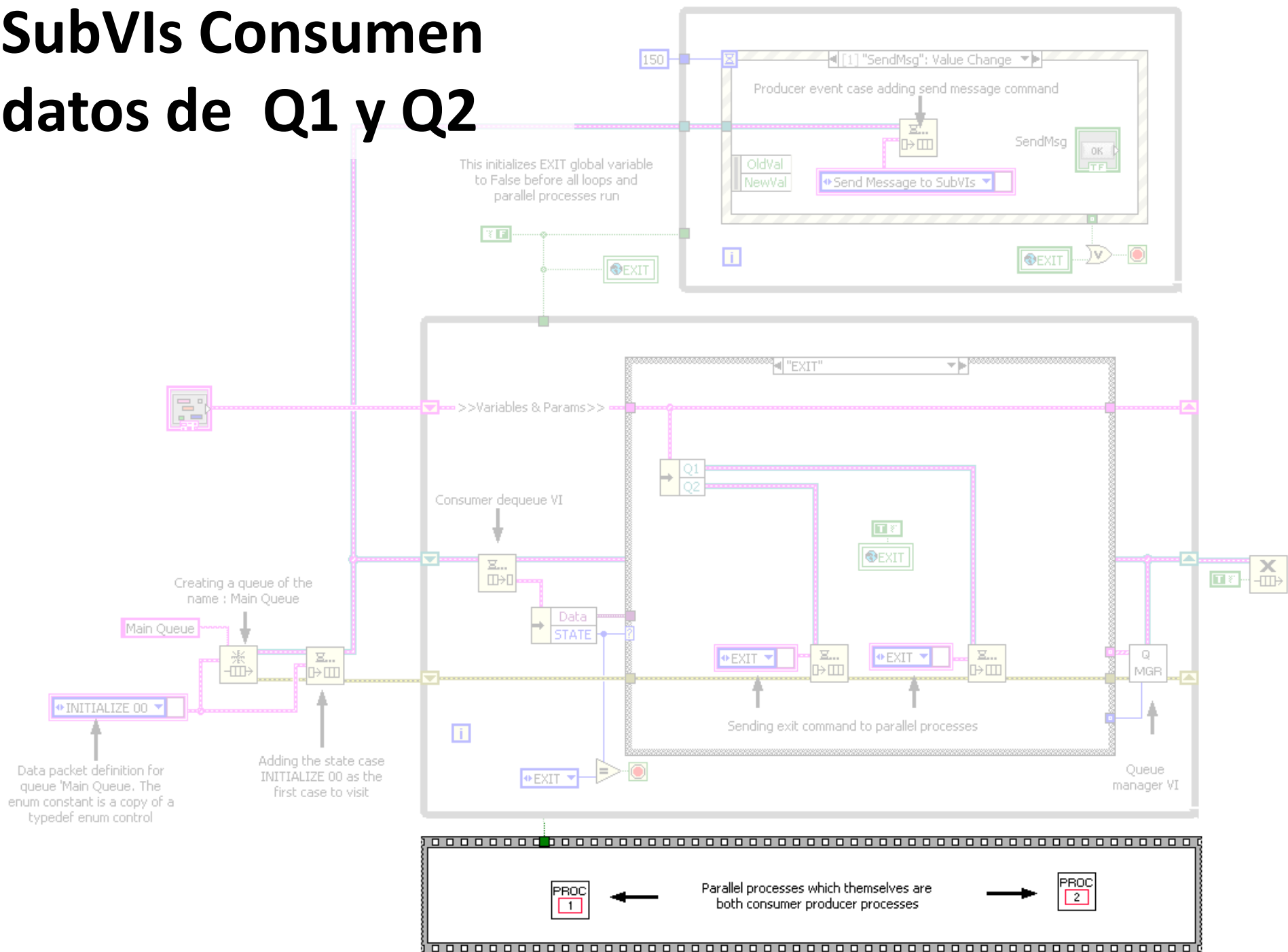


# Estados Producen para los Buffers Adicionales





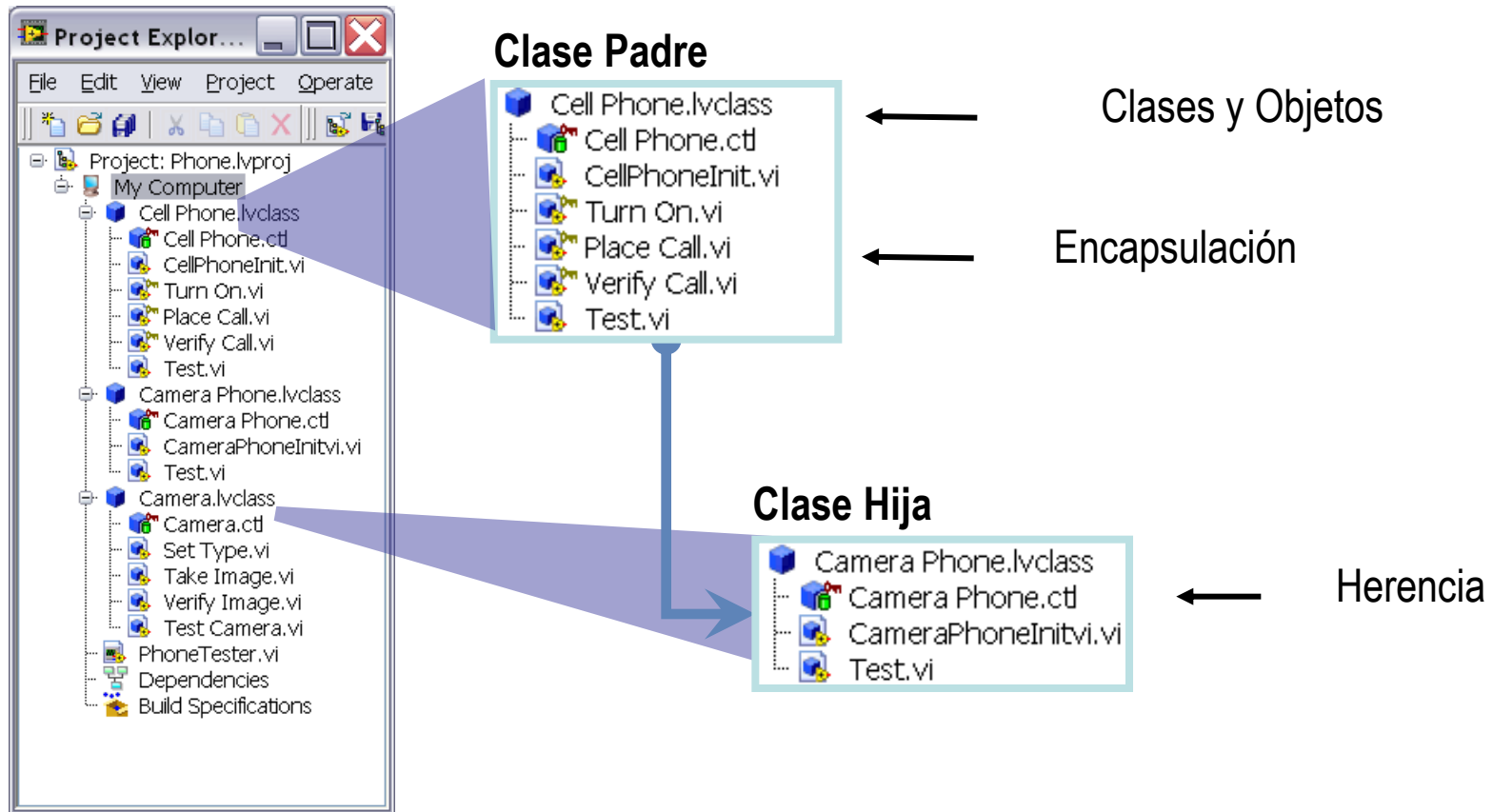
# SubVIs Consumen datos de Q1 y Q2



# Programación Orientada a Objetos

- Un enfoque para el desarrollo de aplicaciones
- Apropiado para aplicaciones de gran escala con un equipo de desarrolladores
- Promueve reutilizar el código
- Reduce el mantenimiento de código
- Simplifica extender aplicaciones

# Programación Orientada a Objetos



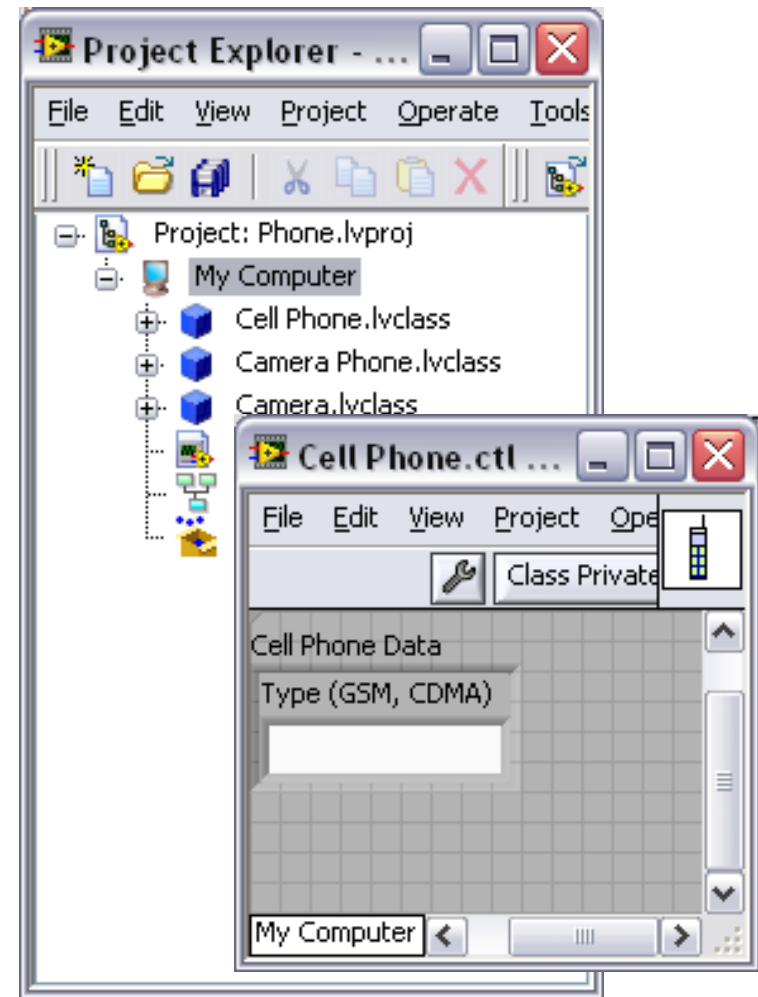
- Para desarrollo de aplicaciones a gran escala

# Clases y Objetos

- Los Objetos son actores en su aplicación
  - Referencias a partes individuales de datos
- La Clase define los datos y comportamiento de los Objetos
  - Los objetos en su aplicación son instancias de una Clase

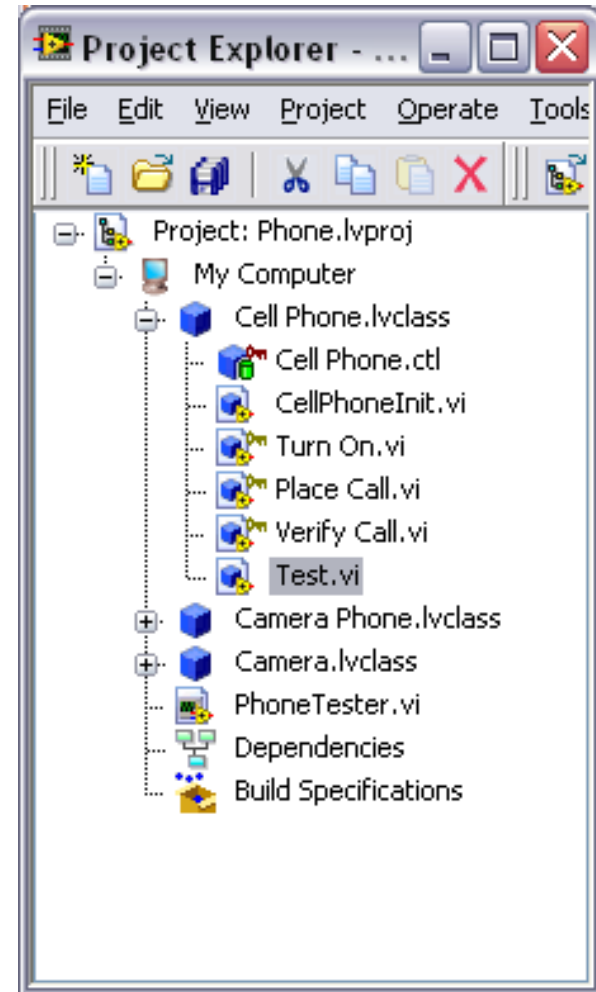
# Creando Clases en LabVIEW

- Crear una clase en el Proyecto
- Especificar los datos con el .ctl para la clase
  - Definir la clase en efecto define un nuevo tipo de dato
- Características adicionales
  - Especificar el ícono de la clase
  - Especificar una plantilla de ícono de VI
  - Especificar el color del cable



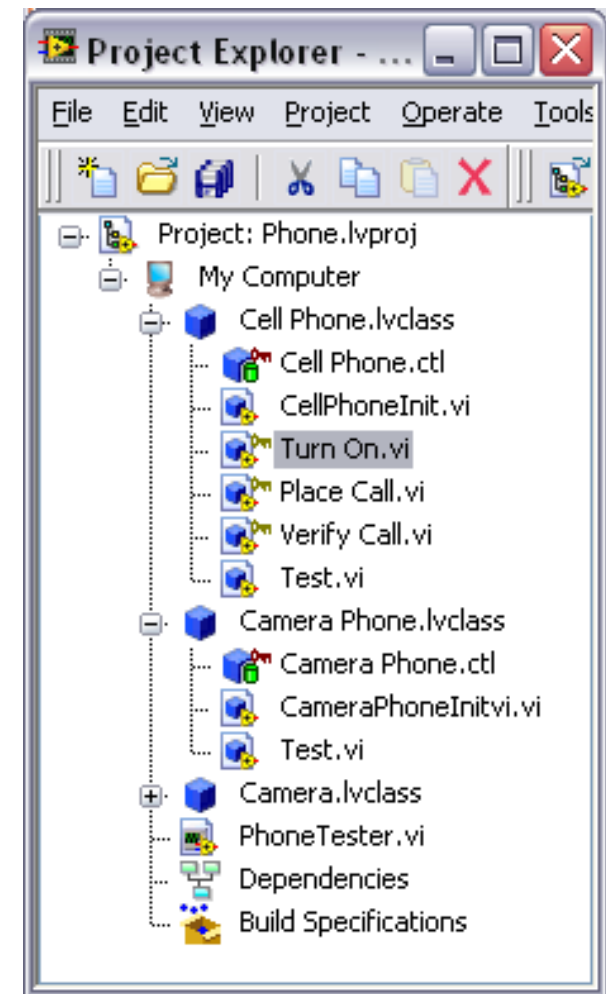
# Creando Métodos para las Clases

- Métodos
  - Acciones o peticiones
  - Desempeñadas por los objetos
  - Generalmente verbos
- Crear un VI
- Especificar alcance
  - Público
  - Privado
  - Protegido



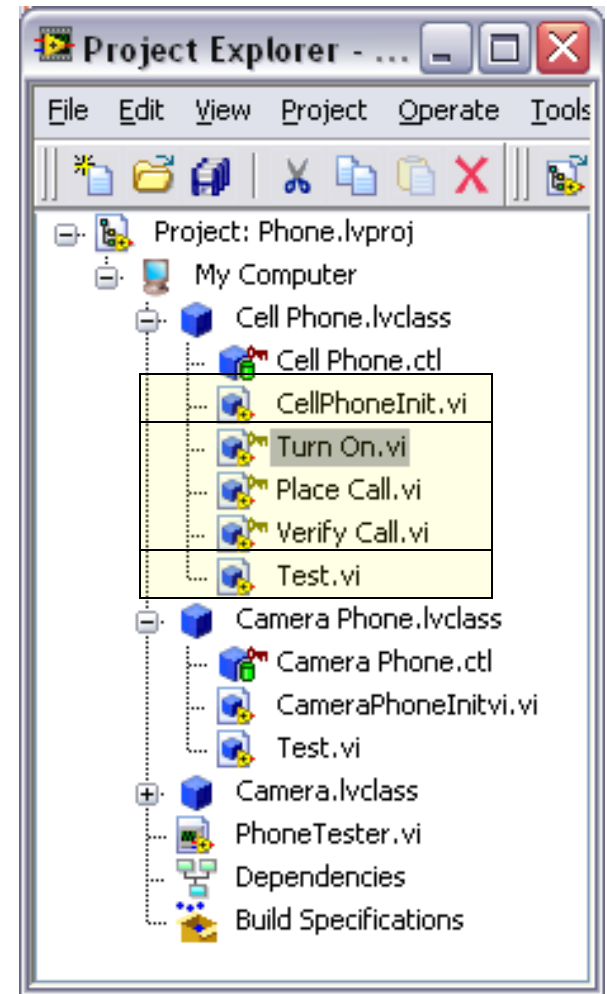
# Herencia

- Define subclases
- Crea una relación de pertenecía
  - Ejemplo: Teléfono con Cámara “es un” Teléfono Celular
  - Reutiliza funcionalidad común
- Especialización
  - Extender o sobrescribir funcionalidad para necesidades específicas



# Encapsulación

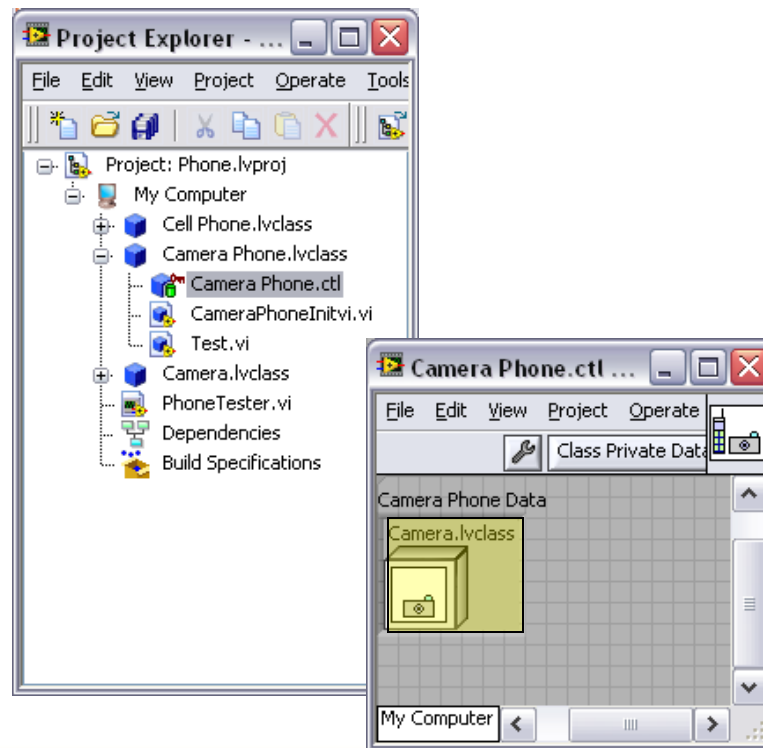
- Tratar cada objeto como una caja negra
  - Interfaz bien definida de datos y métodos
  - Debe utilizar esta interfaz en la aplicación
- Todos los datos son privados
- Métodos pueden ser públicos, privados o protegidos





# Composición de la Clase

- Definir una clase crea un nuevo tipo de dato
- Una clase puede estar hecha de otra clase



# Aplicaciones de LabVIEW Usando Clases

- Llaman métodos y objetos
- Reduce la rescritura de código mediante la herencia y envío dinámico

# ¿PREGUNTAS?

Felipe Rincón

Field Sales Engineer  
National Instruments Andean & Caribbean

[felipe.rincón@ni.com](mailto:felipe.rincón@ni.com)

01 800 010 0793

# Academic Days 2011

Horario	Actividad		
8:00- 8:30 am	Inscripciones		
8:30- 9:15 am	Introduccion y presentacion		
9:15- 10:00 am	Seminario Teorico: Sistemas PXI para pruebas, control y diseño en academia e investigacion		
10:00- 10:30 am	Coffee Break		
10:30-11:00	Caso de Éxito		
11:00- 11:45 am	Seminario Teorico: Diseño y simulacion y prototipos de control y Como utilizar codigo matematico .m con LabVIEW		
11:45-12:15pm	Exposicion Grupo Siatec		
12:15- 1:45 pm	Almuerzo		
1:45-2:30 pm	Seminario Teorico: Simulaciones Mecanicas con Solidworks y LabVIEW	Seminario Practico LabVIEW 2010	Seminario Practico NI ELVIS II
2:30-3:15 pm	Seminario Teorico: Tecnicas de Programacion con LabVIEW		
3:15-3:45 pm	Cierre y entrega de premios		