LabVIEW Lesson Day2

Joe Kao
Application engineer, eCloudValley

2021/1/25

Communicating Data Between Parallel Loops

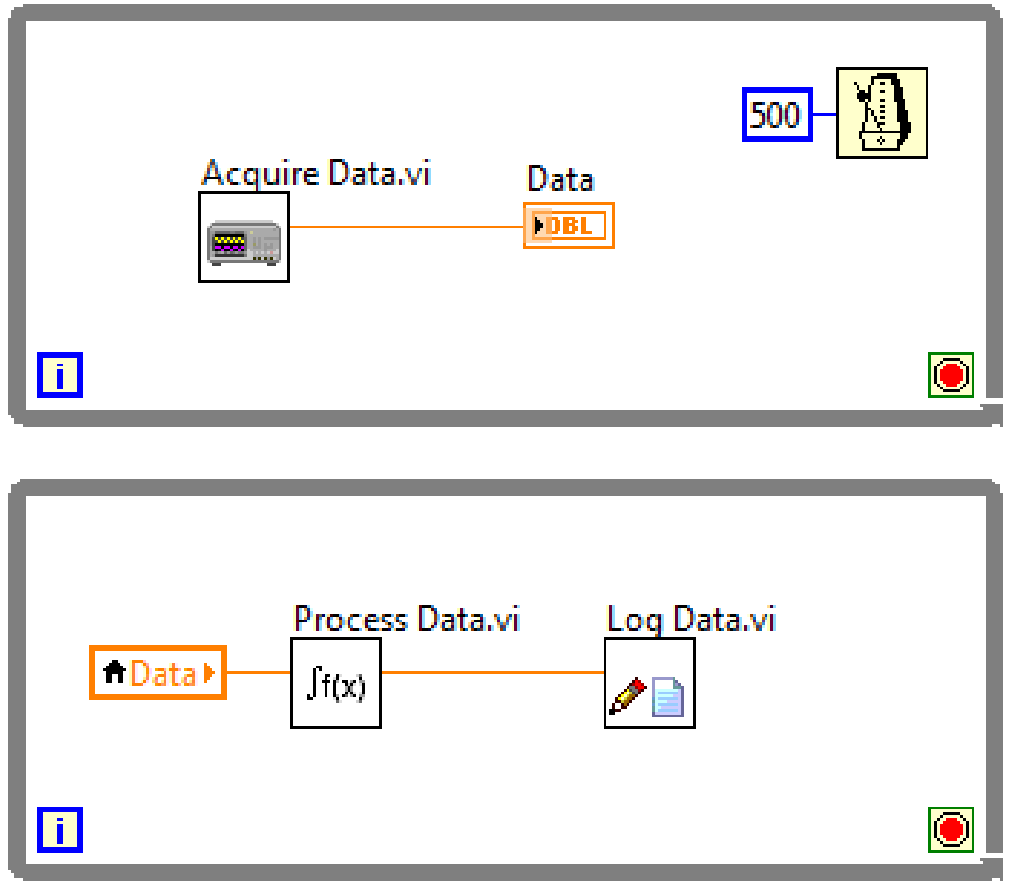
- Develop code that synchronizes data between parallel loops.
  - Introduction
  - Queues
  - Notifiers
  - Summary

A. Introduction

- Review using local variables to communicate between parallel loops.

Communicating Data Between Parallel Loops – Local Variables

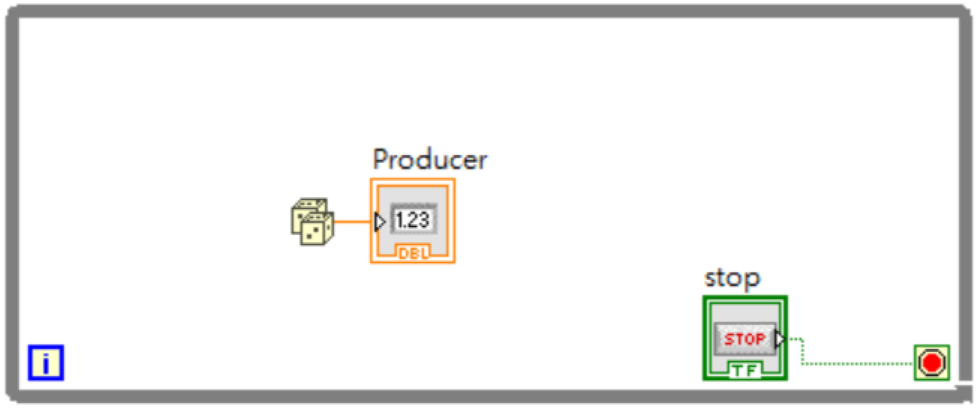
Local variables share the latest data between parallel loops.

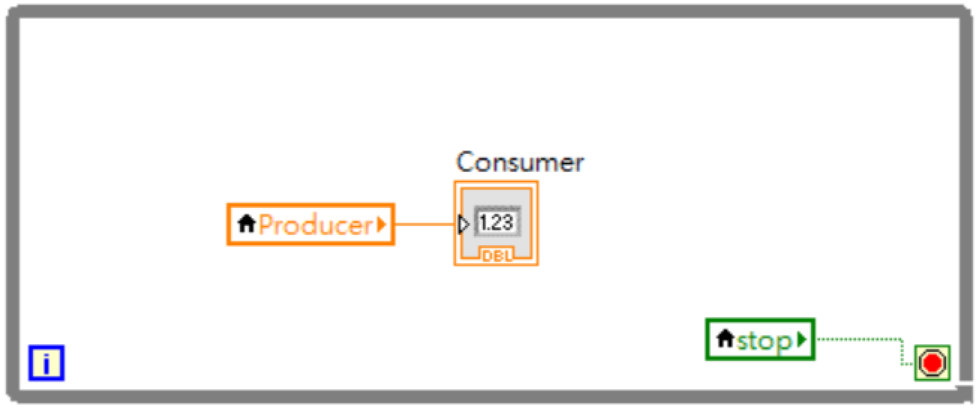


Exercise
Producer/Consumer Loop with Local Variable

- 建立二個平行的While Loop，上面為Producer Loop，下面為Consumer Loop
- 在Producer Loop放置亂數產生器，連接上Indicator，並建立Indicator的Local Variable
- 將Local Variable 放入Consumer Loop，Change to Read後連接上另一個Indicator
- Run VI，觀察二個Indicator的數值

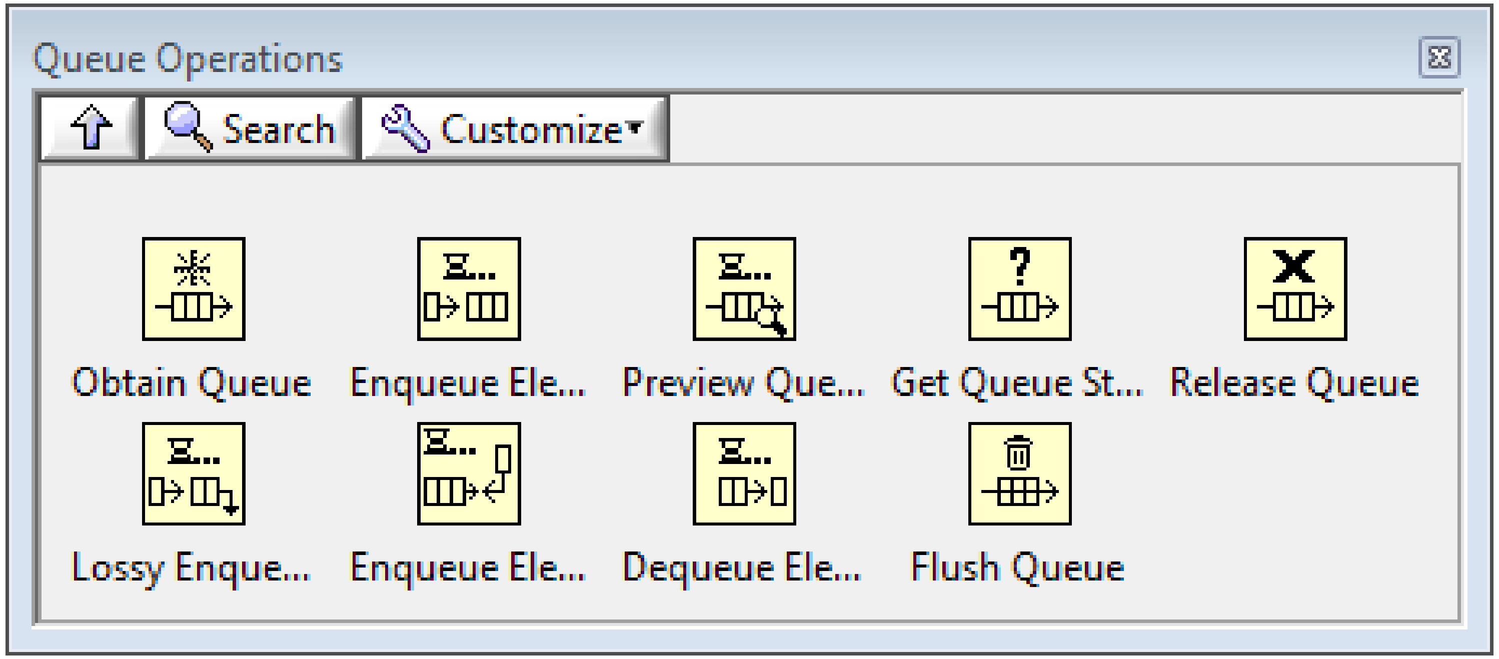
Exercise
Producer/Consumer Loop with Local Variable





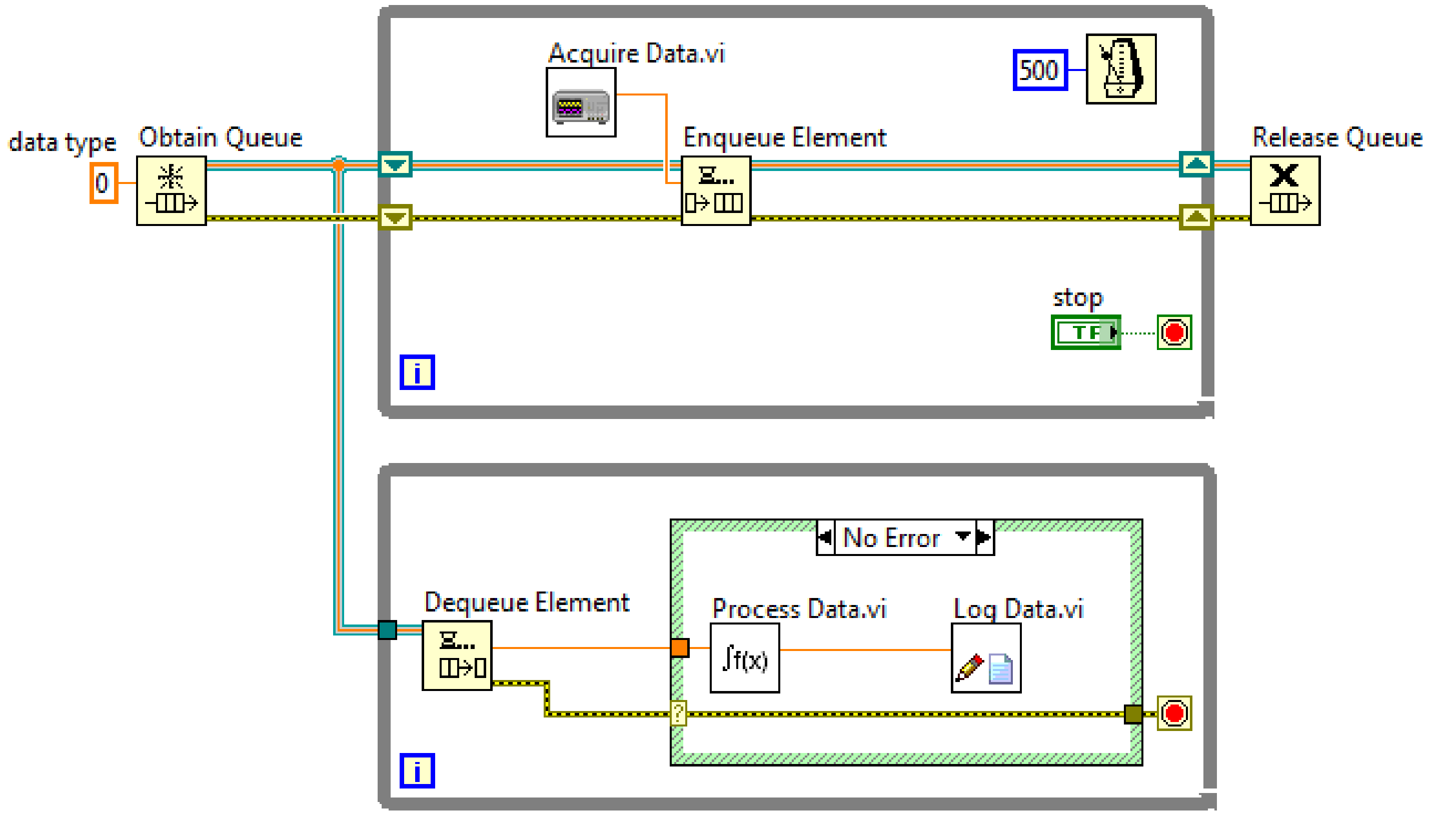
B. Queues

- Demonstrate how to transfer every point of data between parallel loops using queues.



Communicating Data Between Parallel Loops – Queues

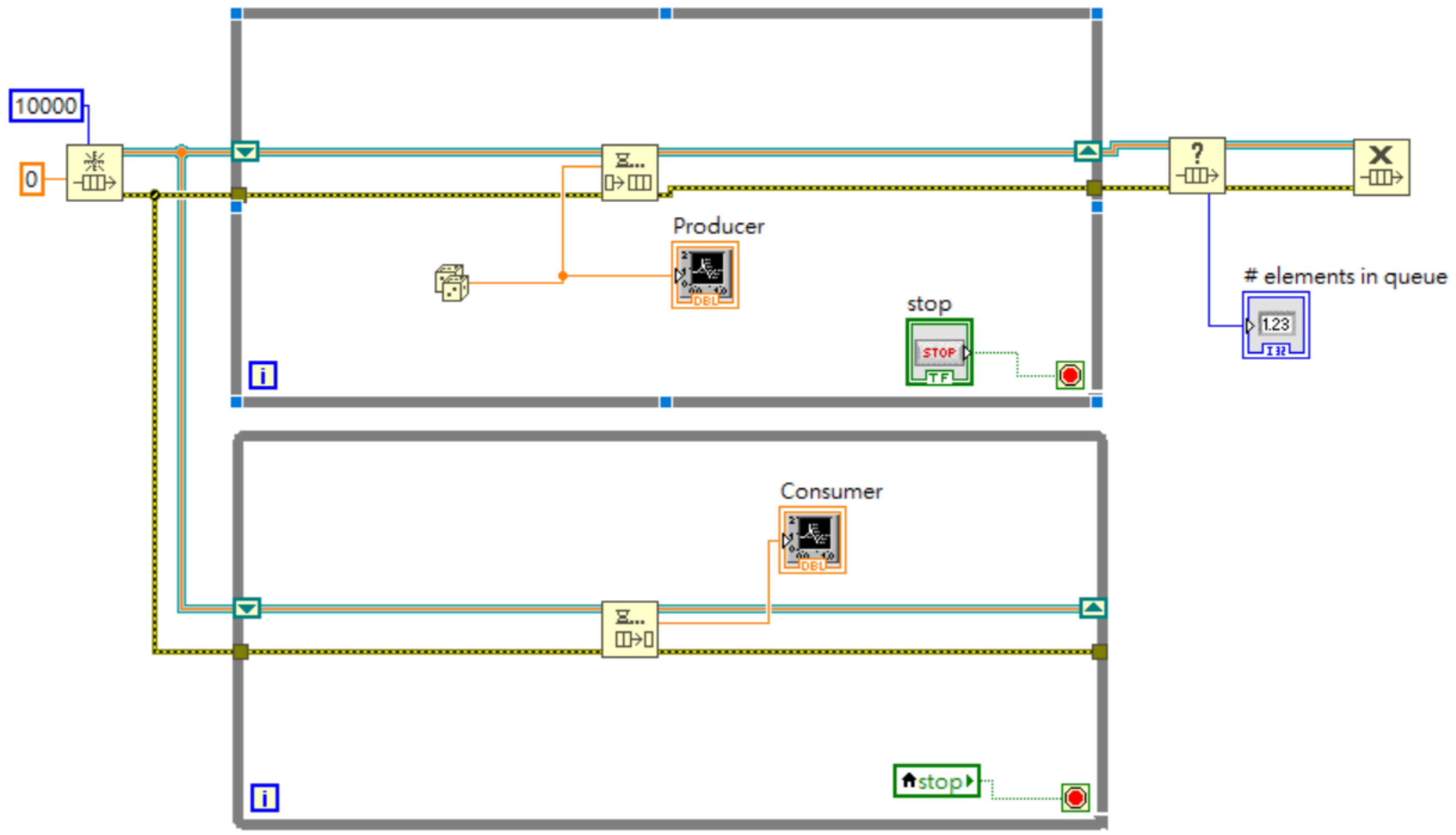
Transfer every point of data between parallel loops or VIs



Exercise
Make a Queue Structure

- 建立2個While Loop (Producer and Consumer)
- 設置Obtain Queue, Enqueue, Dequeue, Get queue status
- Producer 放入亂數產生
- Consumer 讀出亂數

Exercise
Make a Queue Structure



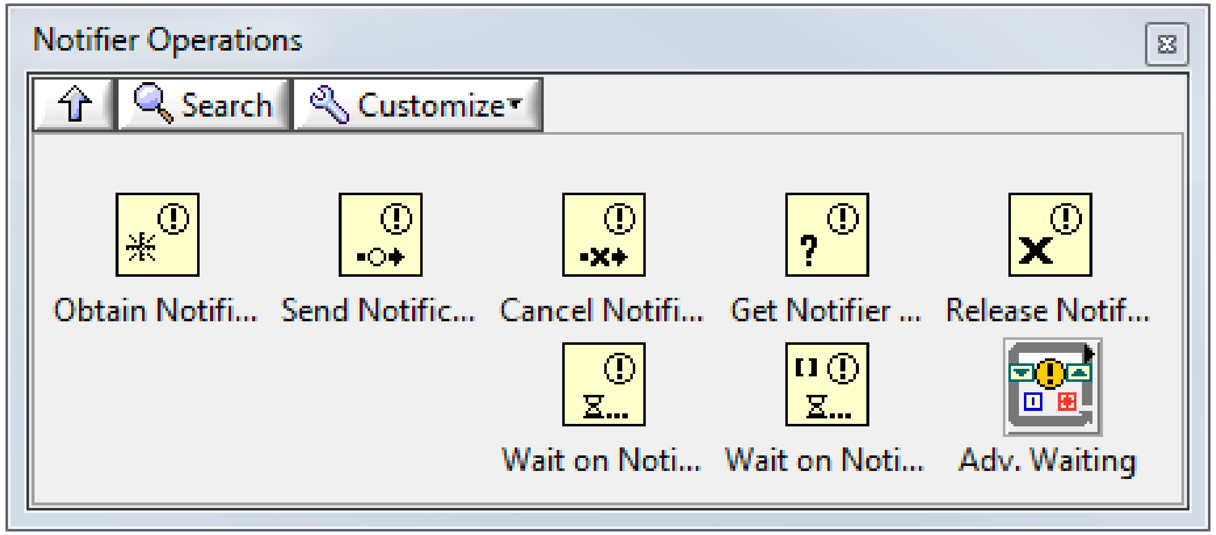
C. Notifiers

- Create code that broadcasts the latest data to waiting parallel loops using notifiers.

  - Notifiers Operations Palette
  - Using Notifiers

What is a Notifier?

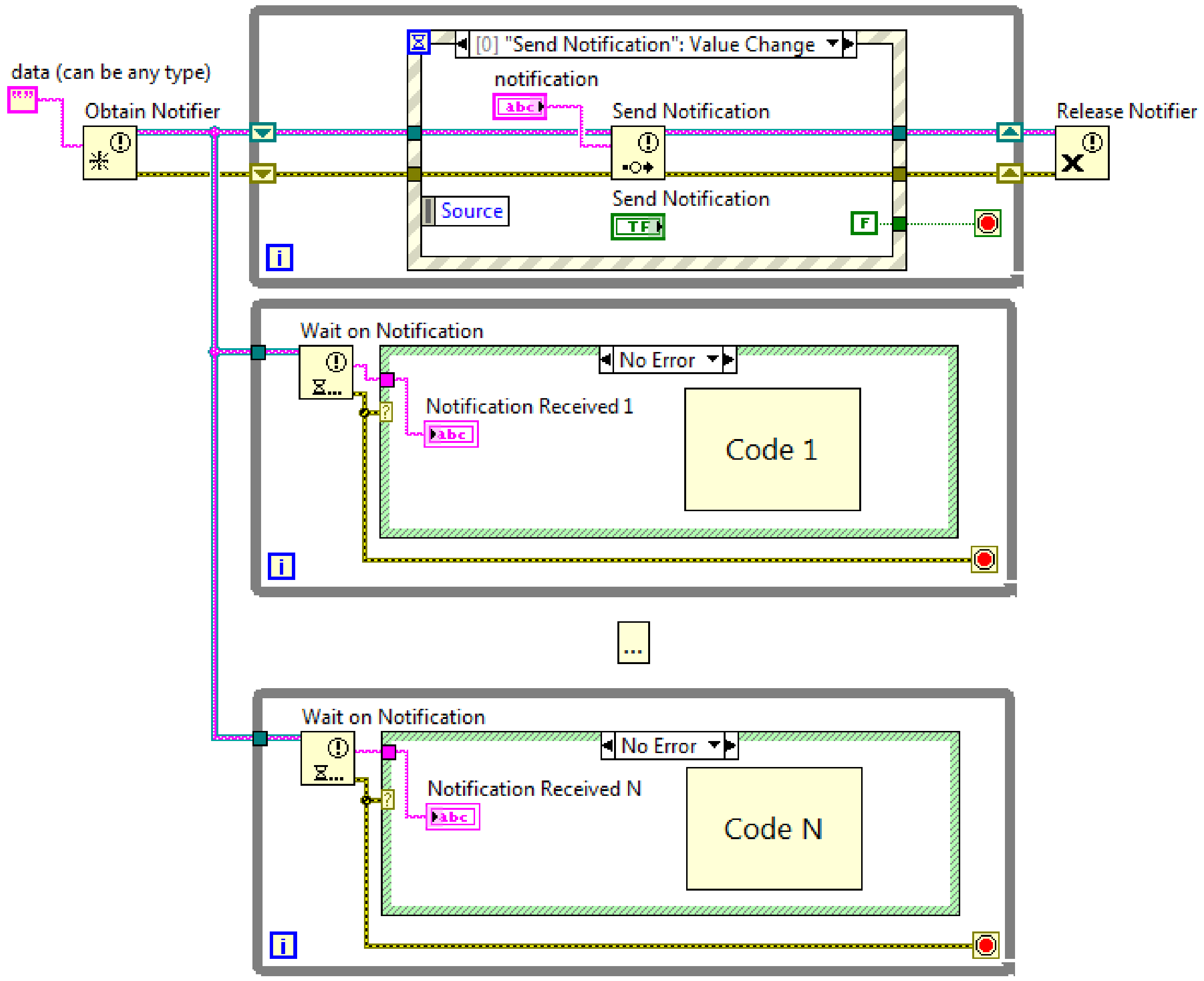
Notifier functions—Suspend the execution of a block diagram until it receives data from another section of the block diagram or from another VI running in the same application instance



Notifiers vs. Queues

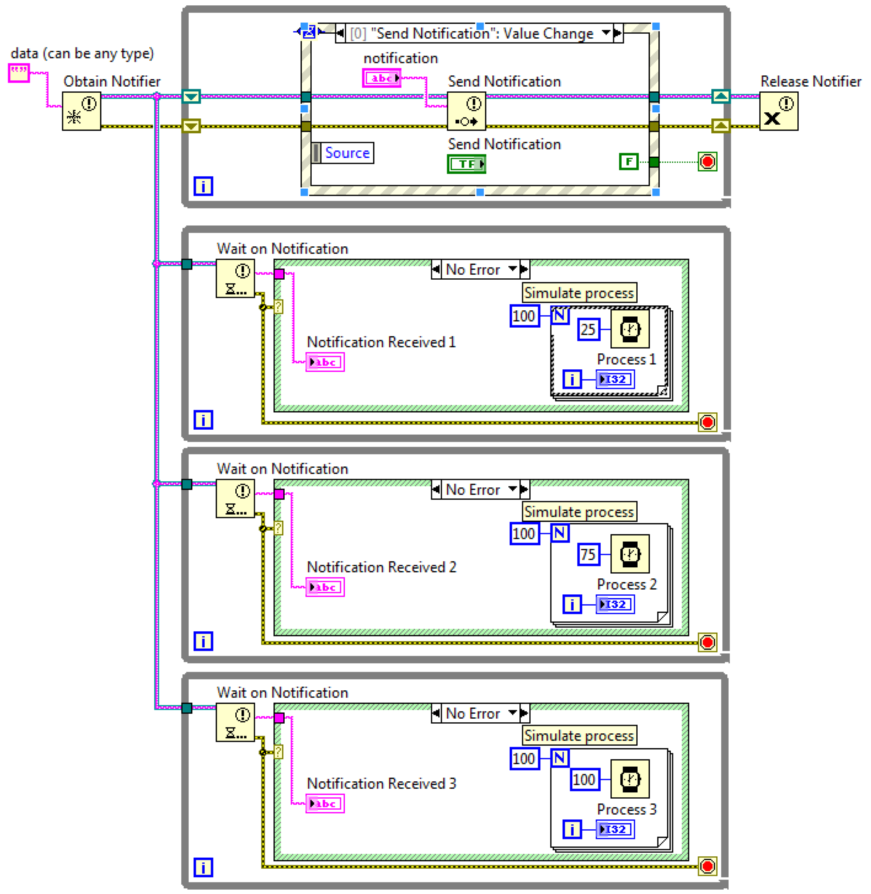
- Notifiers do not buffer data, so receiving loops only receive the latest notification
- Notifiers can broadcast a notification to multiple loops

Broadcast Data to Parallel Loops



Demonstration
Notifier

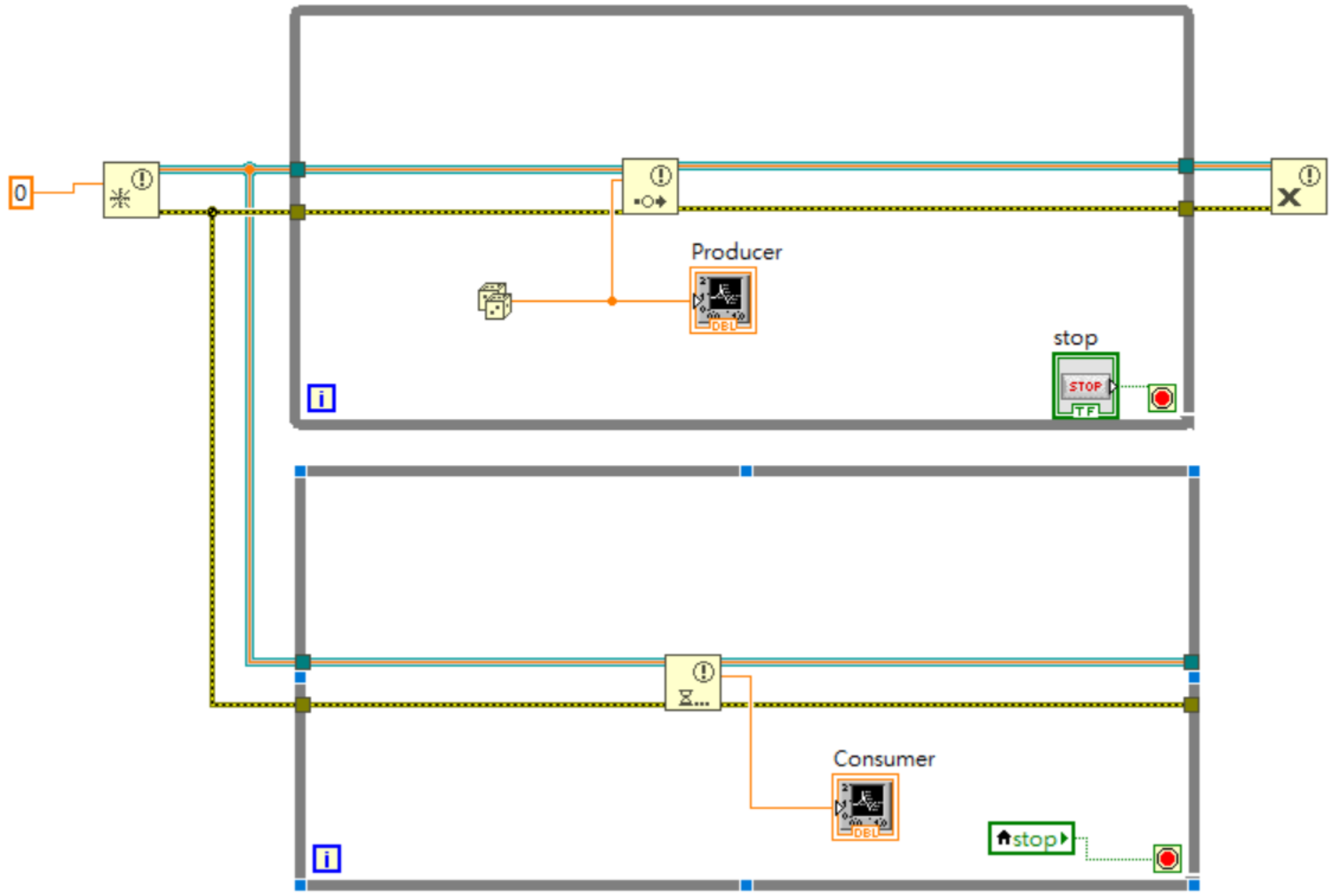
- Broadcast the latest data to waiting parallel loops.



Exercise
Make a Notifier Structure

- 建立2個While loop
- 放置Obtain Notifier, Send Notification, Wait on Notification.(Producer and Consumer Loop)
- Producer loop 輸出亂數
- Consumer loop接收Producer loop 的資料

Exercise
Make a Notifier Structure



D. Summary

- Compare Variables, Queues, and Notifiers

	Suspend execution in reader loop?	Buffers data?	Data can be read by multiple loops?	Use case
Local/global variables			Yes	Transfer latest data
Queues	Yes	Yes		Transfer every point of data
Notifiers	Yes		Yes	Transfer latest data to multiple loops that are waiting on a notification

1. Which of the following buffer data?

a. Queues
b. Notifiers
c. Local Variables

1. Which of the following buffer data?

a. **Queues**
b. Notifiers
c. Local Variables

2. Match the following:

Obtain Queue	a. Destroys the queue reference
Get Queue Status	b. Assigns the data type of the queue
Release Queue	c. Adds an element to the back of a queue
Enqueue Element	d. Determines the number of elements currently in the queue

2. Match the following:

Obtain Queue	b. Assigns the data type of the queue
Get Queue Status	d. Determines the number of elements currently in the queue
Release Queue	a. Destroys the queue reference
Enqueue Element	c. Adds an element to the back of a queue

Implementing Design Patterns

- Implement common design patterns for single and parallel loop applications.

A. Why Use Design Patterns?

B. Simple Design Patterns

C. Multiple Loop Design Patterns

D. Functional Global Variable Design Pattern

E. Error Handlers

A. Why Use Design Patterns?

Design Patterns—Code implementations and techniques that are solutions to specific problems in software design

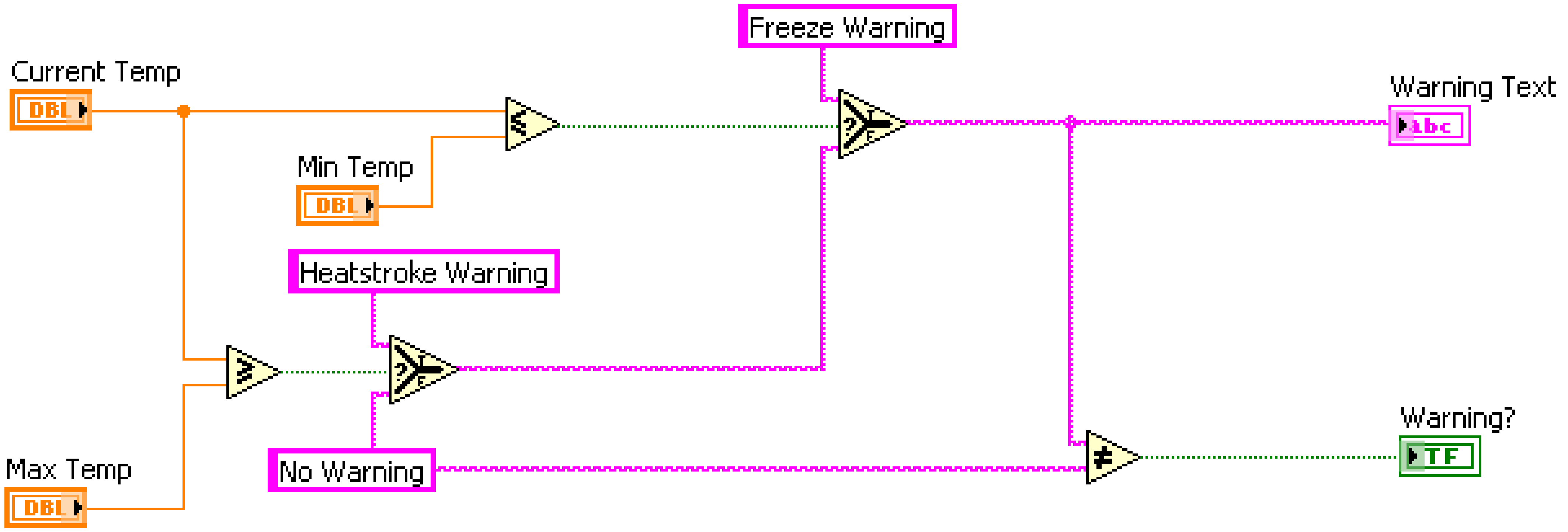
Design Patterns:

- Evolve through the efforts of multiple developers
- Fine-tuned for simplicity, maintainability, and readability

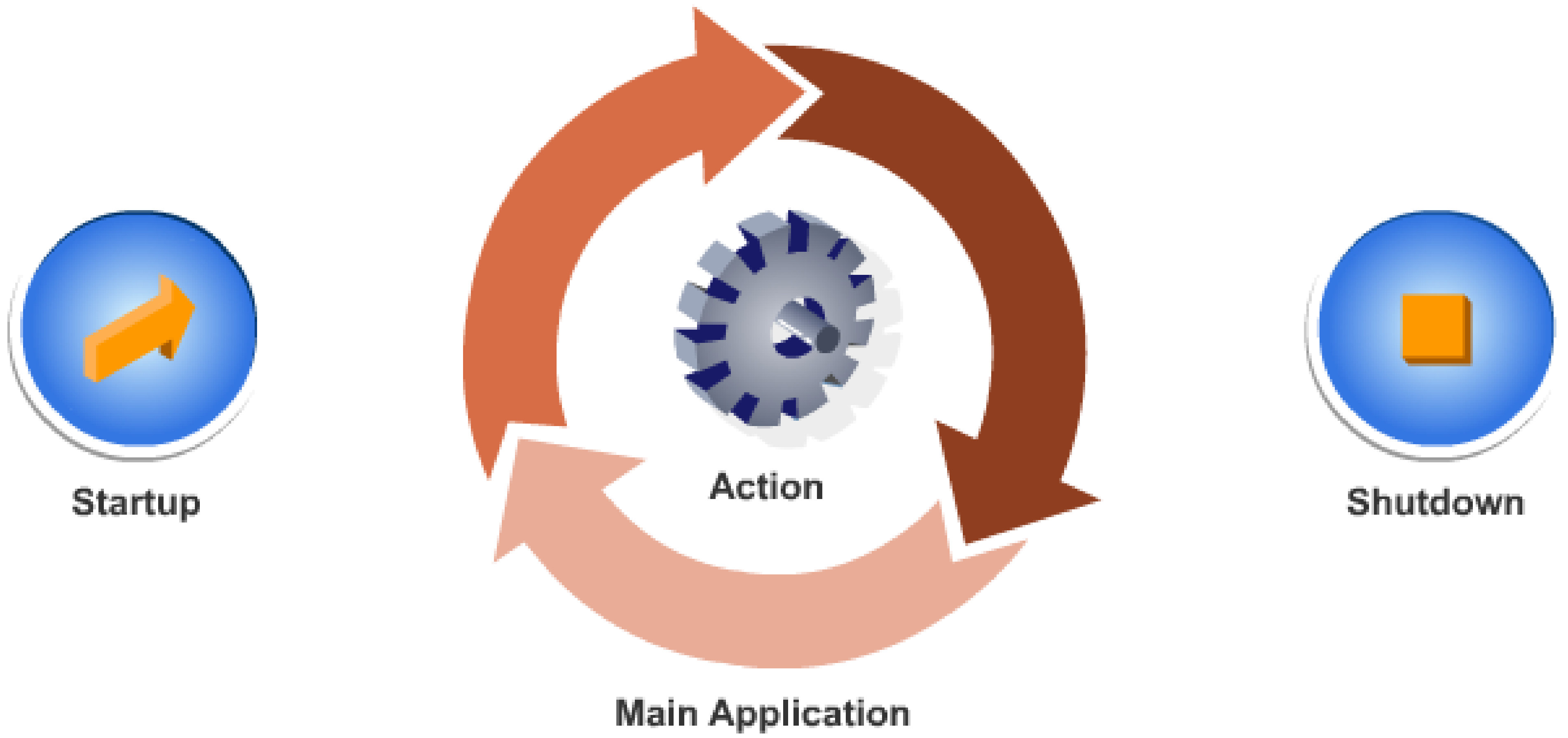
B. Simple Design Patterns

- Describe and use simple design patterns.
  - Simple VI
  - General VI
  - State Machine
  - Event-Based State Machine

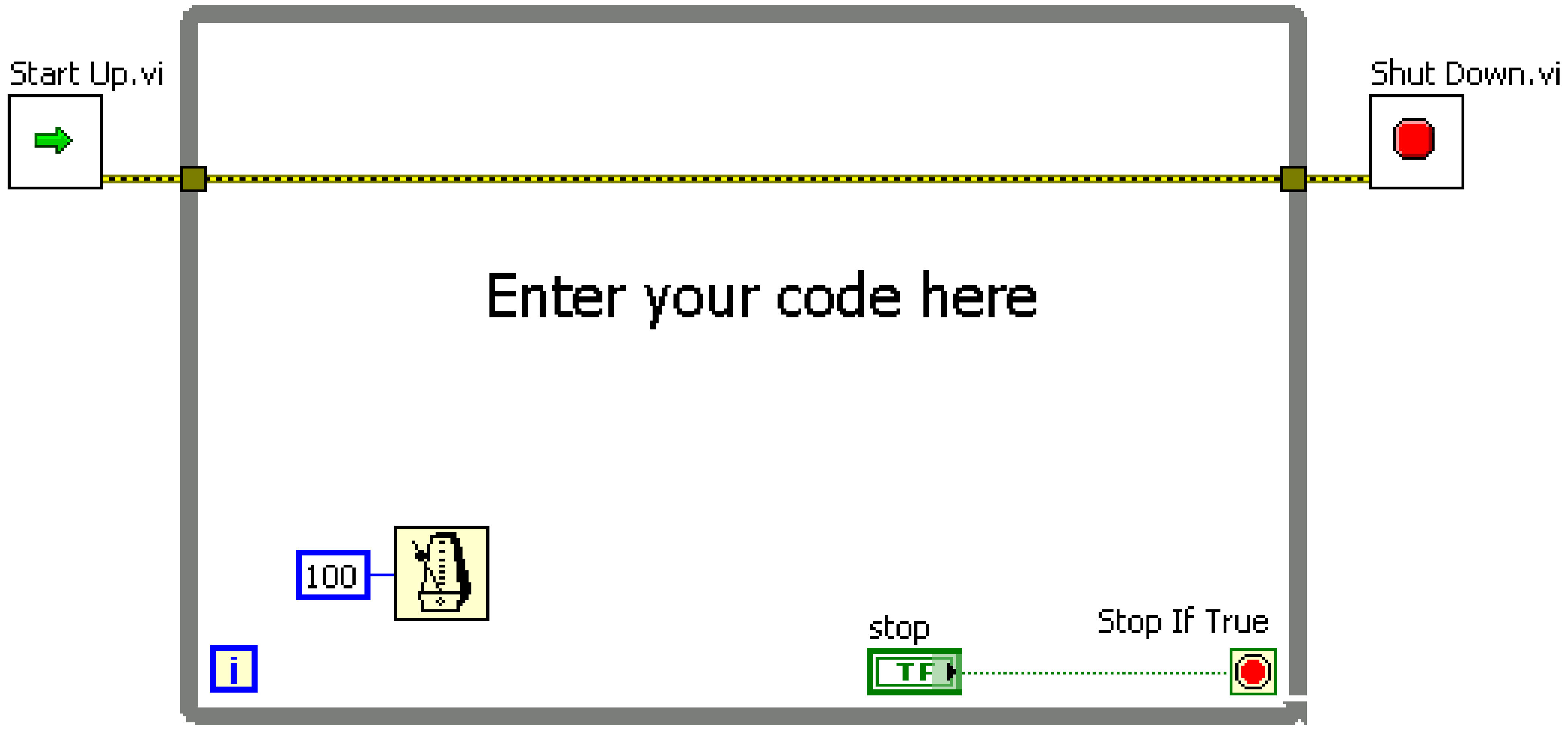
Simple VI Pattern



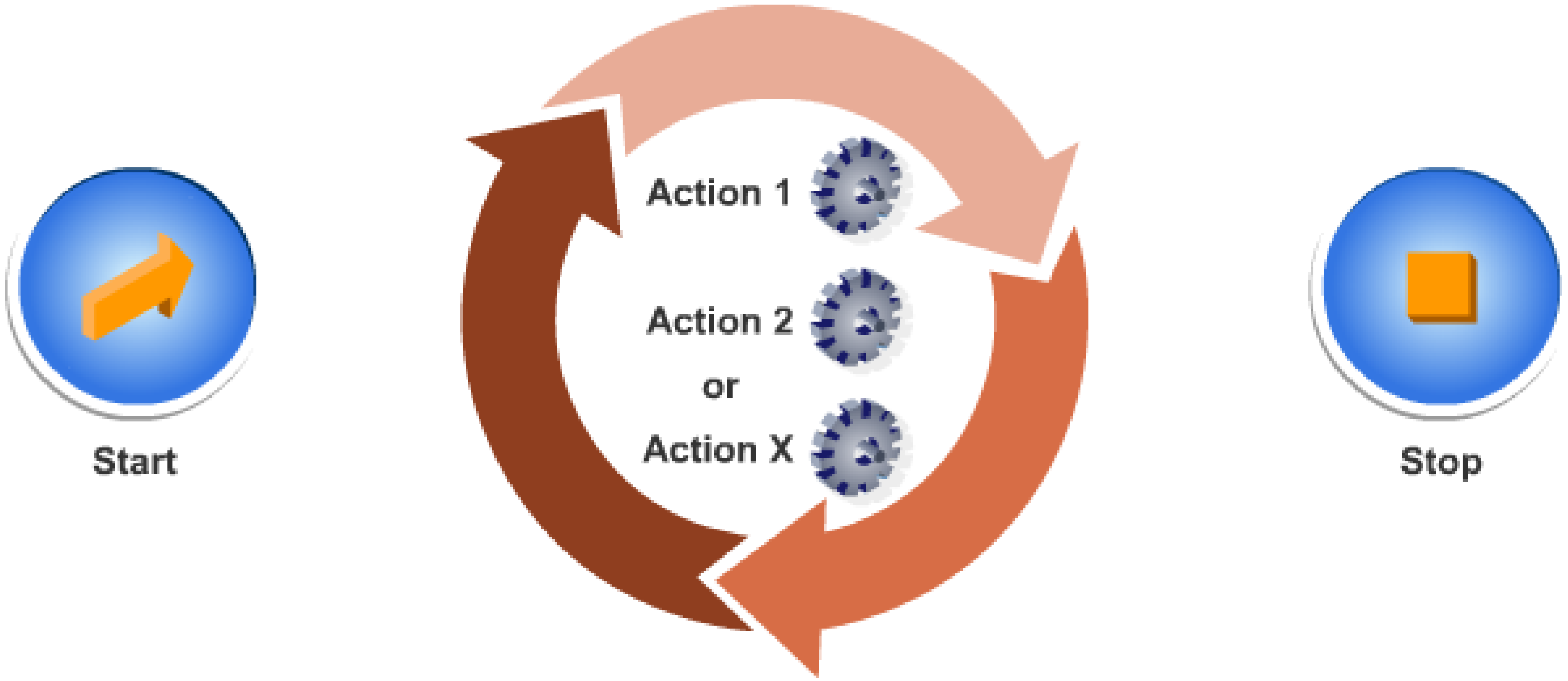
General VI Pattern



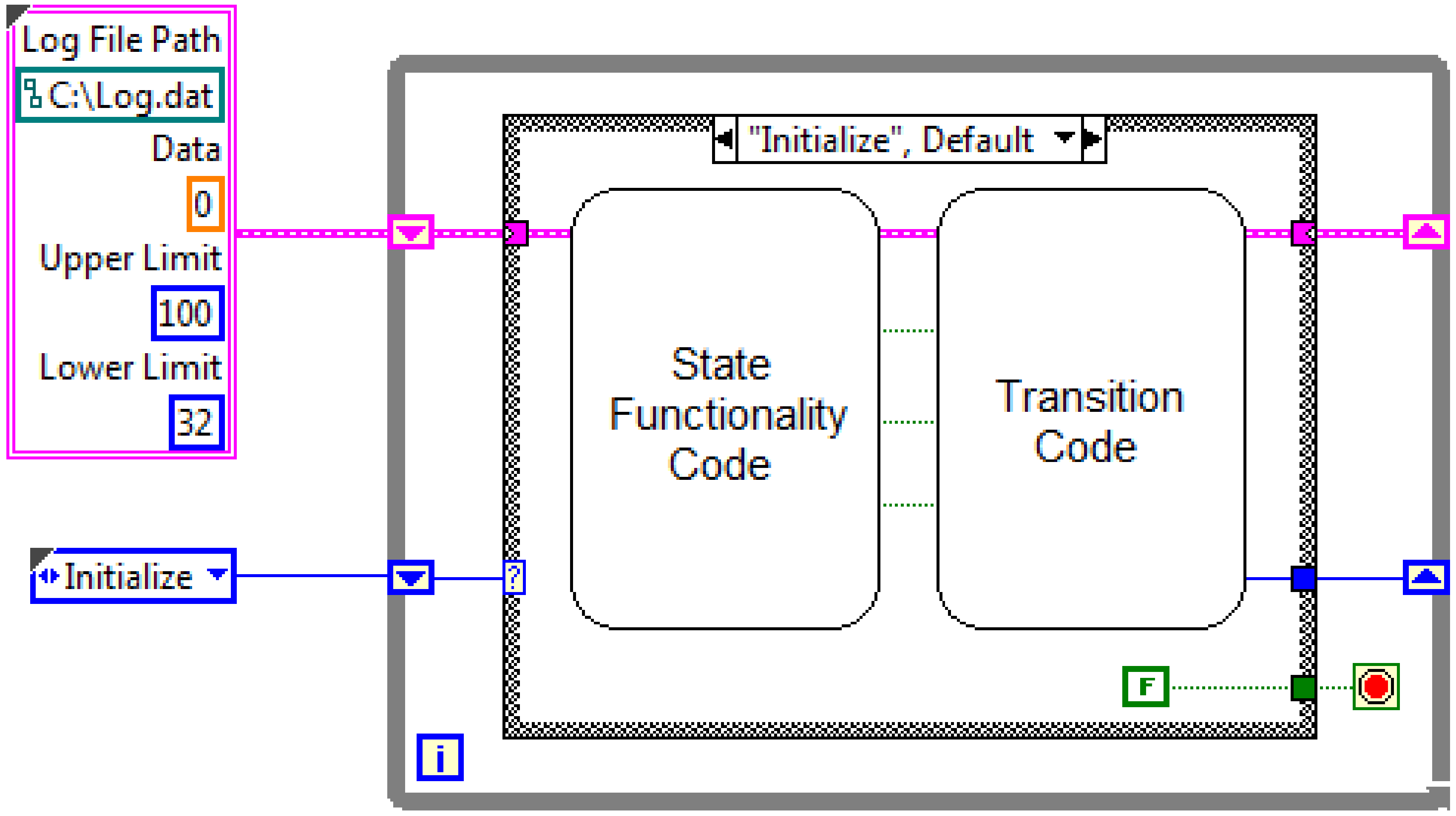
General VI Framework



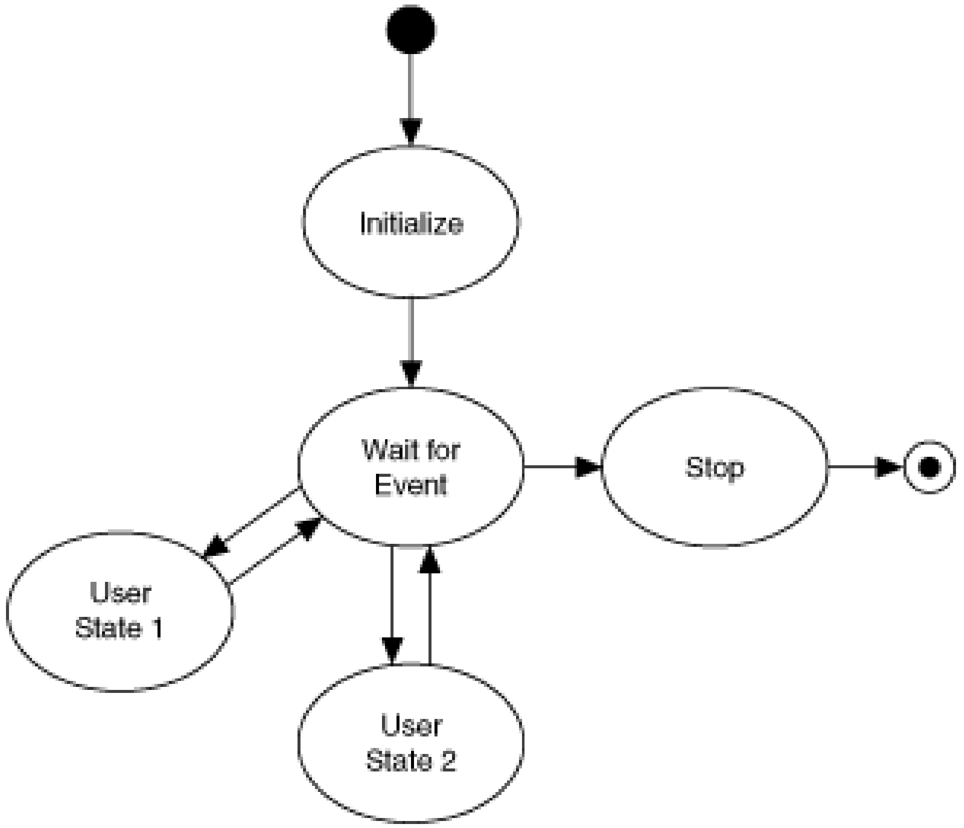
State Machine Pattern

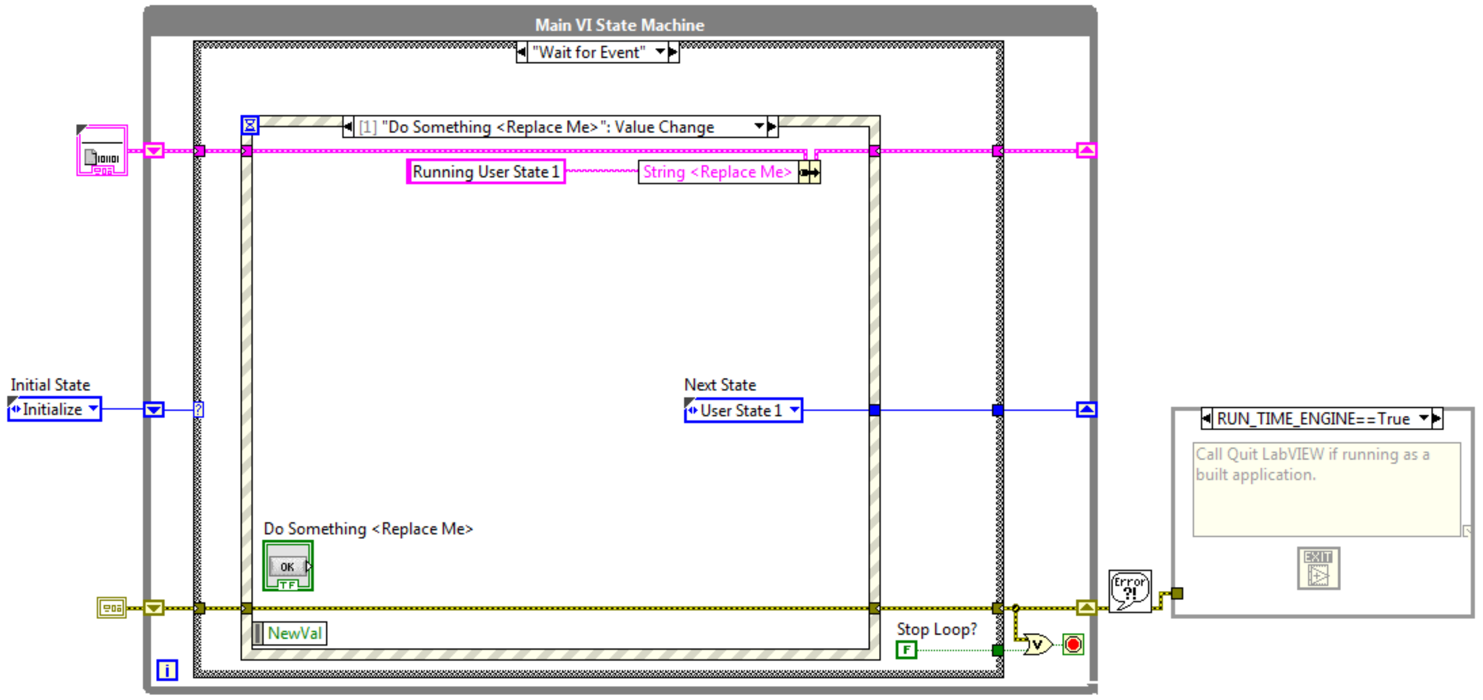


State Machine Framework



Event-Based State Machine





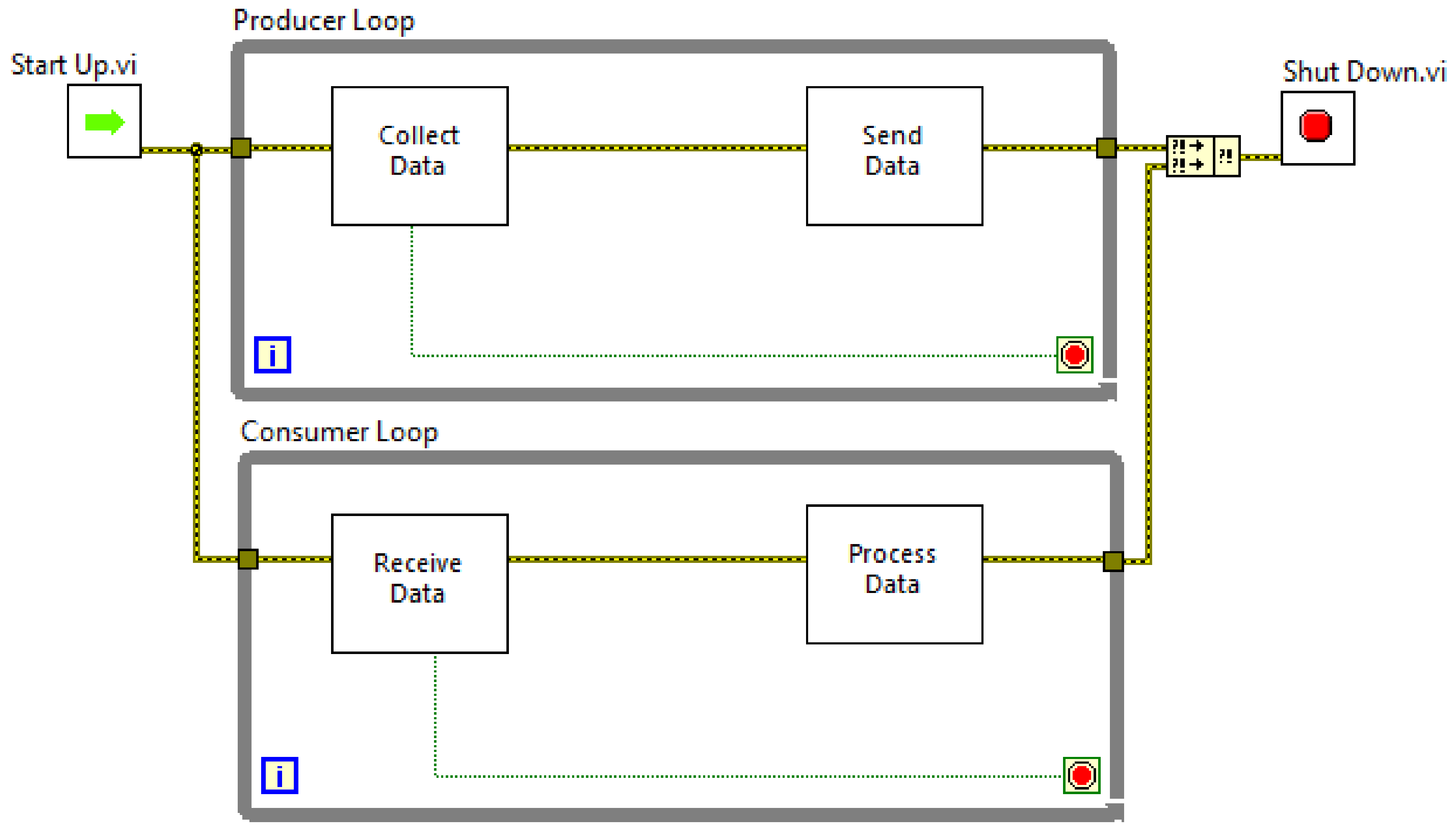
Redefine IT with Cloud

Migration • DevOps • Data Solutions • Serverless • Container

C. Multiple Loop Design Patterns

- Use the producer/consumer (events) design pattern to create multiple loop applications.

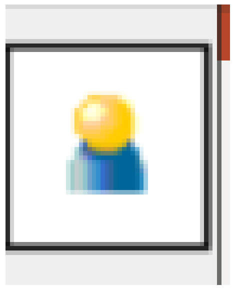
Producer/Consumer Design Pattern

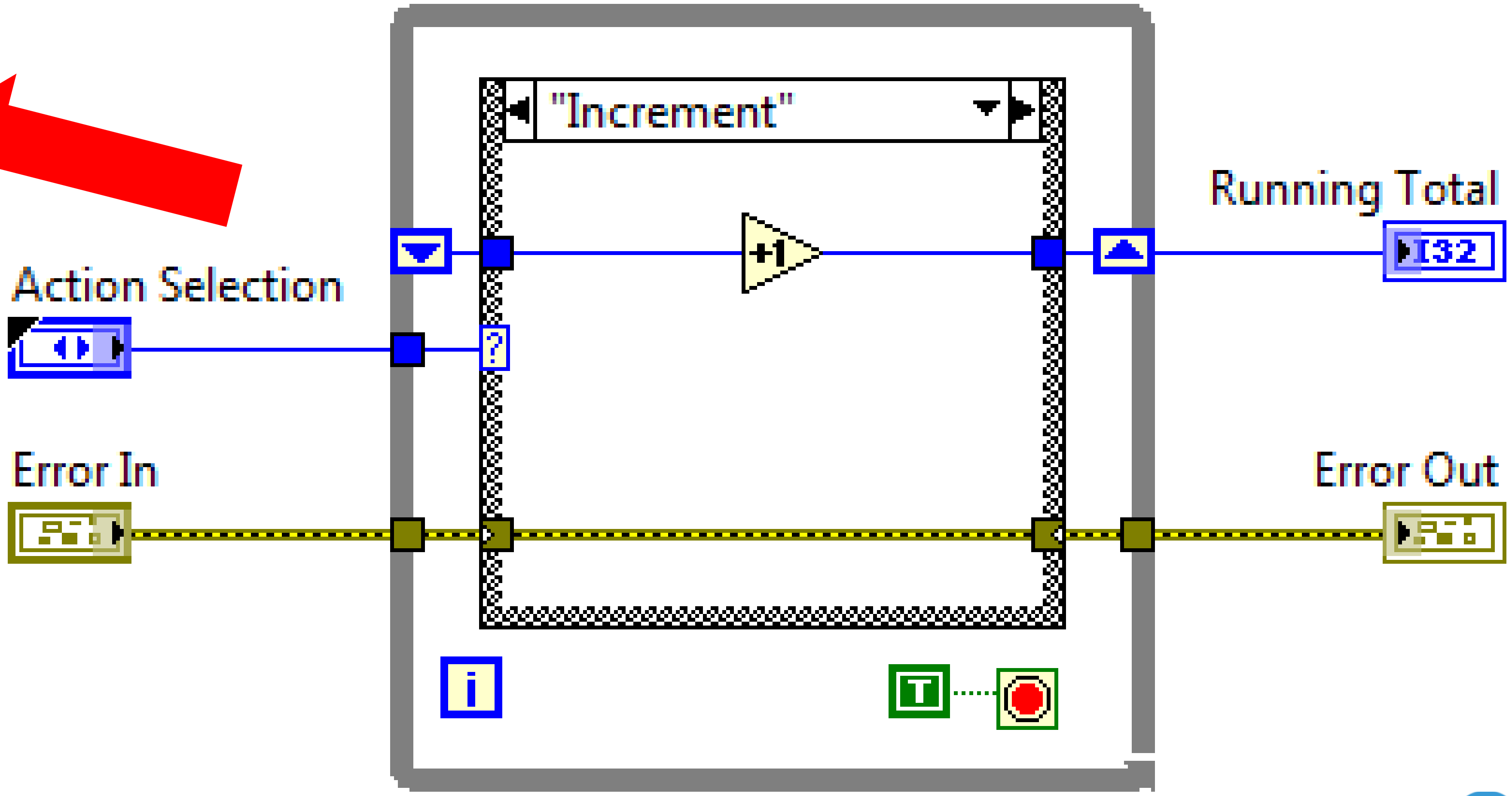


D. Functional Global Variable Design Pattern

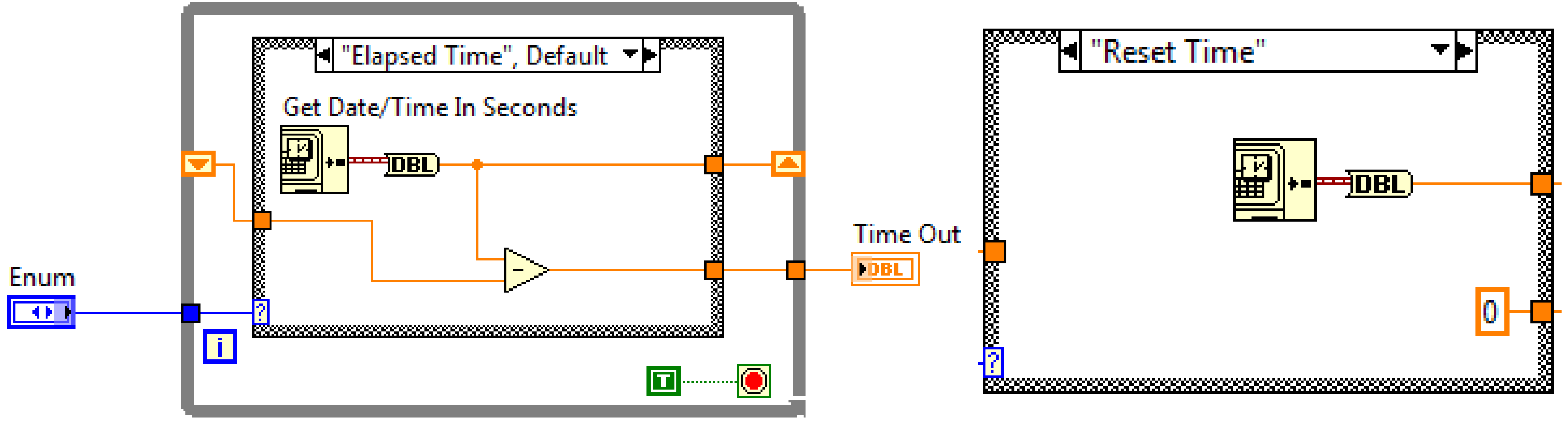
- Describe and use the functional global variable design pattern.
  - Functional Global Variables
  - Timer FGVs

Functional Global Variable Design Pattern





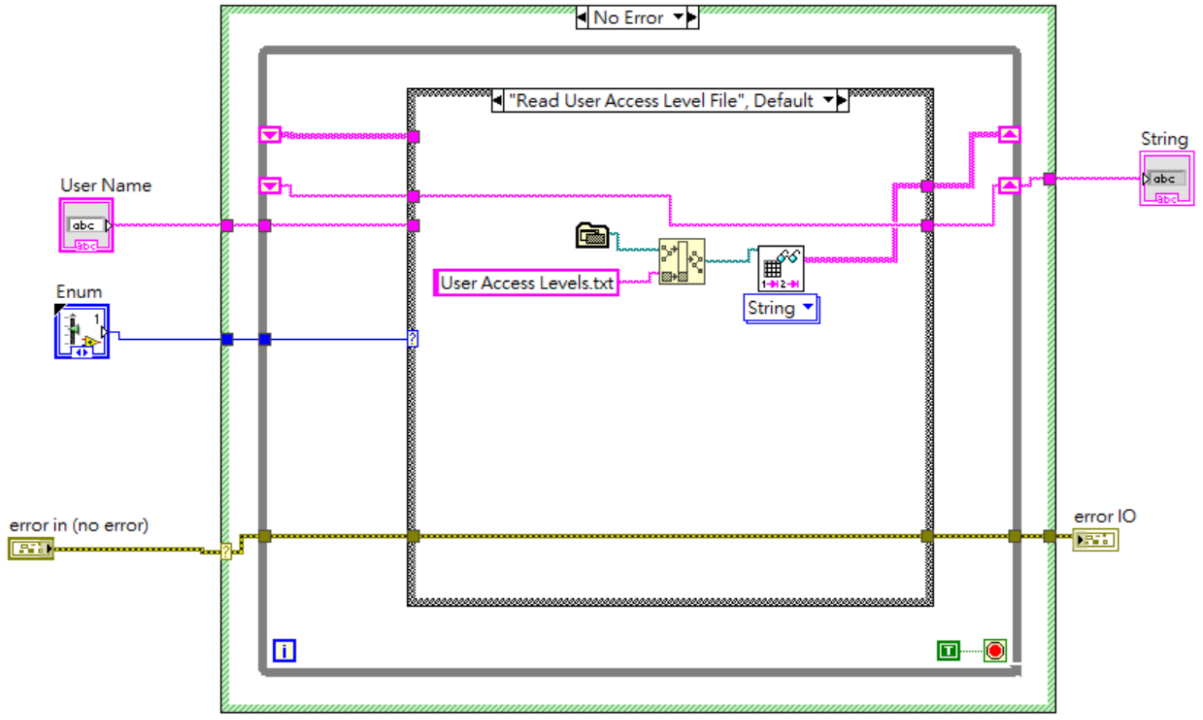
Timer FGVs

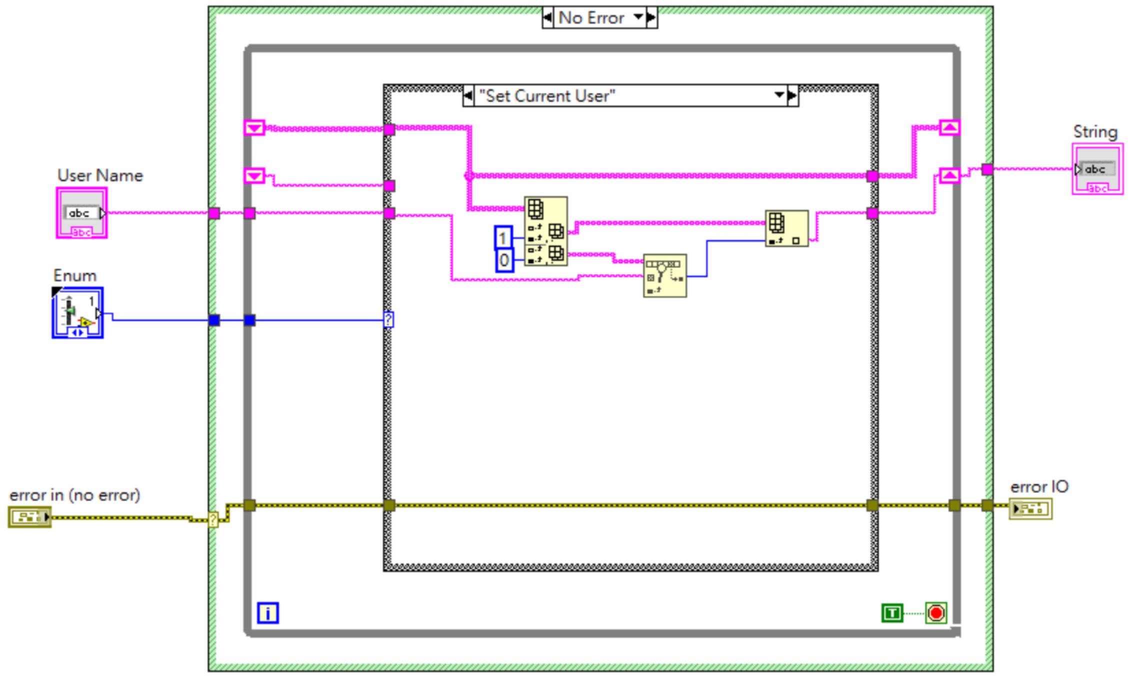

"Elapsed Time", Default
Get Date/Time In Seconds
DBL
Enum
Time Out
DBL
"Reset Time"
DBL
0

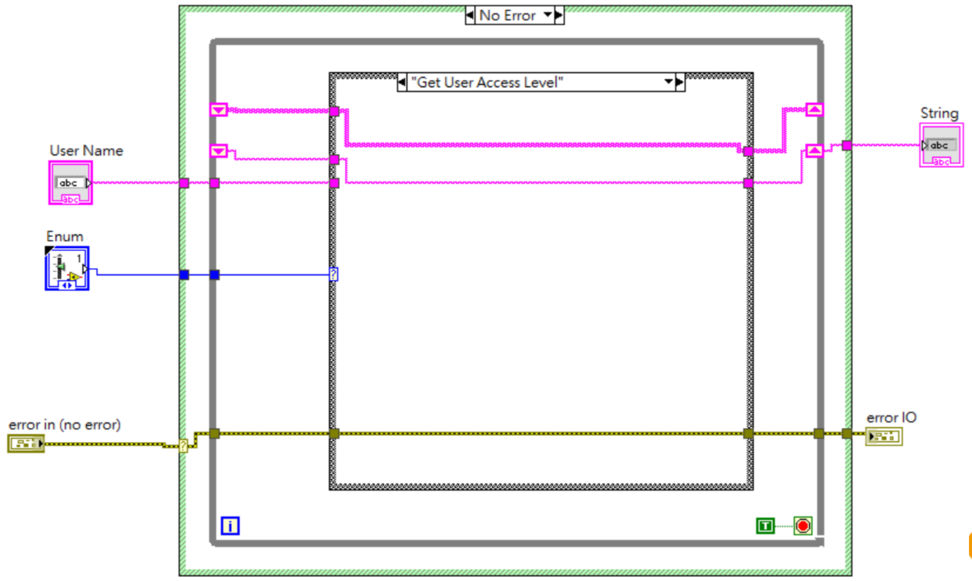
Exercise
Create a User Access Level FGV

- 建立一個Case, While loop, Case的三層架構, While設定為執行1次
- 建立Error Cluster連接到最外層Case, 並穿過整個架構
- 建立Enum存為Typedef, 包含Read User Access Level File, Set Current User, Get User Access Level
- Read Case設定讀取User Access Levels.txt
- Set Case設定讀取User Level
- Get Case 直接穿過
- 完成FGV, 放入User Access Level FGV Unit Test.vi測試，根據不同Event改變Enum Value

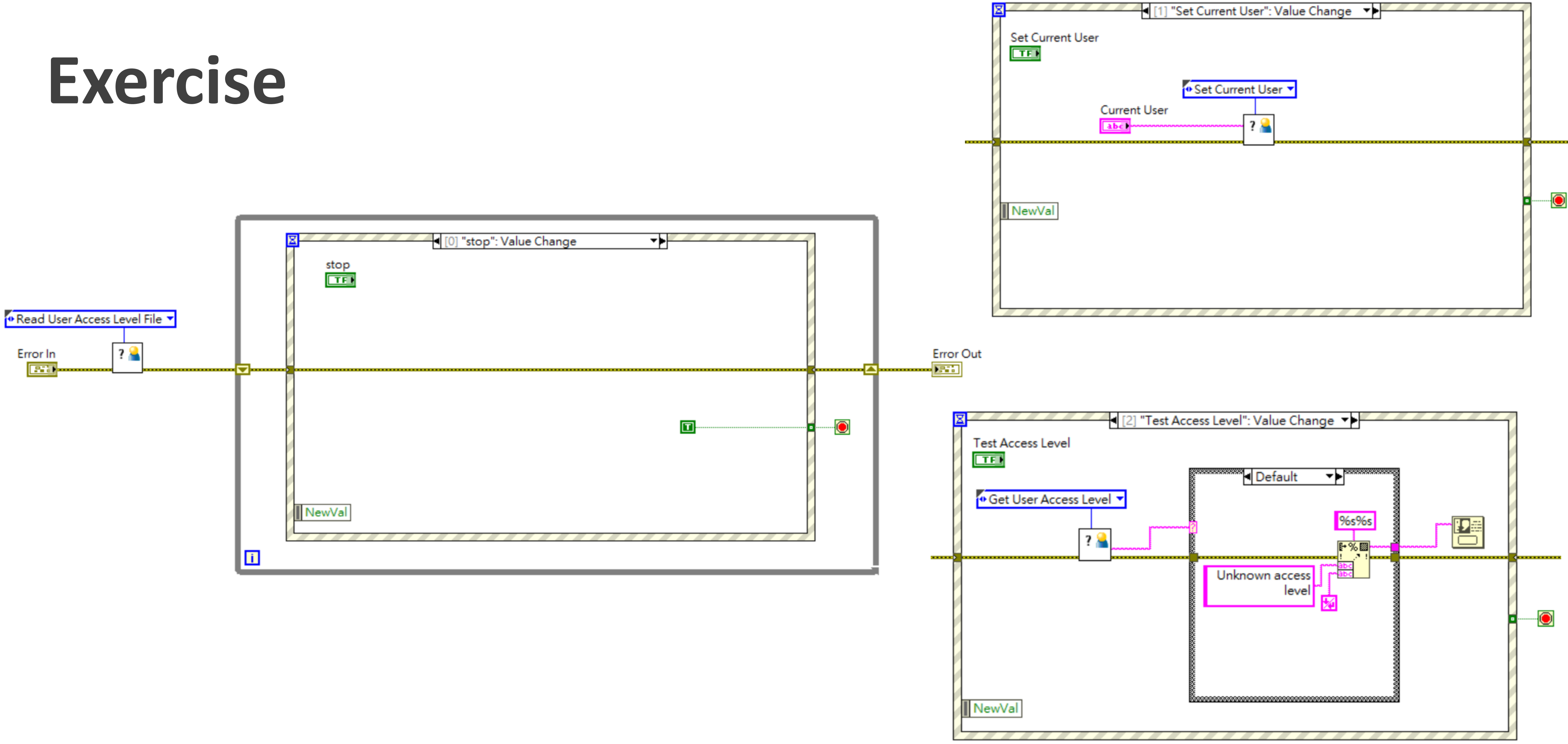
Exercise


No Error
"Read User Access Level File", Default
User Name
Enum
String
User Access Levels.txt
String
error in (no error)
error IO


No Error
"Set Current User"
User Name
Enum
String
error in (no error)
error IO


No Error
"Get User Access Level"
User Name
Enum
String
error in (no error)
error IO

Exercise



Read User Access Level File
Error In
[0] "stop": Value Change
stop
NewVal
Error Out
[1] "Set Current User": Value Change
Set Current User
Set Current User
Current User
NewVal
[2] "Test Access Level": Value Change
Test Access Level
Get User Access Level
Default
%s%s
Unknown access level
NewVal

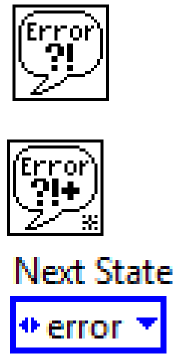
E. Error Handlers

- Use error handlers in design patterns to manage code execution when an error occurs.
  - Examples of Error Handlers

Examples of Error Handlers

Error Handler—A VI or code that changes normal flow of program execution when an error occurs

- Simple Error Handler VI
- General Error Handler VI
- State machine error handler



- **1. Which of the following are reasons for using a multiple loop design pattern?**

a. Execute multiple tasks concurrently
b. Execute different states in a state machine
c. Execute tasks at different rates
d. Execute start up code, main loop, and shutdown code

- **1. Which of the following are reasons for using a multiple loop design pattern?**

a. **Execute multiple tasks concurrently**
b. Execute different states in a state machine
c. **Execute tasks at different rates**
d. Execute start up code, main loop, and shutdown code

Source Code Control

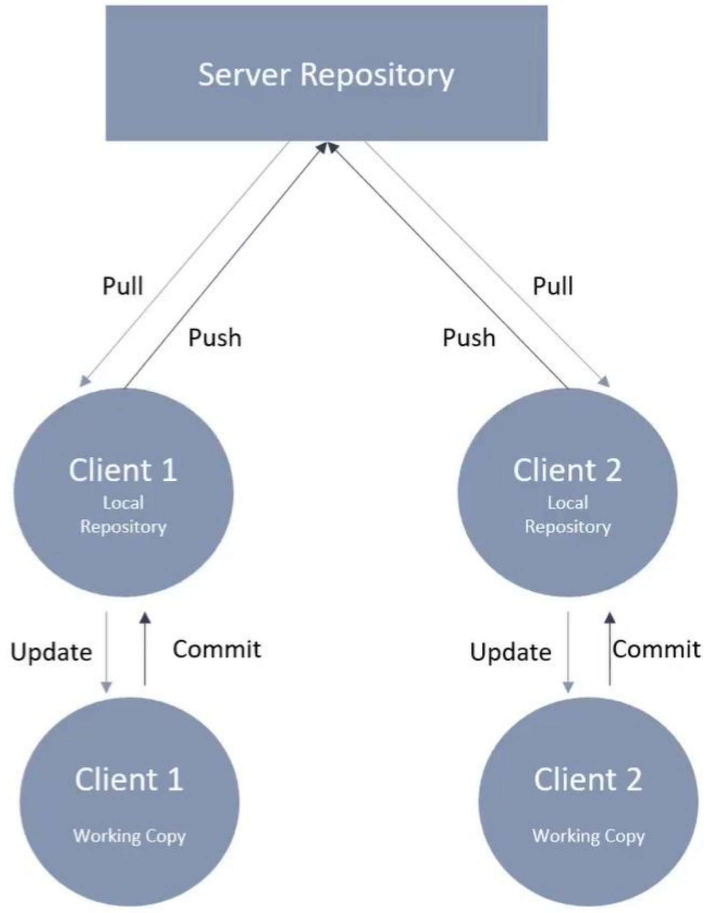
- Software that tracks changes to files
  - Stores all versions of files and their change records
  - Provides multiple developers access to files

Distributed Version Control

Github

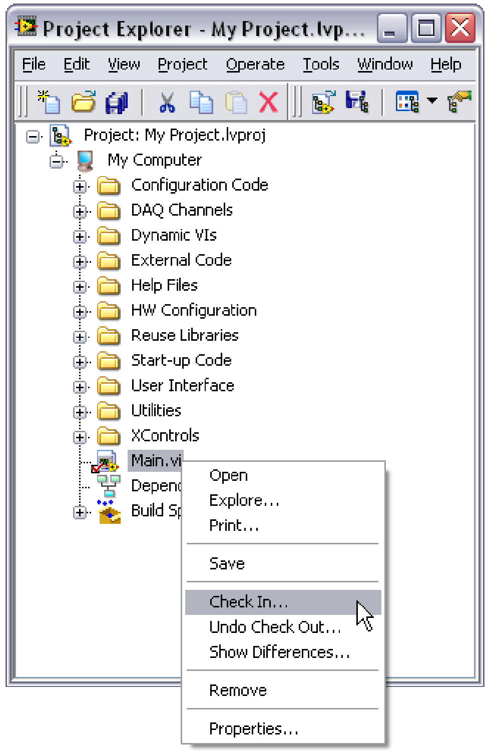
Tortoise
Perforce

PC
(LabVIEW)

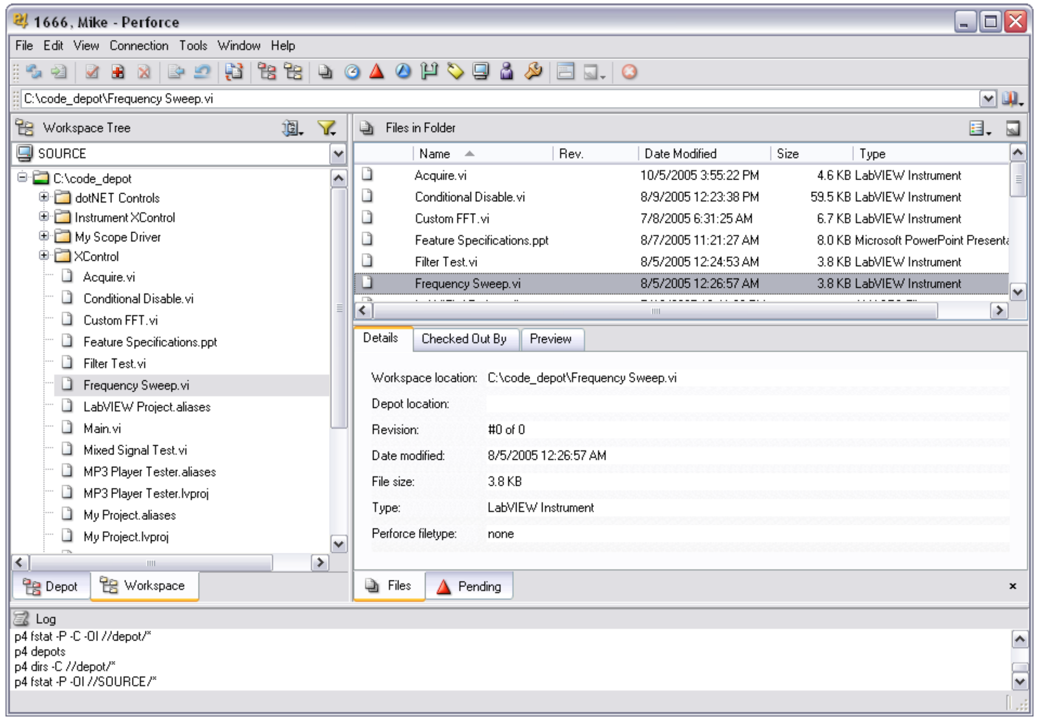


Source Code Control – Options

- Use within LabVIEW Professional Development System



- Use directly through a source code control tool



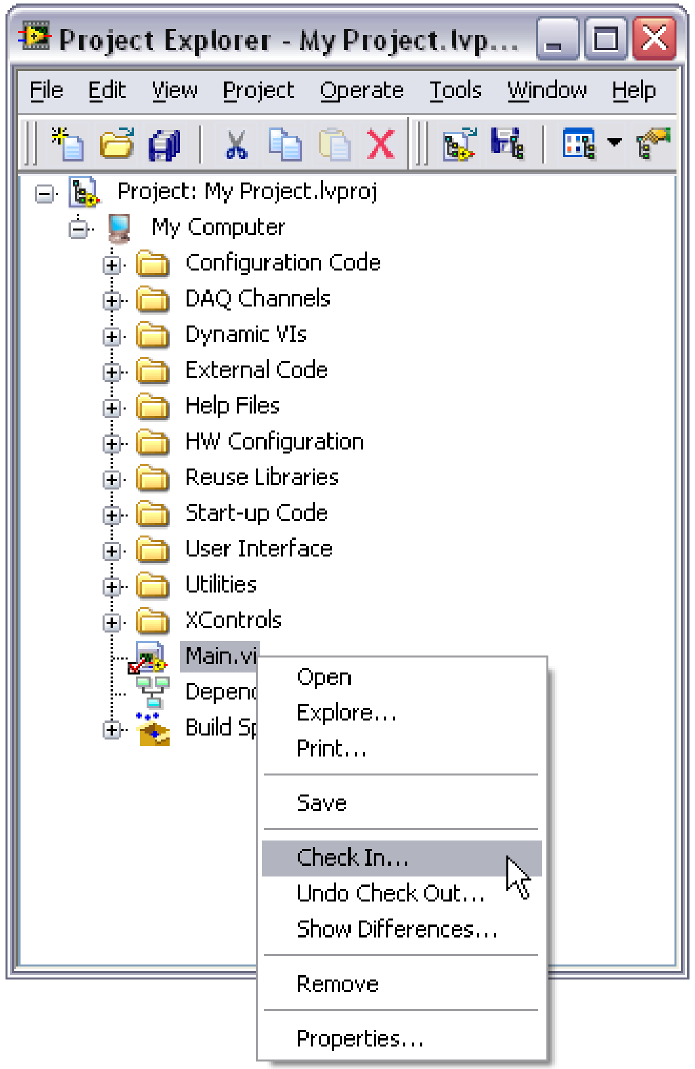
Name	Rev.	Date Modified	Size	Type
Acquire.vi		10/5/2005 3:55:22 PM	4.6 KB	LabVIEW Instrument
Conditional Disable.vi		8/9/2005 12:23:38 PM	59.5 KB	LabVIEW Instrument
Custom FFT.vi		7/8/2005 6:31:25 AM	6.7 KB	LabVIEW Instrument
Feature Specifications.ppt		8/7/2005 11:21:27 AM	8.0 KB	Microsoft PowerPoint Presenta
Filter Test.vi		8/5/2005 12:24:53 AM	3.8 KB	LabVIEW Instrument
Frequency Sweep.vi		8/5/2005 12:26:57 AM	3.8 KB	LabVIEW Instrument

Workspace location:	C:\code_depot\Frequency Sweep.vi
Depot location:	
Revision:	#0 of 0
Date modified:	8/5/2005 12:26:57 AM
File size:	3.8 KB
Type:	LabVIEW Instrument
Perforce filetype:	none

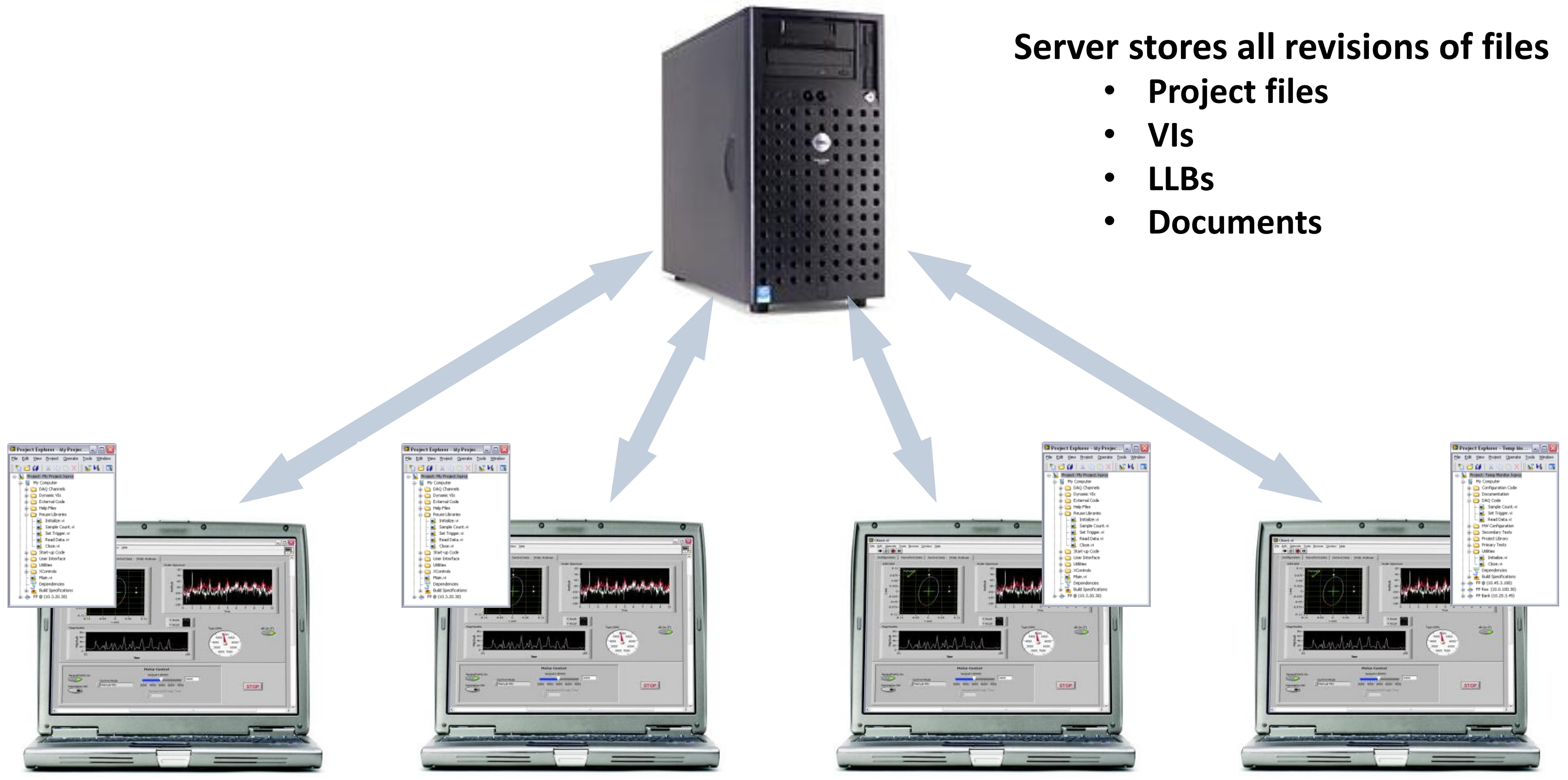
```
p4 fstat -P -C -Ol //depot/*
p4 depots
p4 dirs -C //depot/*
p4 fstat -P -Ol //SOURCE/*
```

Source Code Control – Integration with the LabVIEW Project

- LabVIEW Project makes accessing SCC in LabVIEW simpler
  - Right-click one or more files to check in or out
  - Right-click and select Show Differences to view edits interactively (Perforce and VSS)
- File icon shows current status
  - Checked in
  - Checked out



Source Code Control – Team-Based Development



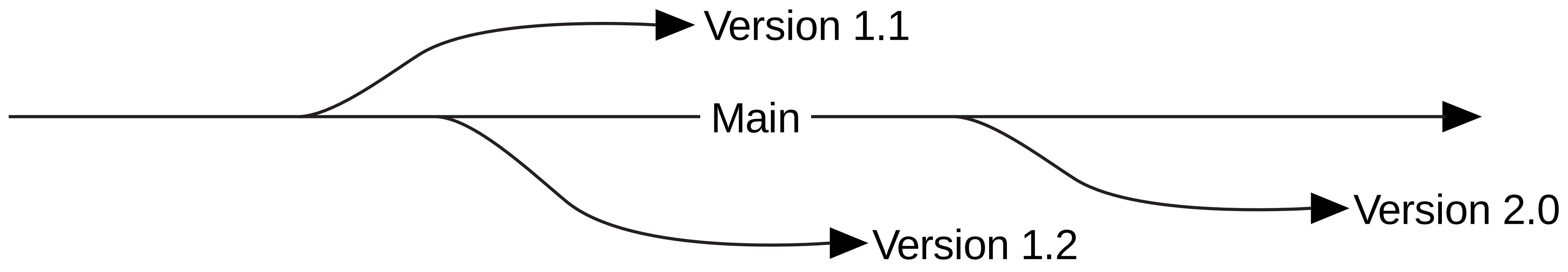
Server stores all revisions of files

- Project files
- VIs
- LLBs
- Documents

Each developer can check copies of files in and out as needed

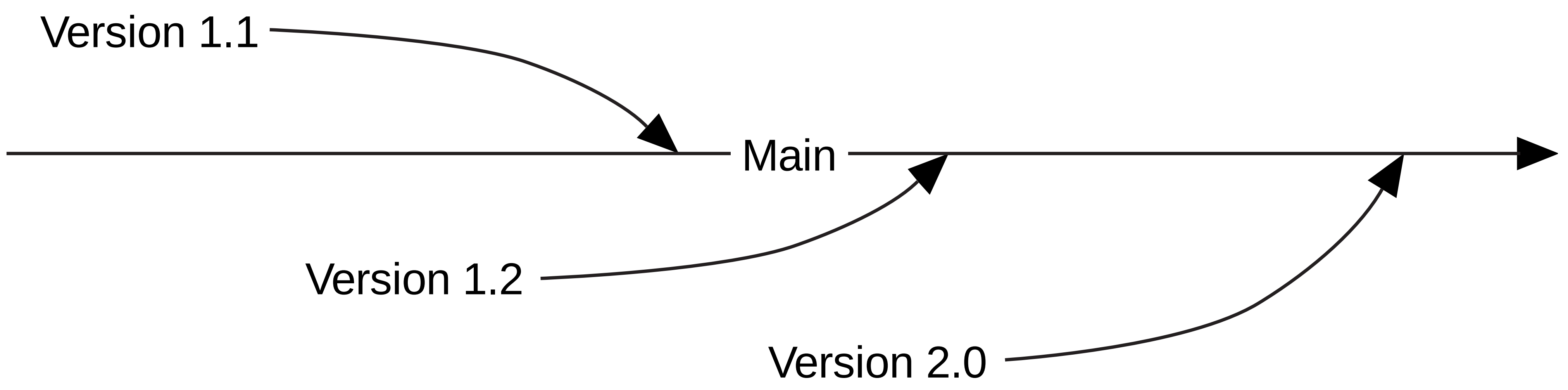
Source Code Control – Branch and Merge

Branch—Split from the main development line to create a new version of the code



Source Code Control – Branch and Merge (continued)

Merge—Integrate the development split into the main development line



Demonstration
Source Code Control with Tortoise and Github

- Install and configure a source code control system with LabVIEW to improve the configuration management of a project and learn common techniques of using a source code control system.

Thank you for kind attention~

Joe Kao
Solution Team
Industrial Intelligence BU
E: joe.kao@ecloudvalley.com