LabVIEW Lesson Day1

Joe Kao
Application engineer, eCloudValley

2022/1/24

Speaker Introduction

- Joe Kao, Application Engineer, eCloudvalley
- National Taiwan University, Institute of Applied Mechanics
- Expertise: Advanced Fluid Mechanics, Advanced Engineering Mathematics, LabVIEW

Using Decision-Making Structures

- Learn to create different decision-making structures and be able to identify applications where using these structures can be beneficial.

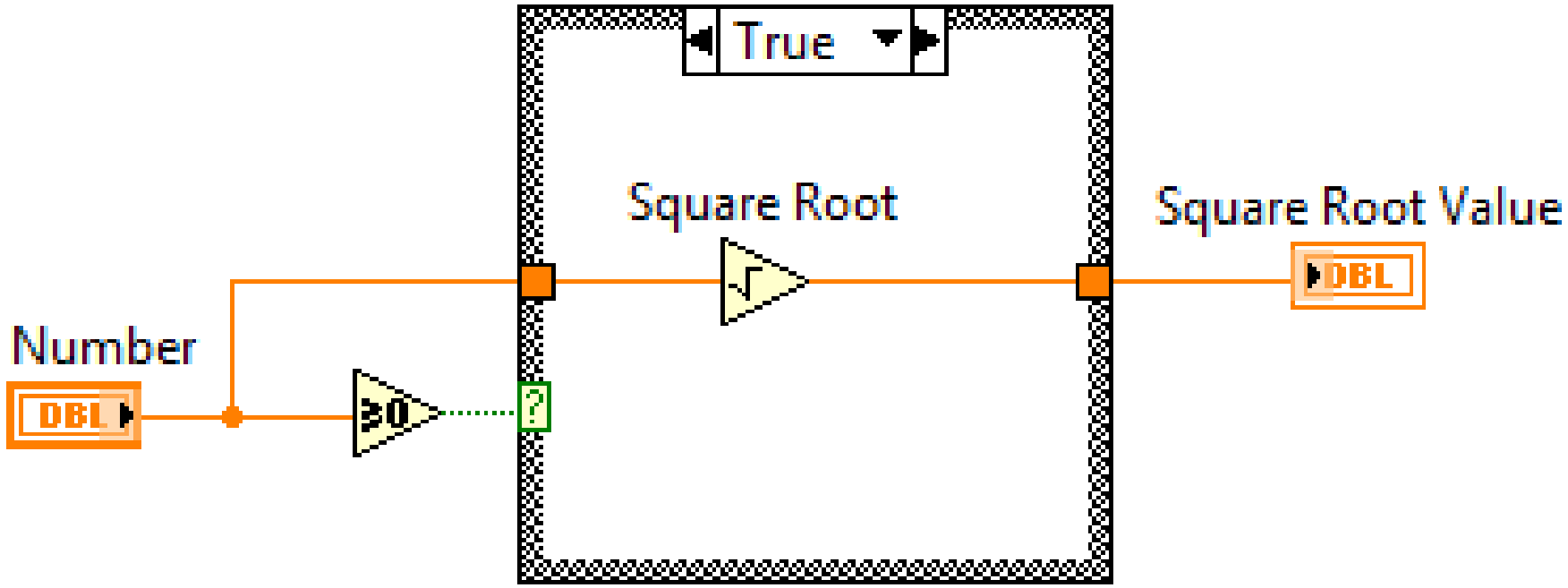
- Case Structures
- Event-Driven Programming

A. Case Structures

- Recognize and use the basic features and functionality of Case Structures.

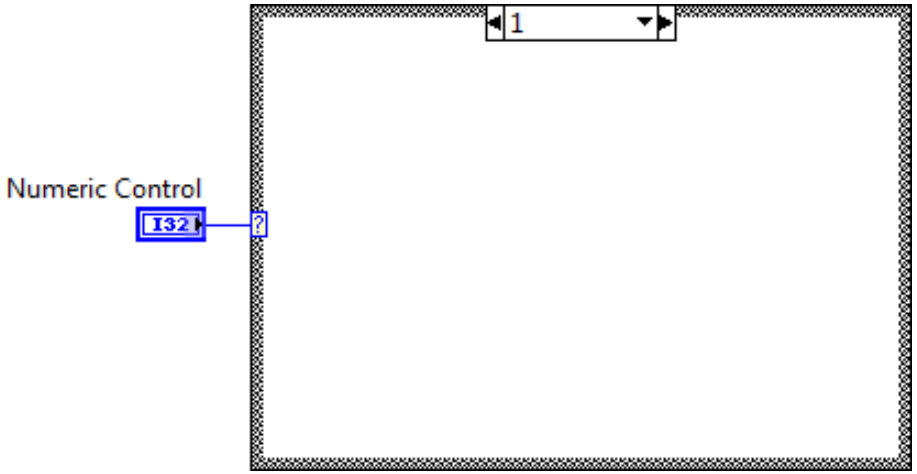
- Case Structures Review
- Selector Terminal Data Types
- Input and Output Tunnels

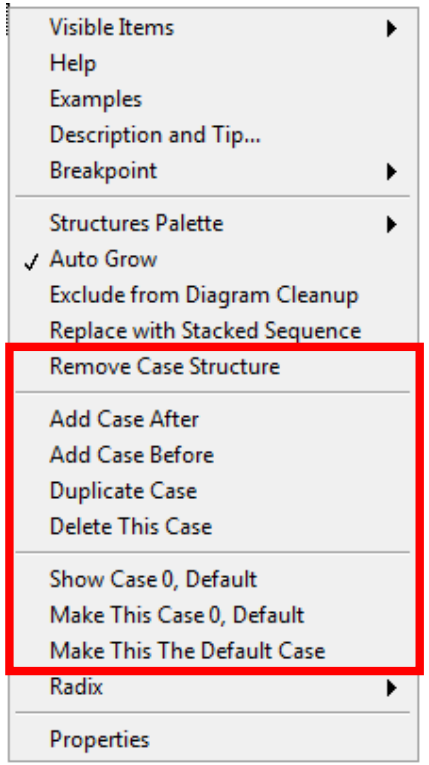
Case Structures Review



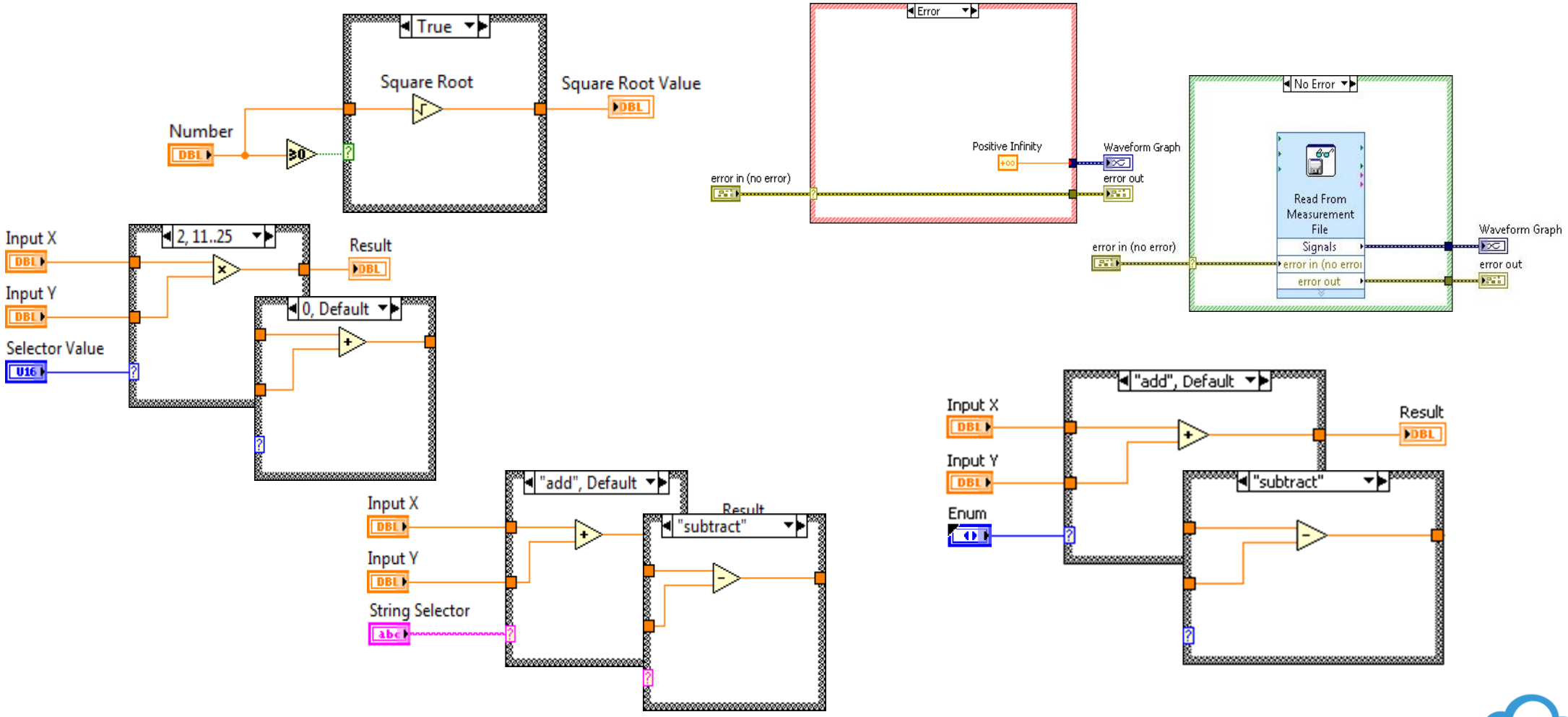
Demonstration

- Create and configure case structures.



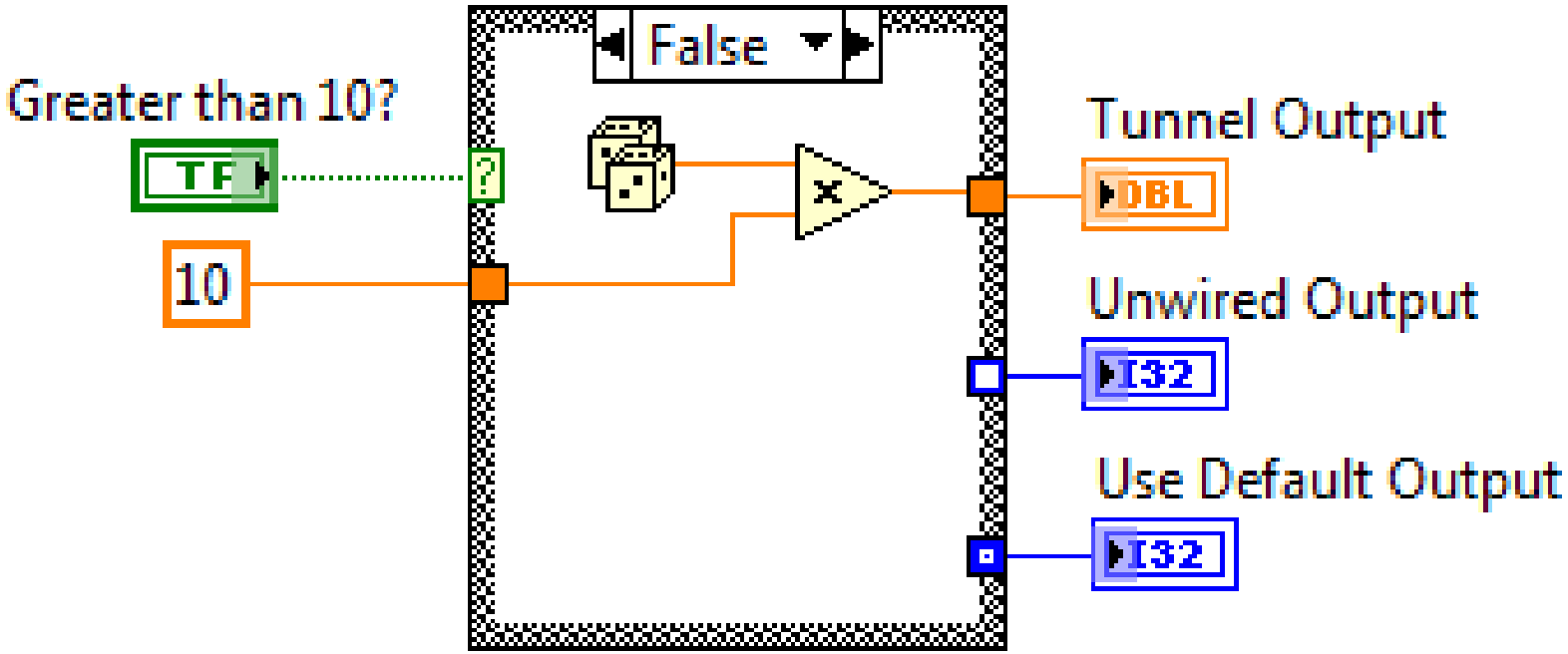


Selector Terminal Data Types



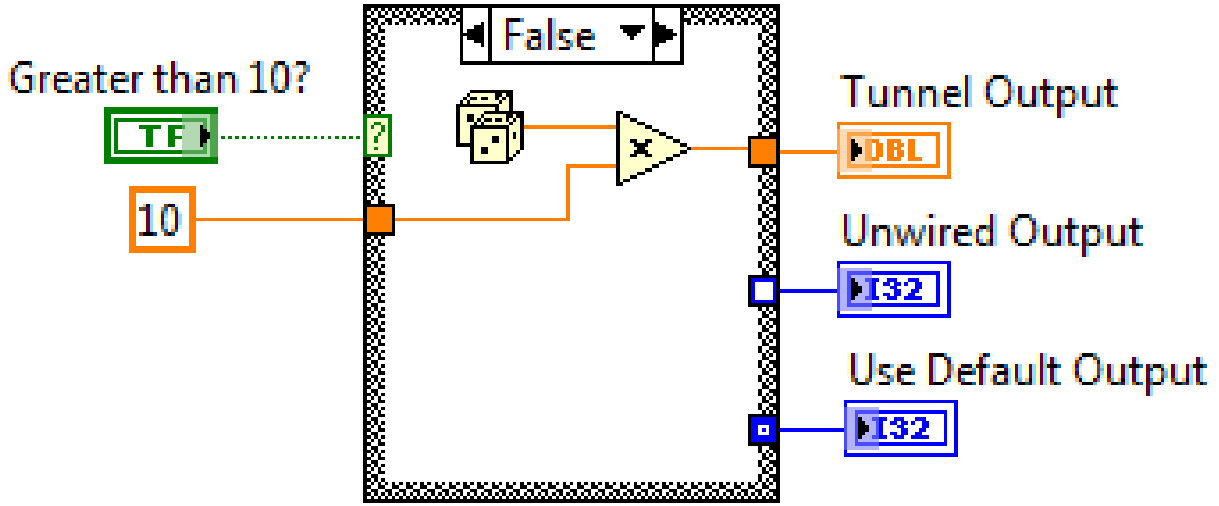
Input and Output Tunnels

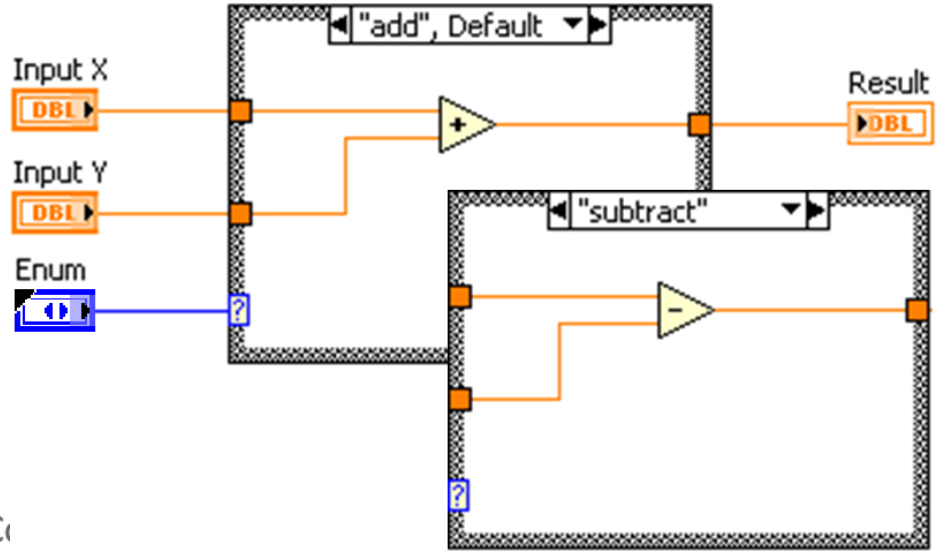
- Input tunnels are available to all cases if needed.
- You must define every output tunnel for each case.



Demonstration

- Create Case structures using different data type selectors.
- Create different types of output tunnels.

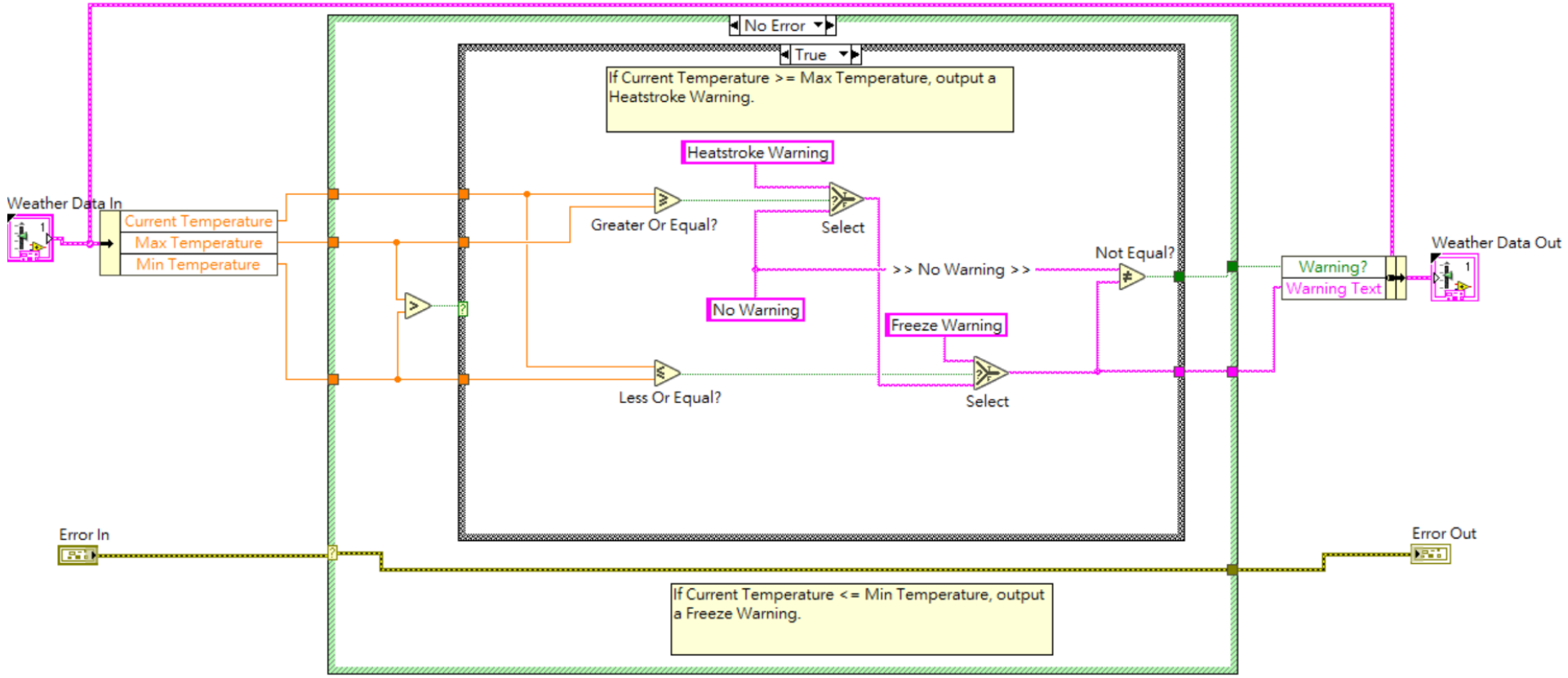




Exercise
Temperature Warnings with Error Handling

- 打開Weather Warning.lvproj
- 打開Temperature Warning.vi
- 打開Block Diagram建立Weather Data Cluster並Make Typedef後儲存
- Unbundle by name取出Cluster內參數
- 加上Case Structure(Error, Max Temp<Min Temp?)
- 完成各Case端點

Exercise

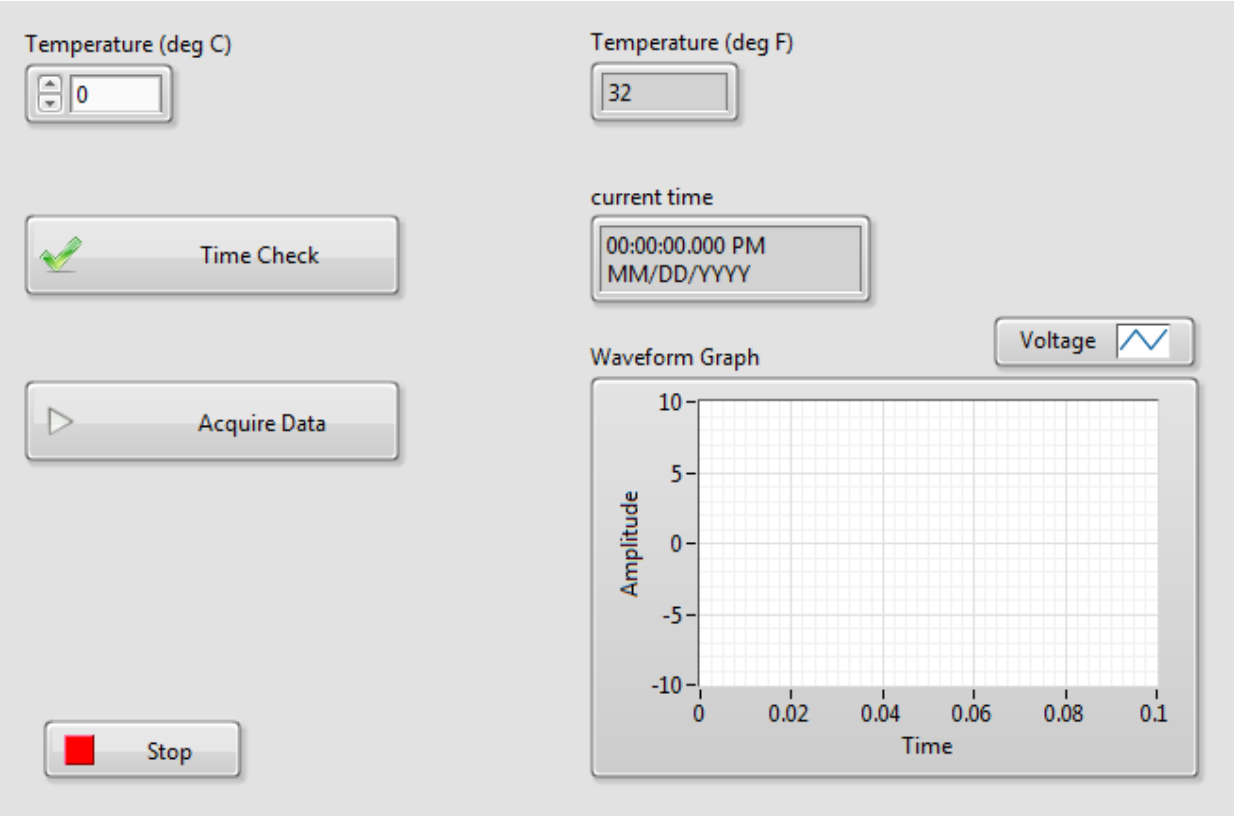

No Error
True
If Current Temperature >= Max Temperature, output a Heatstroke Warning.
Heatstroke Warning
Weather Data In
Current Temperature
Max Temperature
Min Temperature
Greater Or Equal?
Select
>> No Warning >>
Not Equal?
Weather Data Out
Warning?
Warning Text
No Warning
Freeze Warning
Less Or Equal?
Select
Error In
Error Out
If Current Temperature <= Min Temperature, output a Freeze Warning.

B. Event-Driven Programming

- Recognize basic features and functionality of event structures.

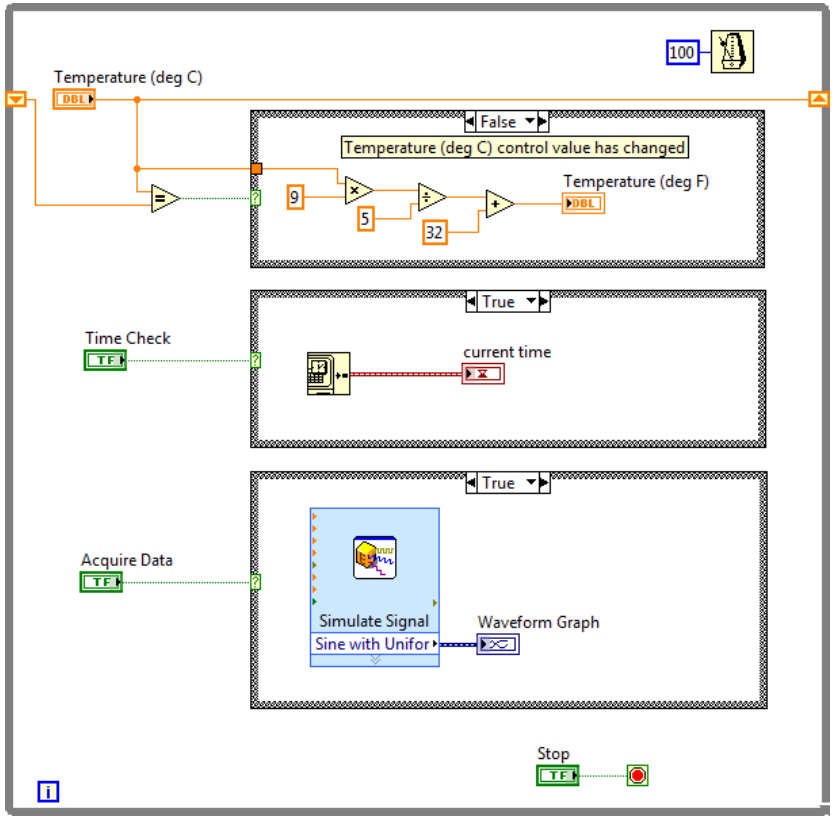
- Event-Driven Scenario
- Polling Versus Events
- Event-Driven Programming
- Configuring the Event Structure
- Caveats and Recommendations

Demonstration
Event-Driven Scenario

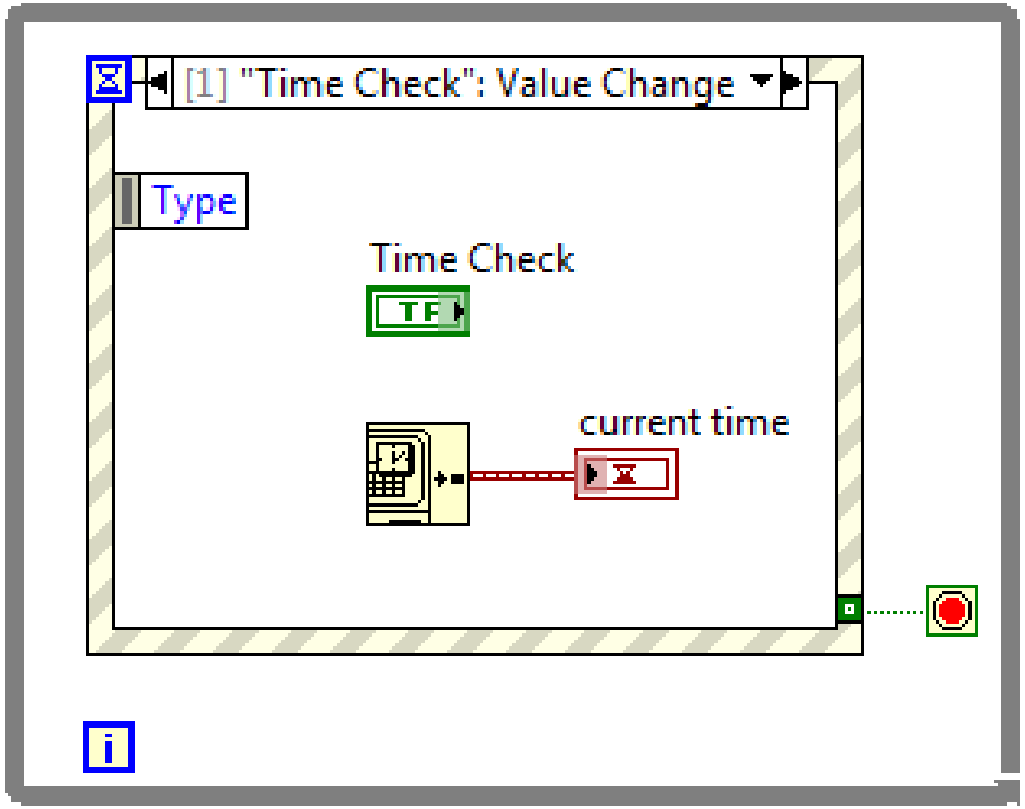


Polling Versus Events

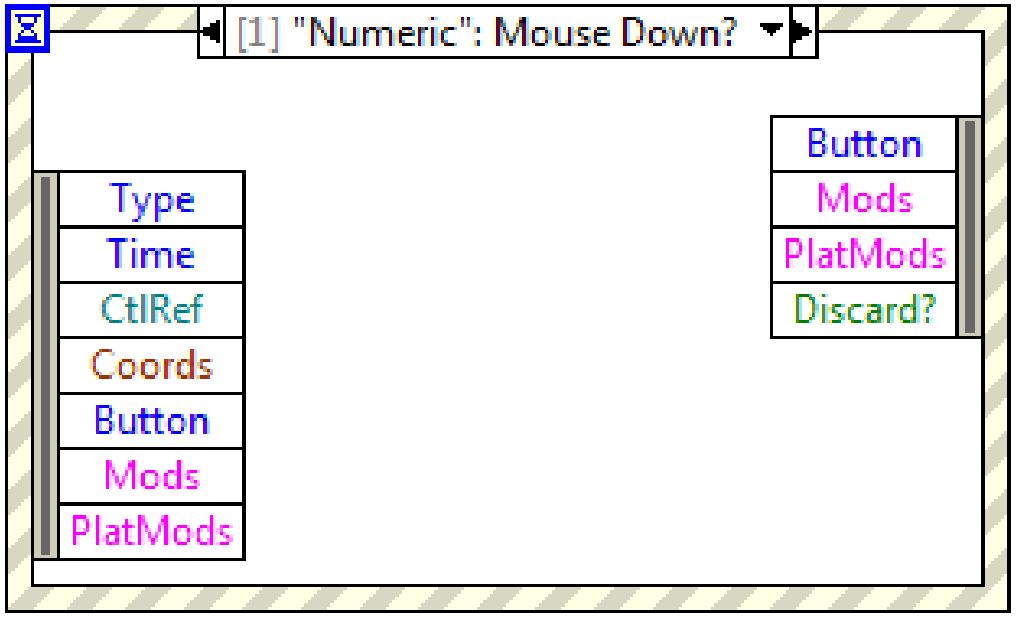
Polling

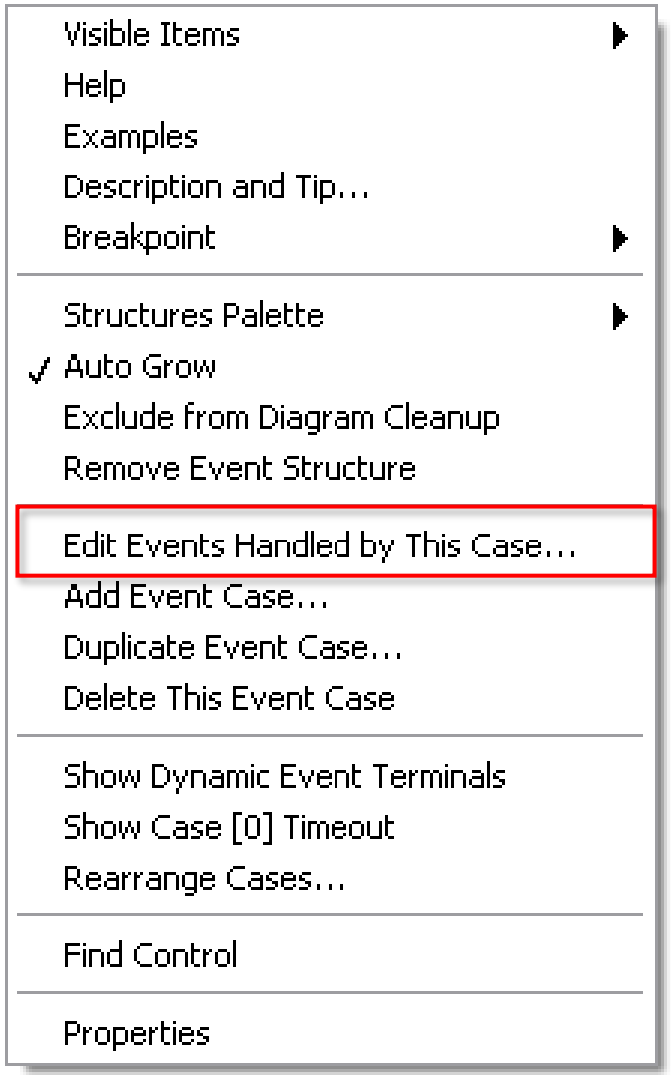


Events

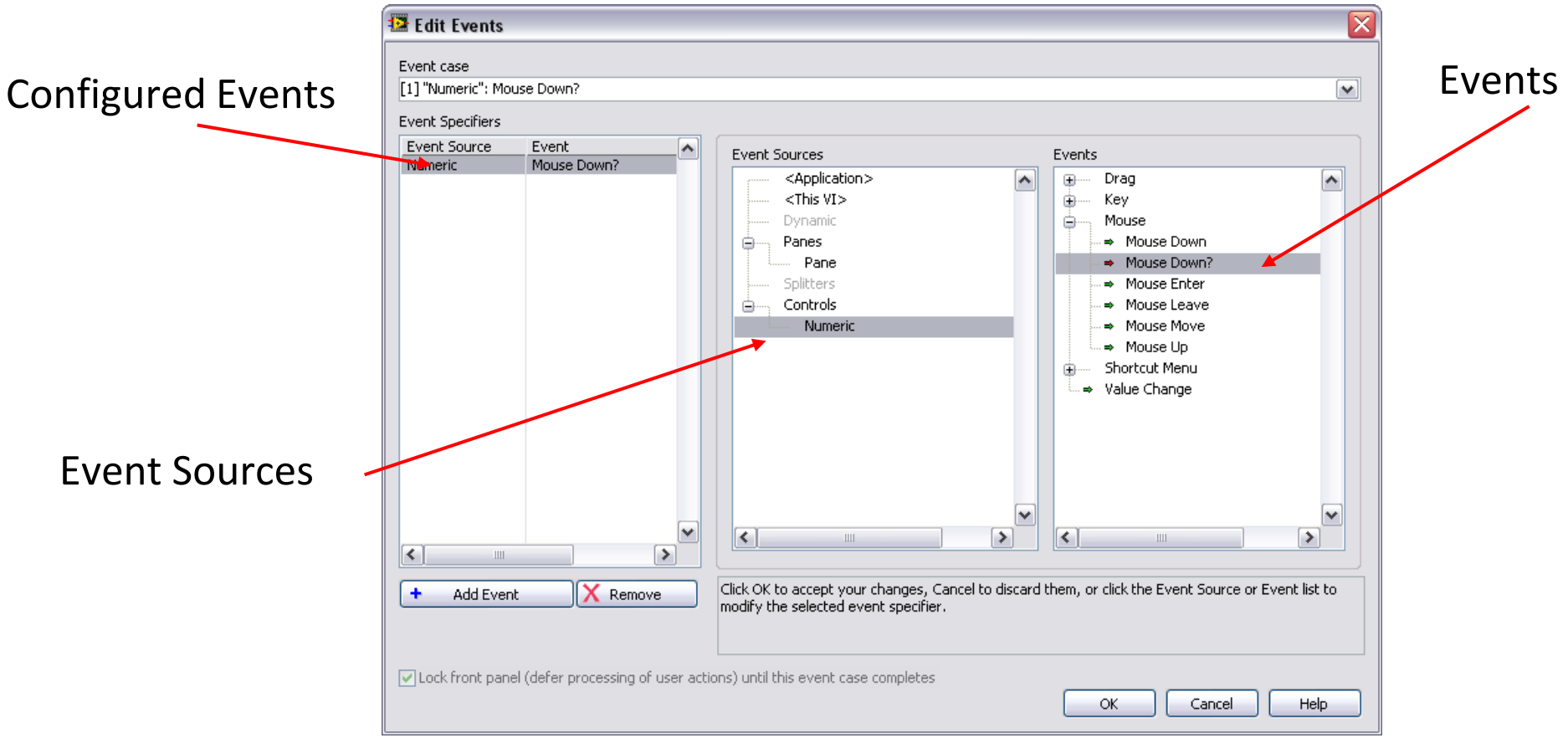


Configuring the Event Structure

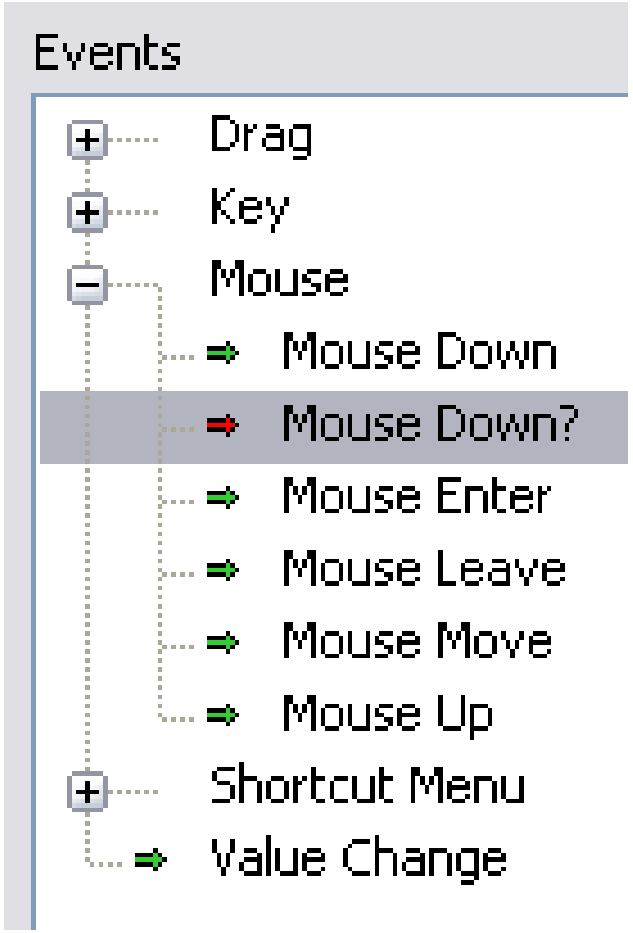




Edit Events Dialog Box



Notify and Filter Events



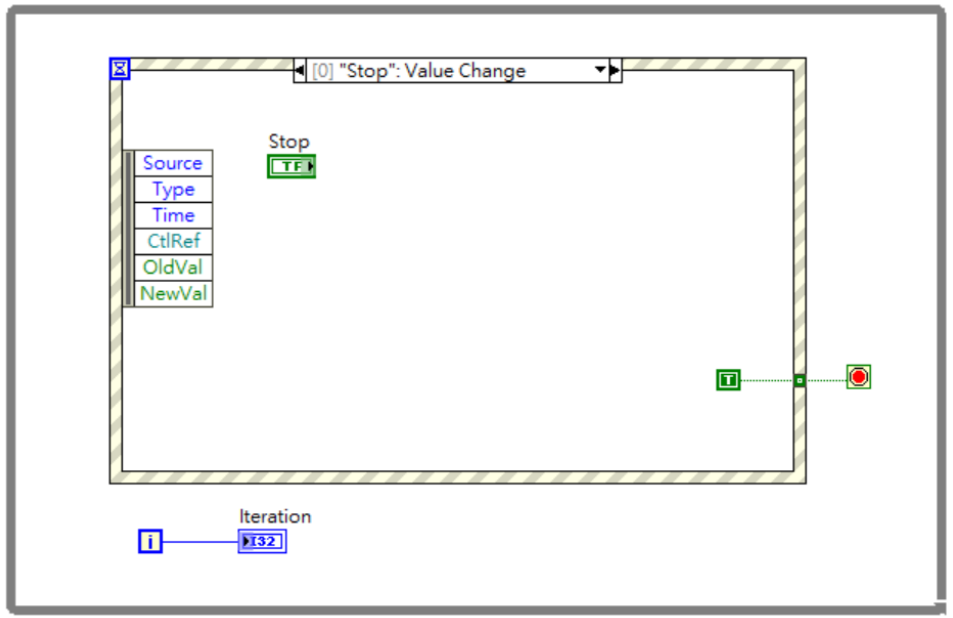
Notify Events
(green arrow)

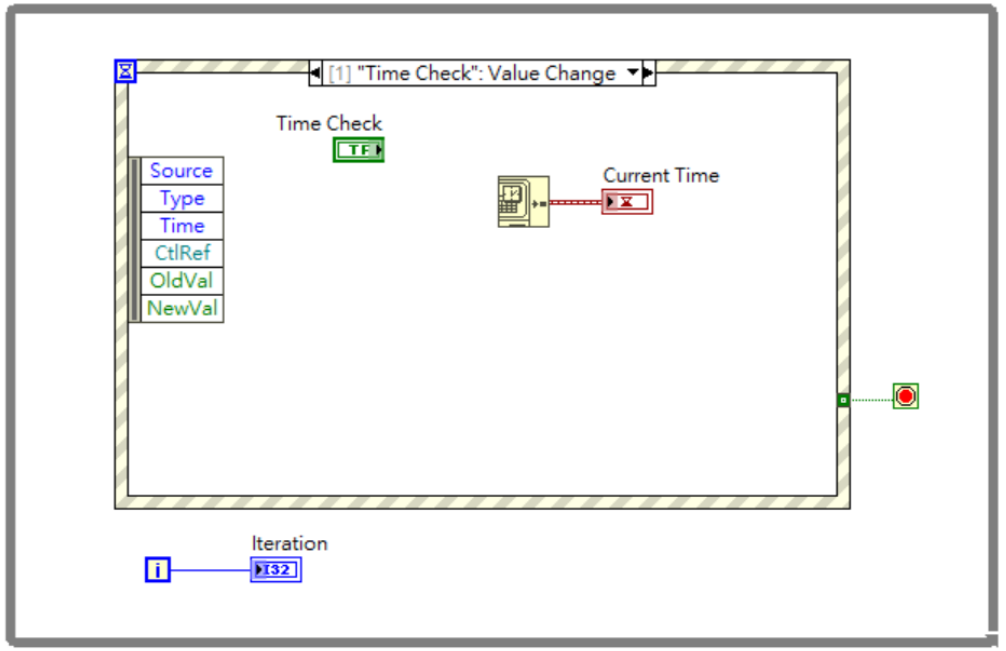
Filter Events
(red arrow)

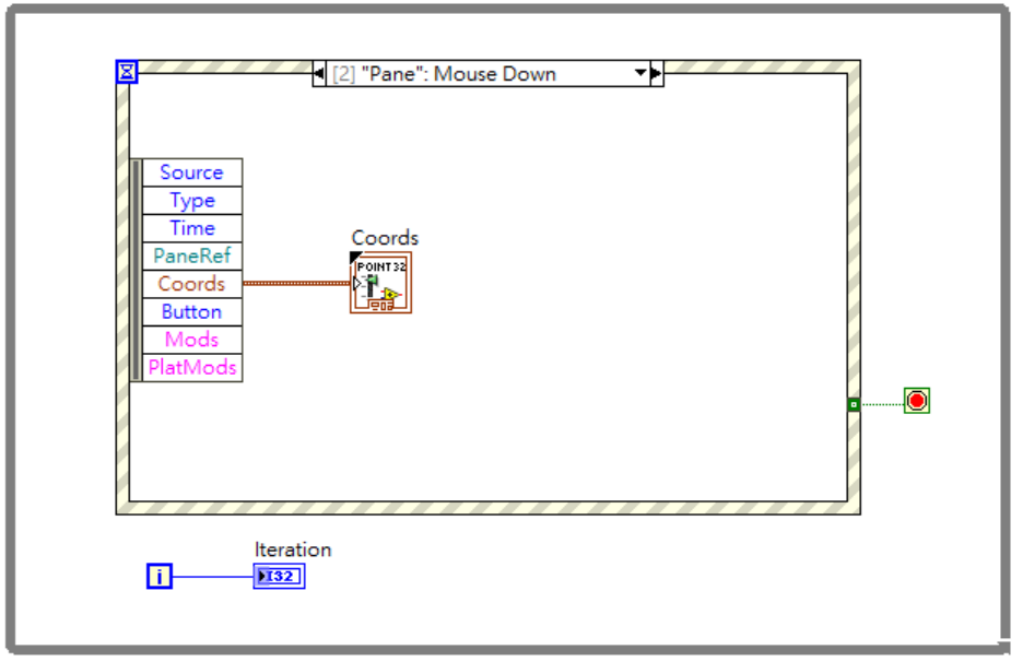
Exercise
Converting a Polling Design to an Event Structure Design

- 開啟Polling.vi，另存為UI_EventHandler.vi
- 建立一個Event Structure
  - 設定STOP Event，觀察STOP 在Event Structure 內外的差異
  - 設定Time Check Event
  - 設定Mouse Down Event

Exercise

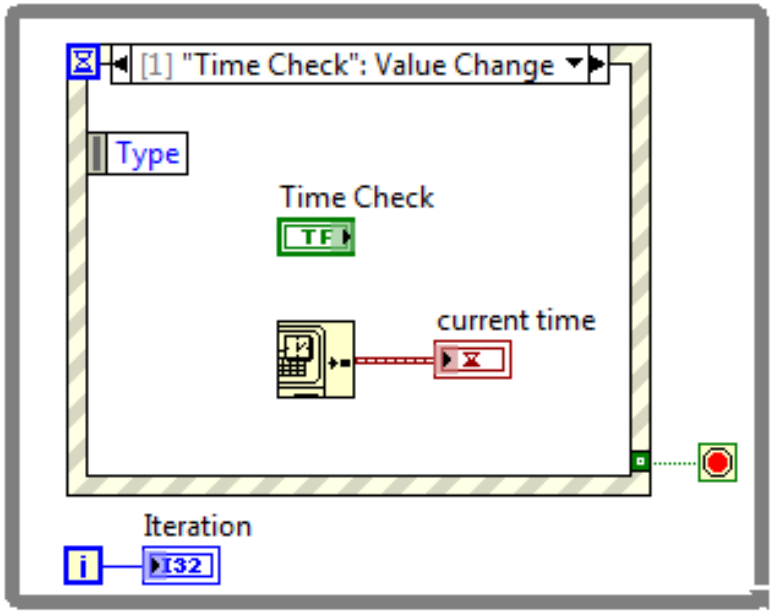






Caveats and Recommendations

- Place only one Event structure in a loop.
- Use a Value Change event to detect value changes.
- Keep event handling code short and quick.
- Place Boolean control terminals inside an event case for latched operations to work properly.
- Avoid using an Event structure outside of a loop.
- Avoid configuring two Event structures for the same event.



Modularity

- Recognize the benefits of reusing code and create a subVI with a properly configured connector pane, meaningful icon, documentation, and error handling.
  - Understanding Modularity
  - Icon
  - Connector Pane
  - Documentation
  - Using SubVIs

A. Understanding Modularity

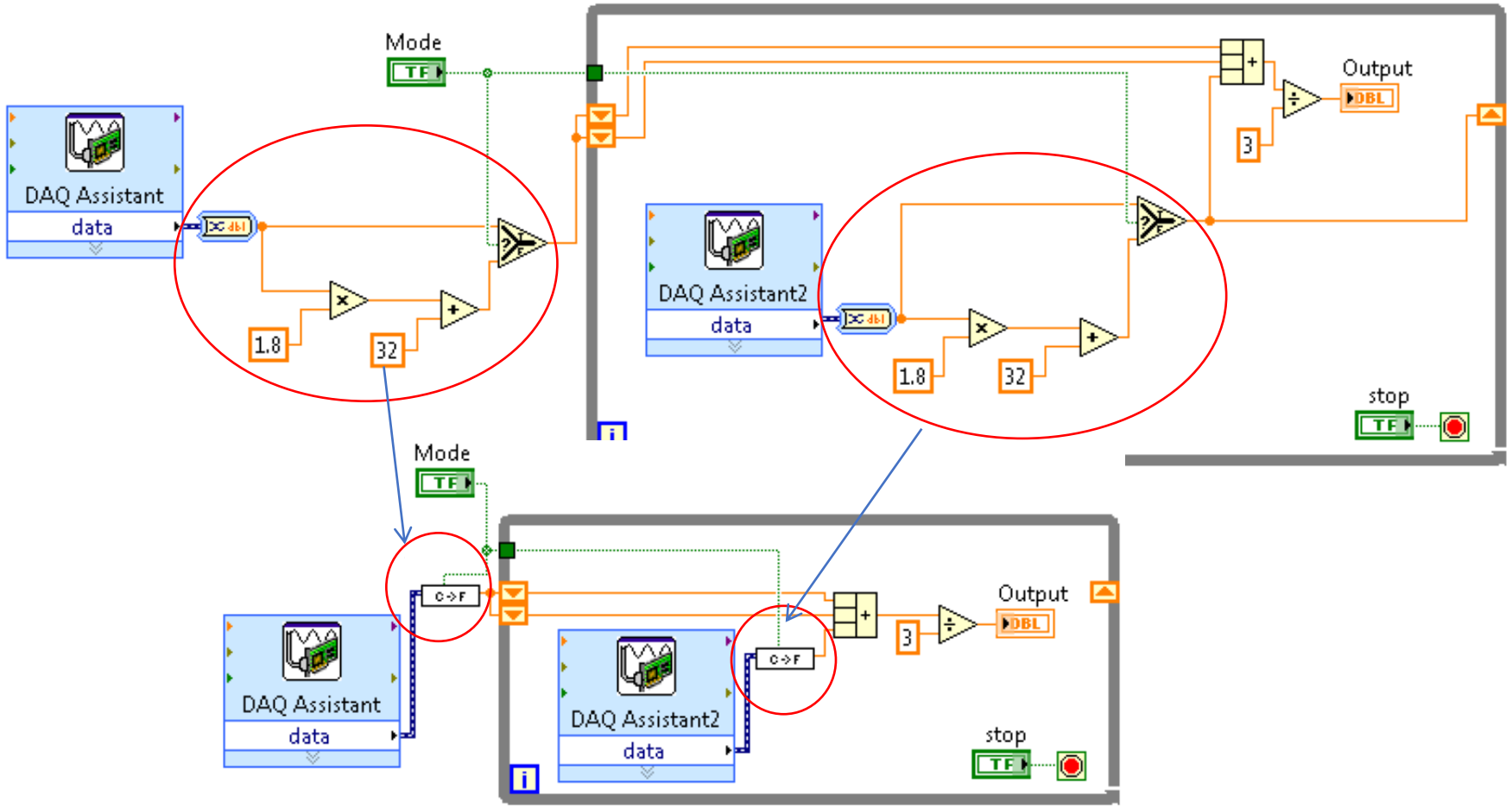
- Recognize the benefit of using modular code and identify sections of code that could be reused.
    - Modularity
    - SubVIs

Modularity and SubVIs

Modularity — The degree to which a program is composed of discrete modules such that a change to one module has minimal impact on other modules.

SubVI—A VI used within another VI.

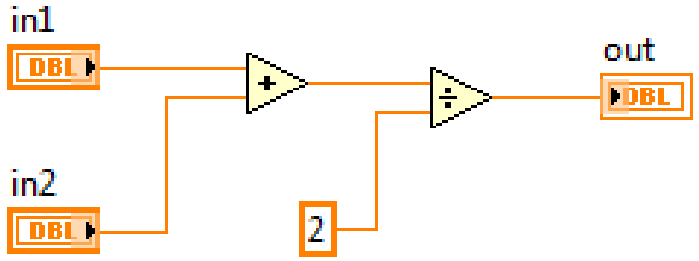
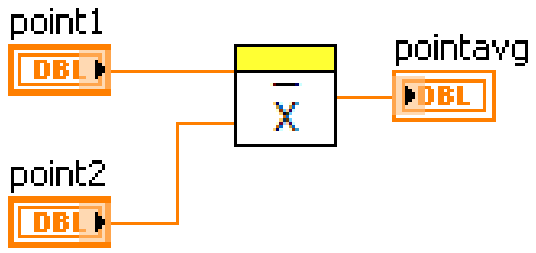
SubVIs—Reusing Code



SubVIs

Function Code	Calling Program Code

`function average (in1, in2, out)`
`{`
`out = (in1 + in2)/2.0;`
`}`
	`main`
	`{`
	`average (point1, point2, pointavg)`
	`}`

SubVI Block Diagram	Calling VI Block Diagram
	

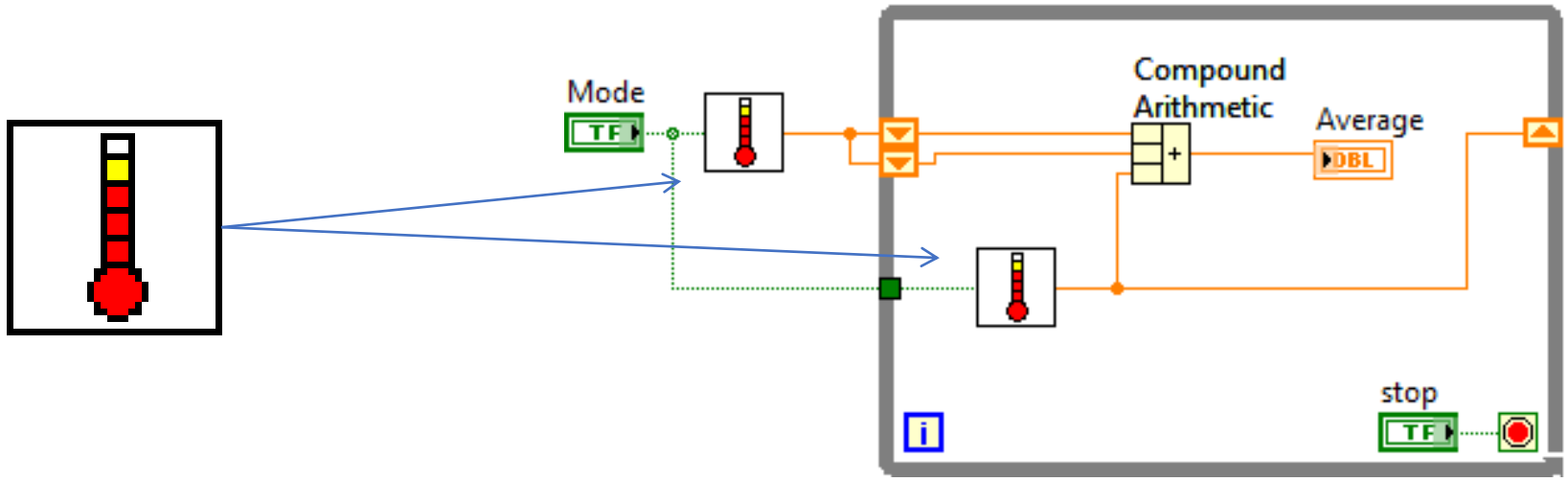
B. Icon

- Recognize characteristics of a good icon and use the LabVIEW Icon Editor to create a custom icon.

  - Characteristics of a Good Icon
  - Using the Icon Editor

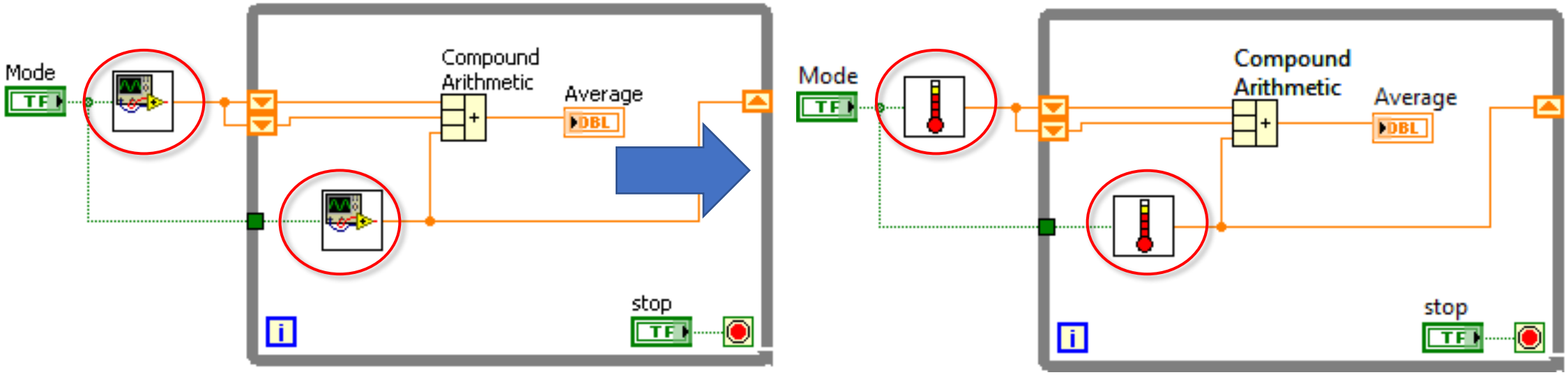
Purpose of Icon

- Graphical representation of a VI
- Identifies the subVI on the block diagram of the VI

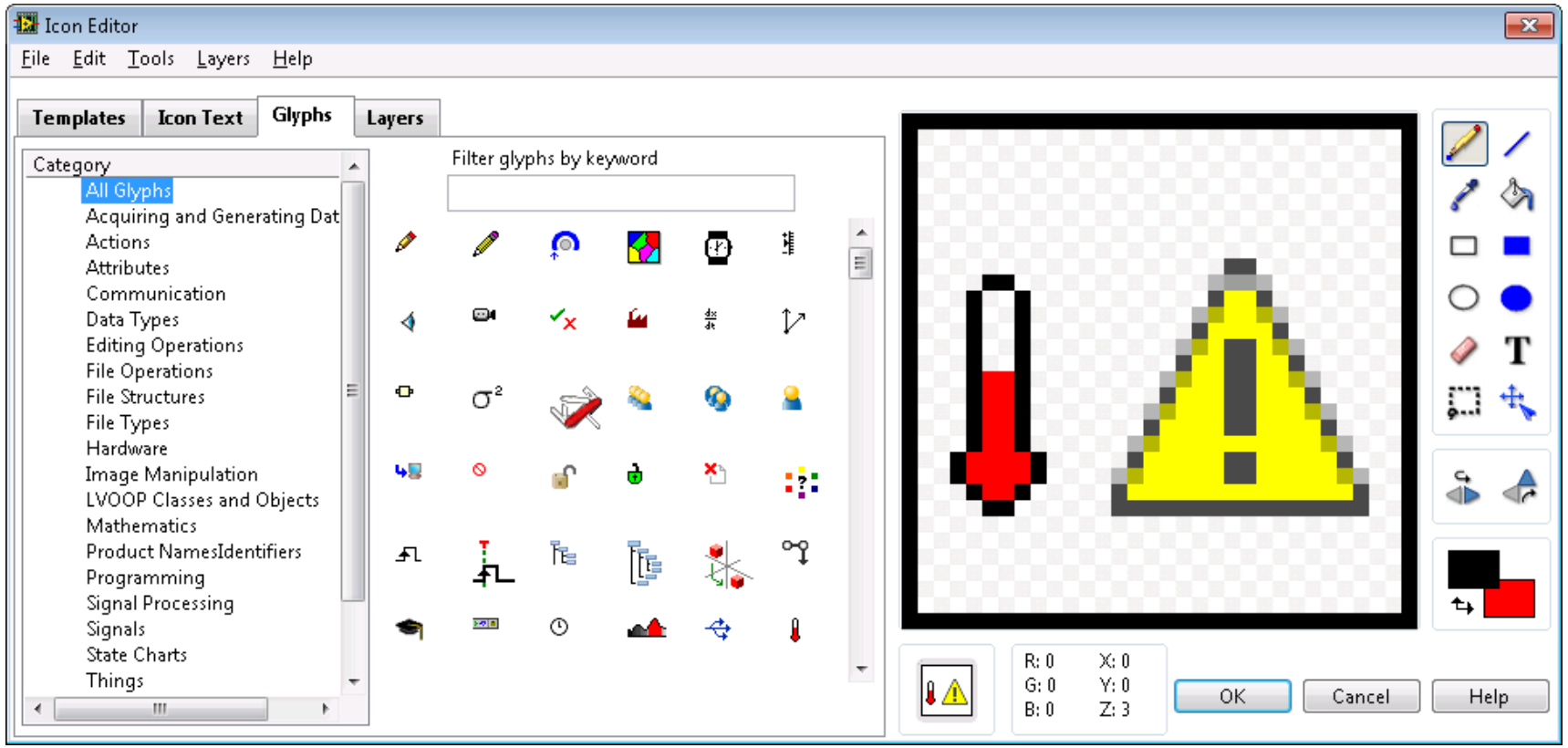


Characteristics of a Good Icon

Good icons convey the functionality of the VI.



Creating Icons—Icon Editor



Demonstration
Creating an Icon

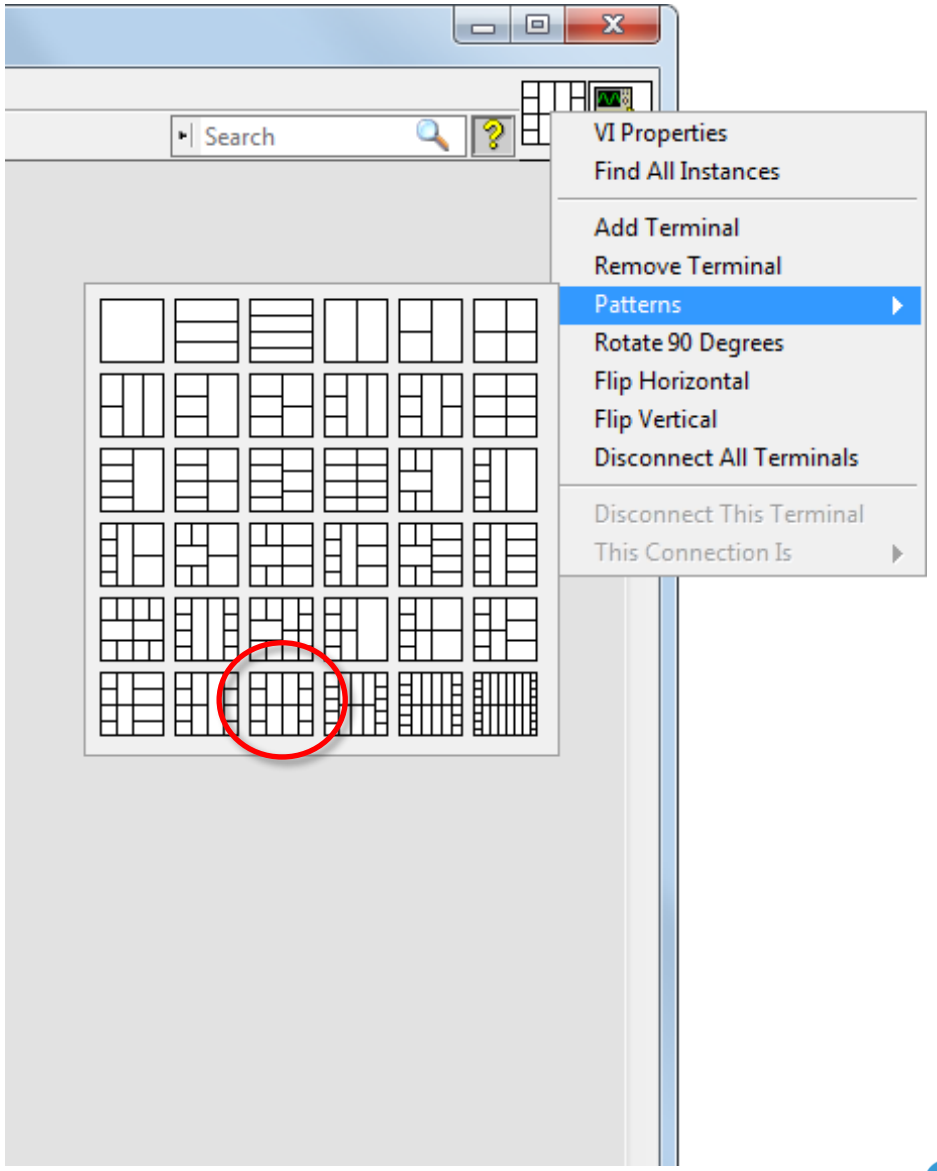
- Use the LabVIEW Icon Editor to create a custom icon.

C. Connector Pane

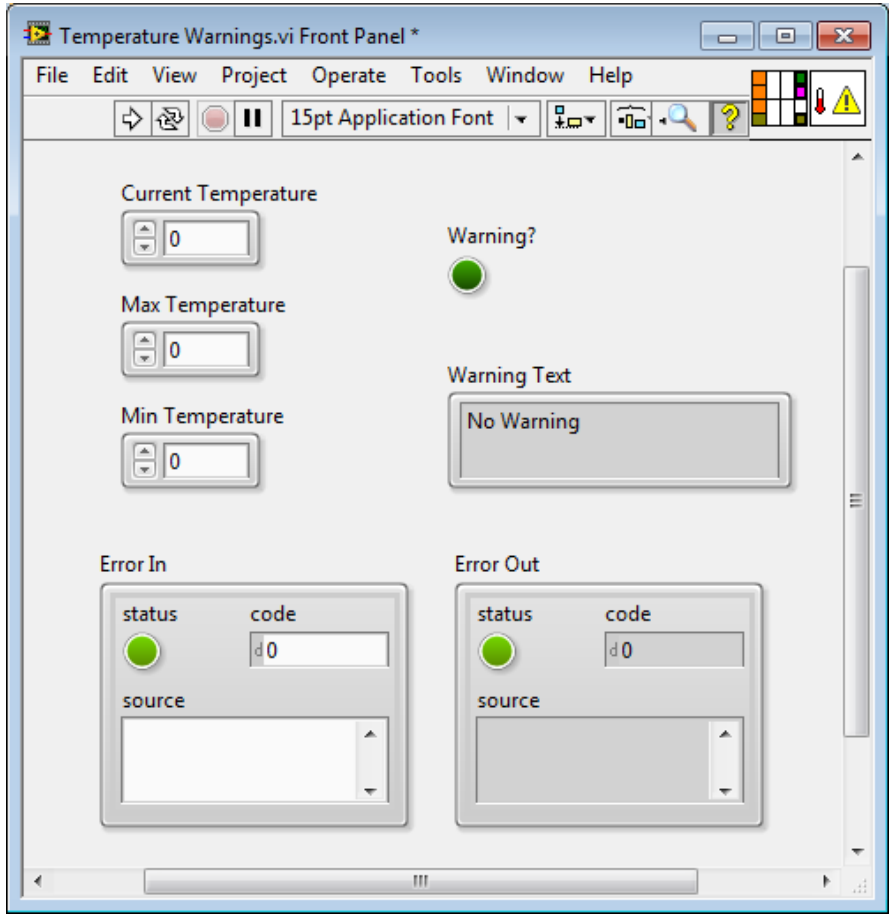
- Select and configure a connector pane for a subVI.
  - Patterns
  - Standards

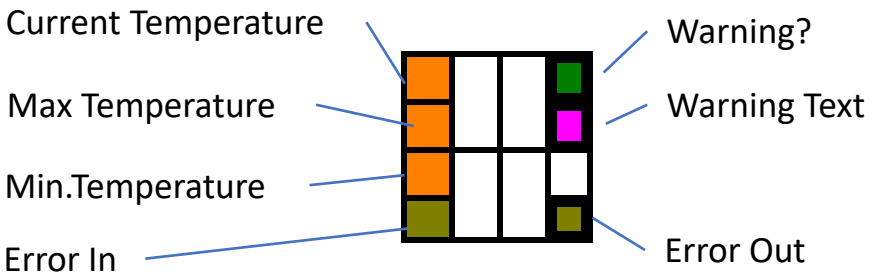
Patterns

- Displayed next to icon
- Select from different patterns

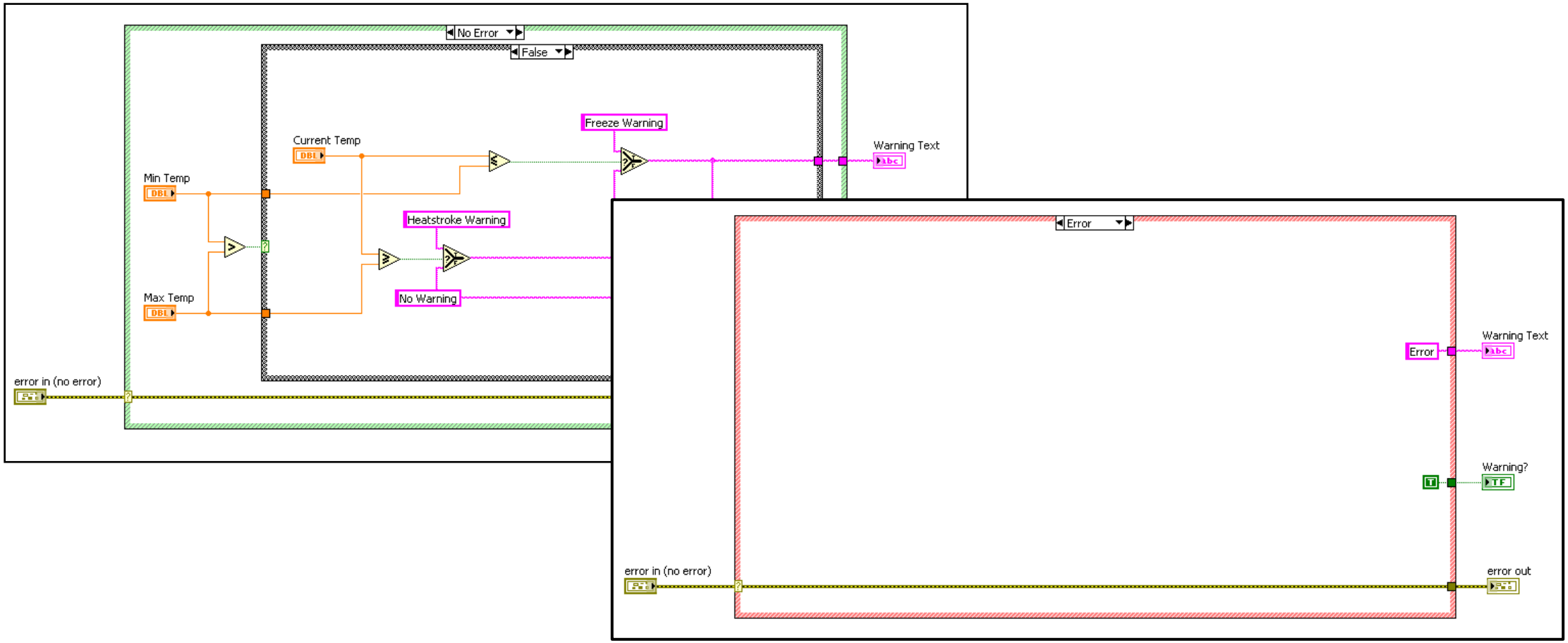


Assigning Terminals

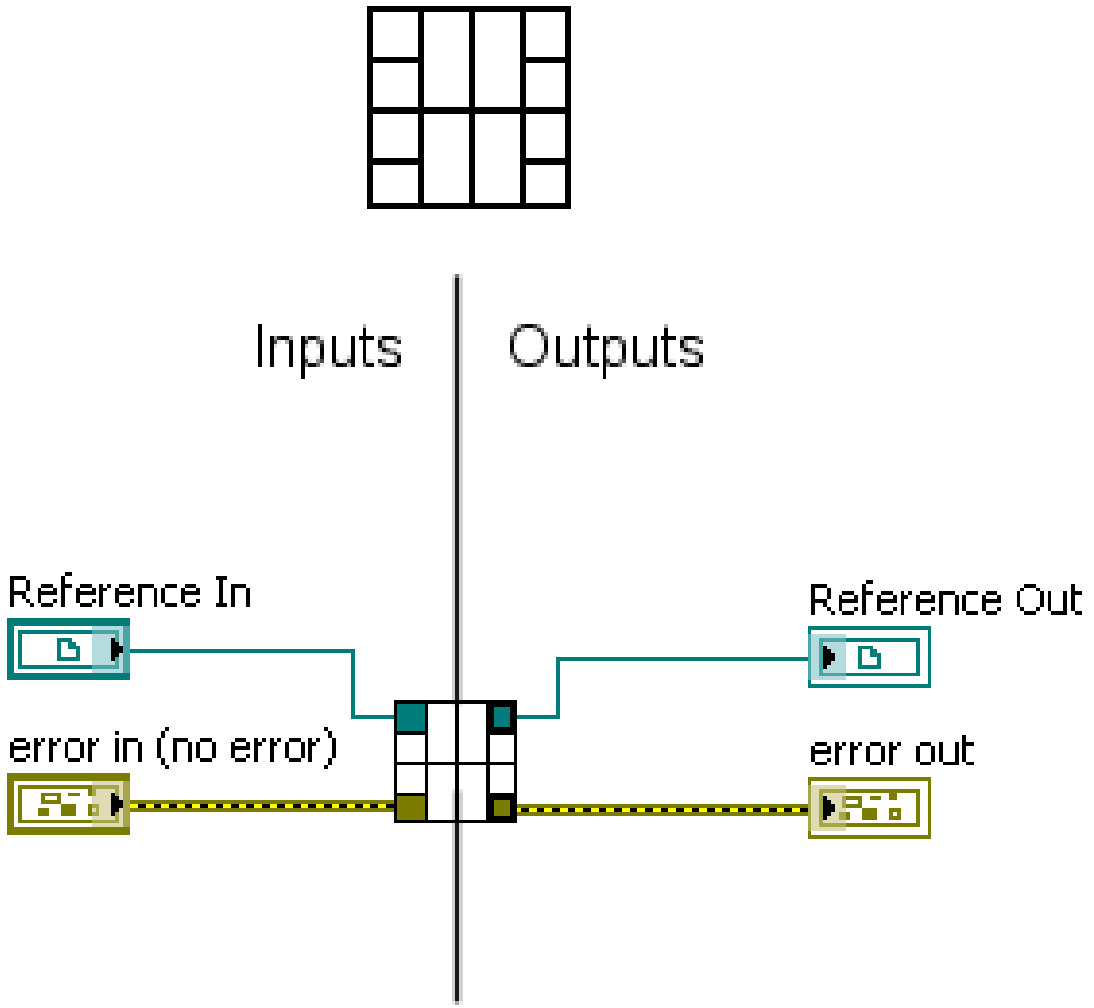




Handling Errors



Standards

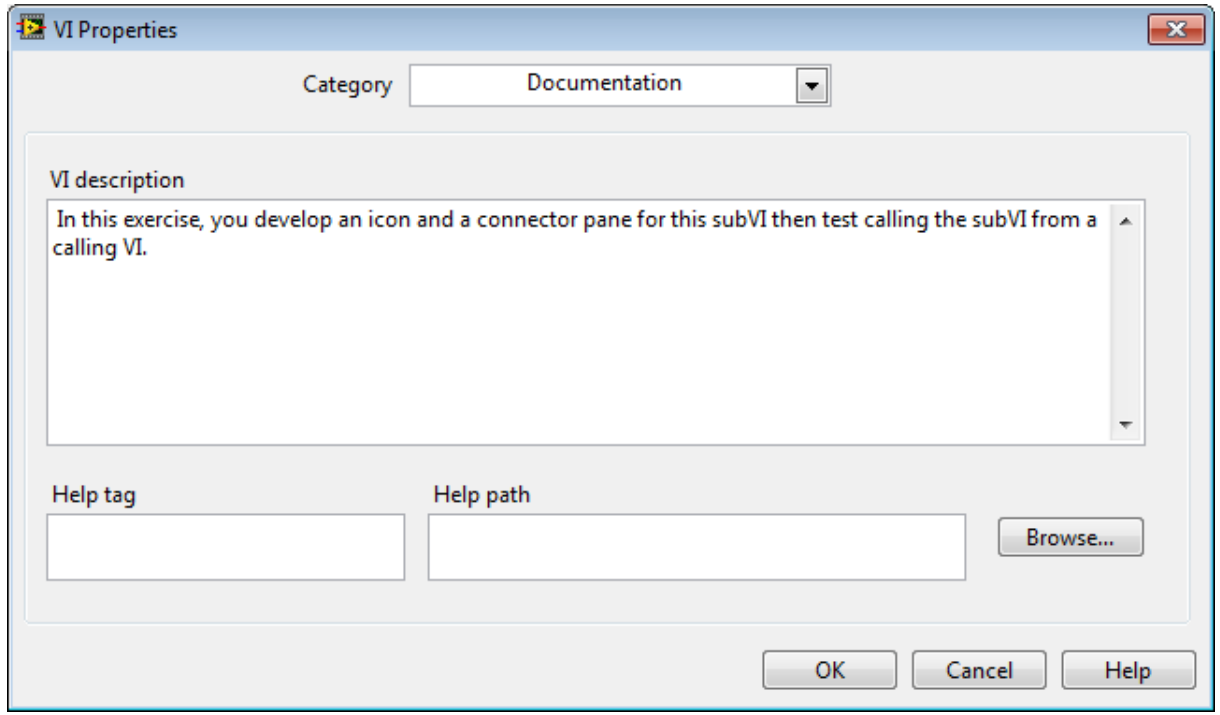

Inputs
Outputs
Reference In
error in (no error)
Reference Out
error out

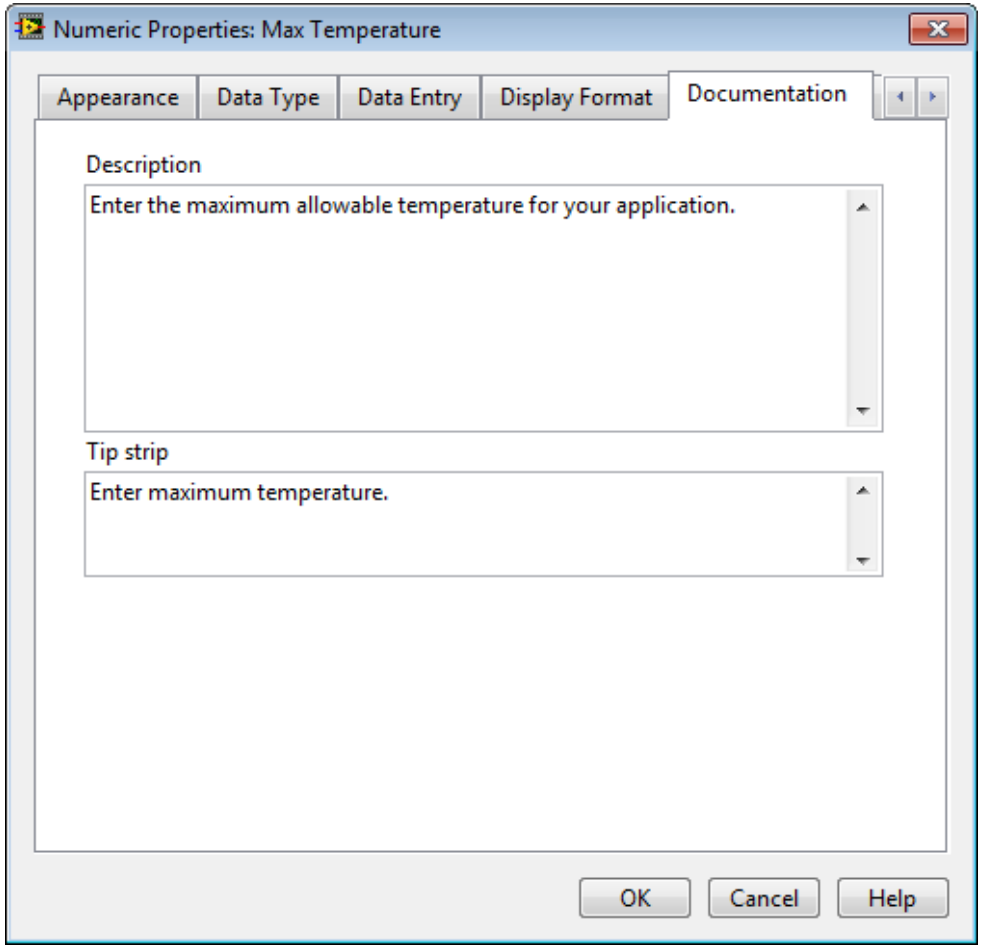
D. Documentation

- Explain how to document code in LabVIEW using descriptions and tip strips, and describe four methods for documenting code on the block diagram.

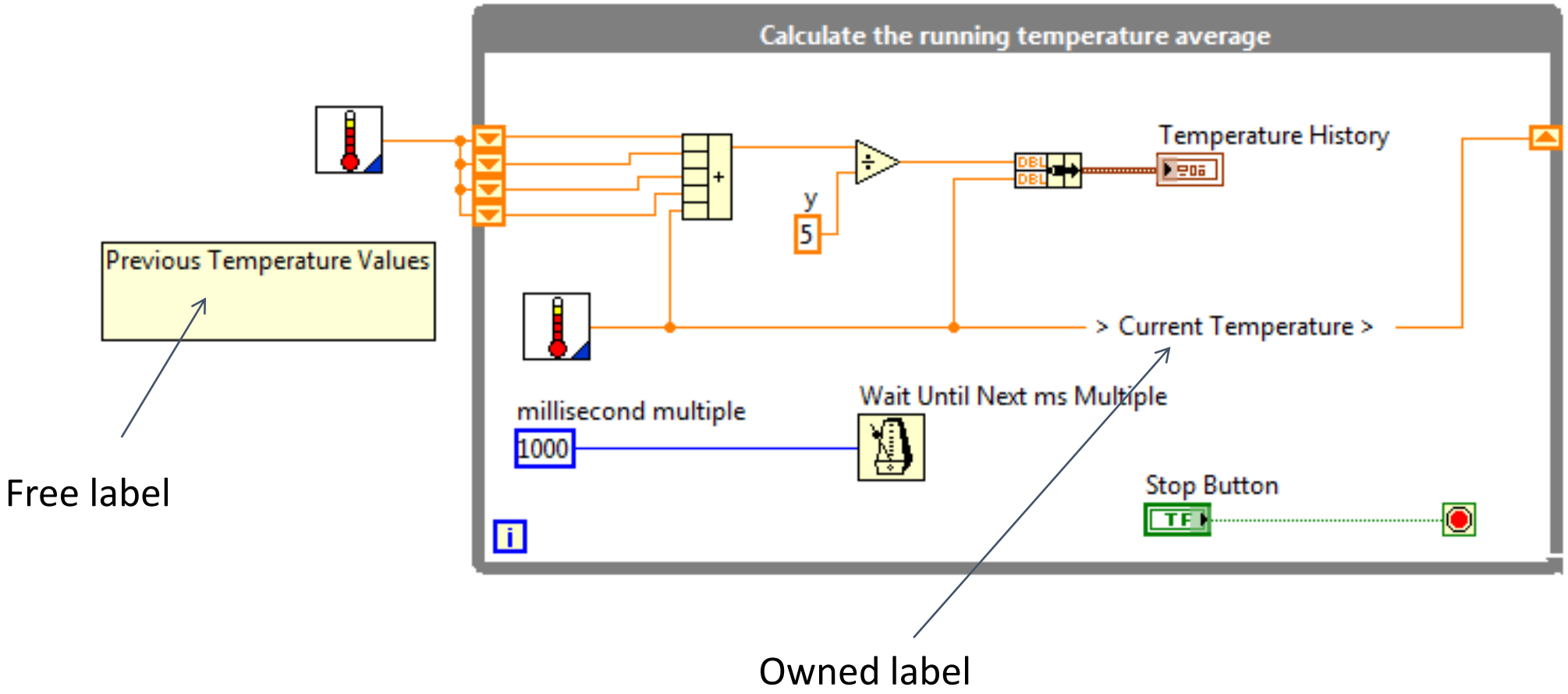
  - VI Descriptions and Tip Strips
  - Labels

Creating Descriptions and Tip Strips





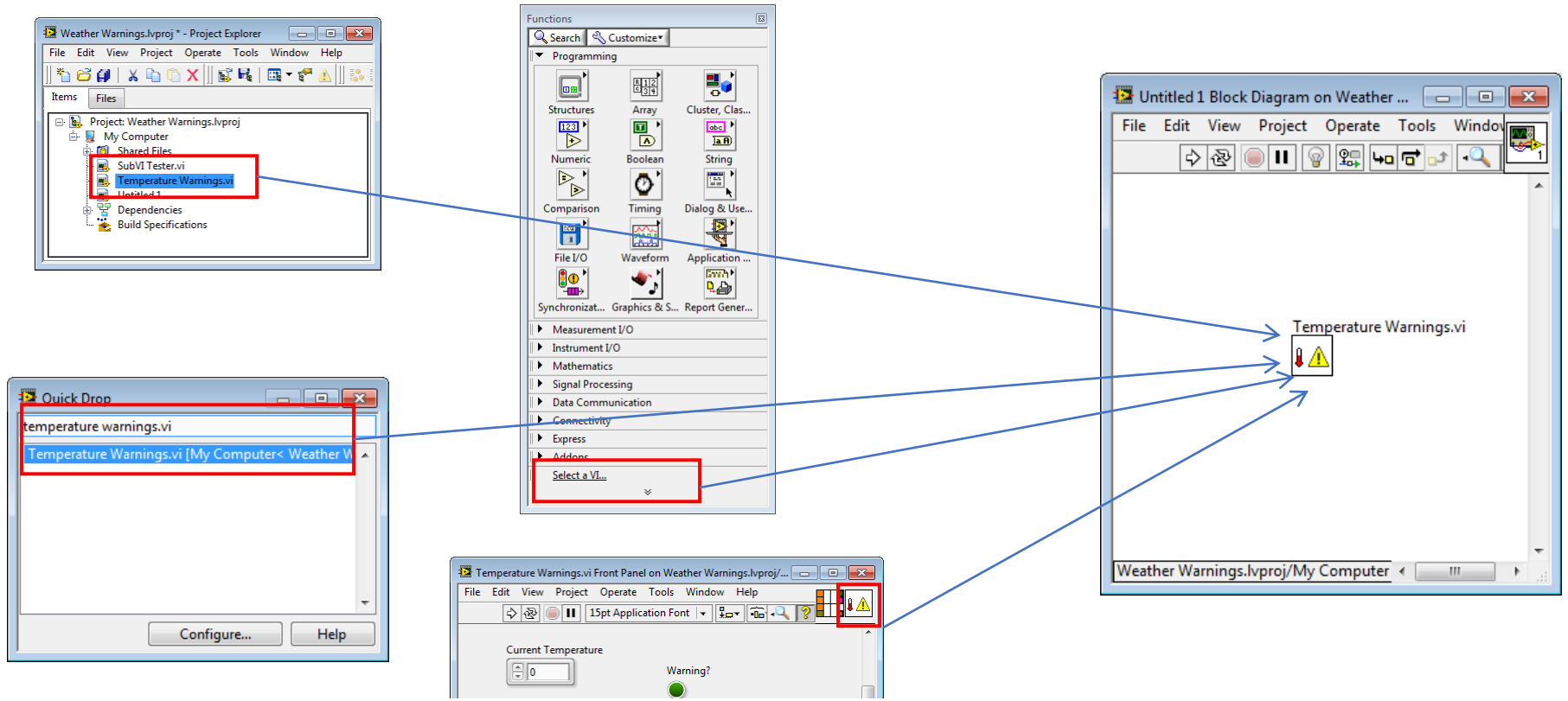
Documenting Block Diagram Code



E. Using SubVIs

- Demonstrate how to place subVIs on the block diagram, explain terminal settings and error handling, and create subVIs from a section of existing code.
  - Placing SubVIs on the Block Diagram
  - Terminal Settings
  - Handling Errors
  - Creating from a Section of Block Diagram

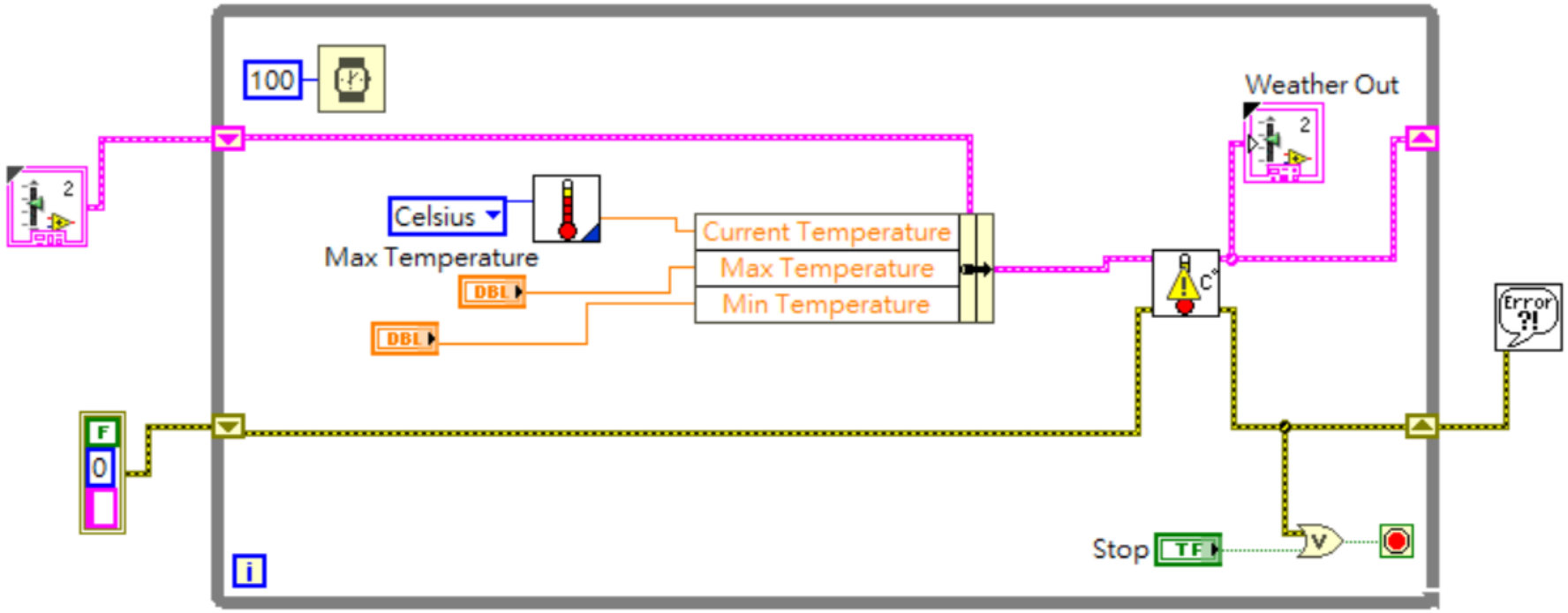
Placing SubVIs on the Block Diagram



Exercise
Temperature Warnings VI – As SubVI

- 將Temperature Warning.vi 做成SubVI
- 打開SubVITester.vi，放入Temperature Warning (SubVI), Thermometer(demo), WeatherData並接上

Exercise


100
2
Celsius
Max Temperature
DBL
DBL
Current Temperature
Max Temperature
Min Temperature
Weather Out
2
C°
Error
?!
F
0
i
Stop
TF

Using State Programming

- Describe the functionality represented by a state transition diagram.
    - Why Use State Programming?
    - State Transition Diagrams

Why Use State Programming?

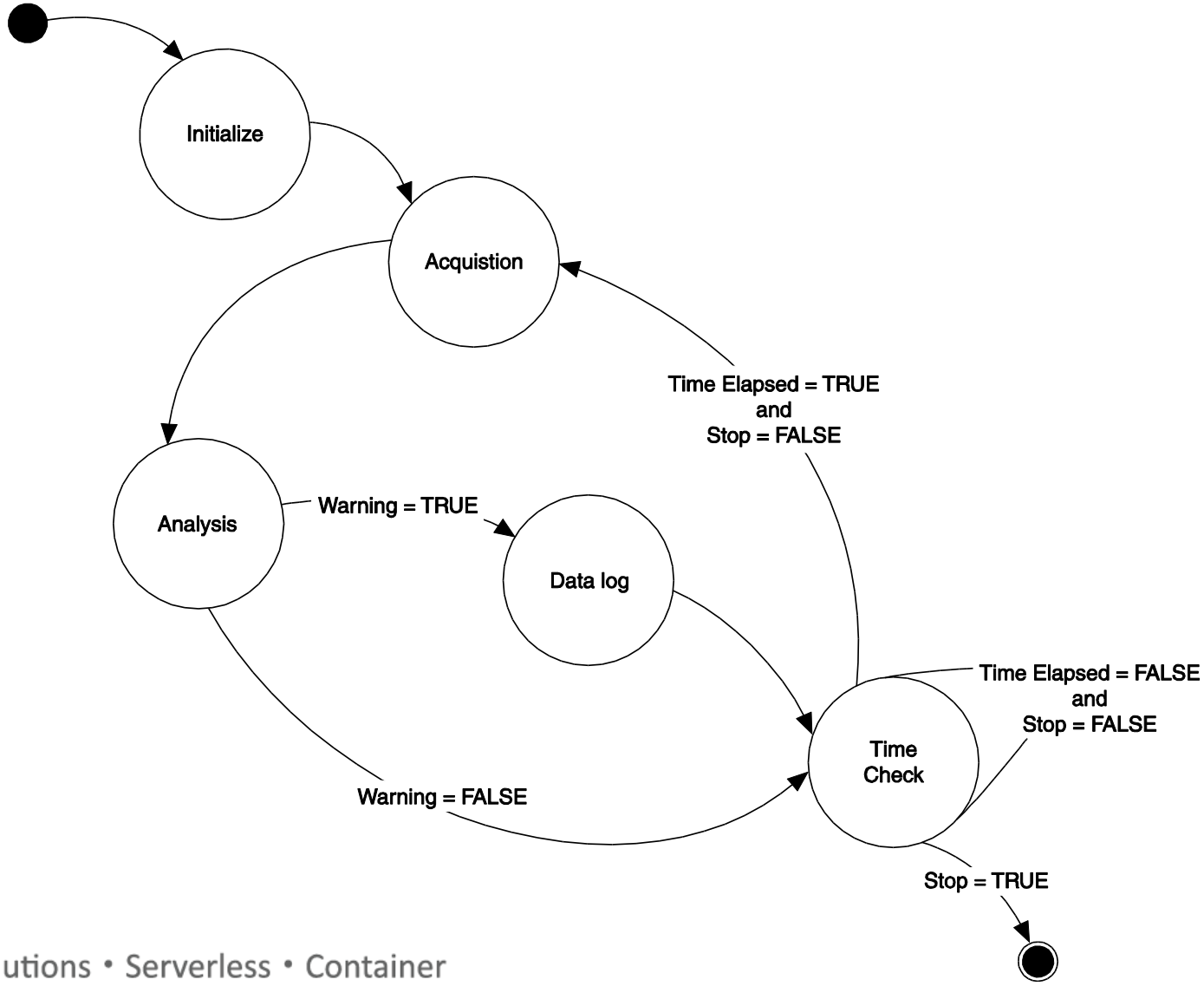
- Change the order of a sequence
- Repeat an item in sequence more often than others
- Execute only if certain conditions are met
- Stop the program immediately

State Transition Diagram

State—Part of a program that satisfies a condition, performs an action, or waits for an event

Transition—Condition, action, or event that causes the program to move to the next state

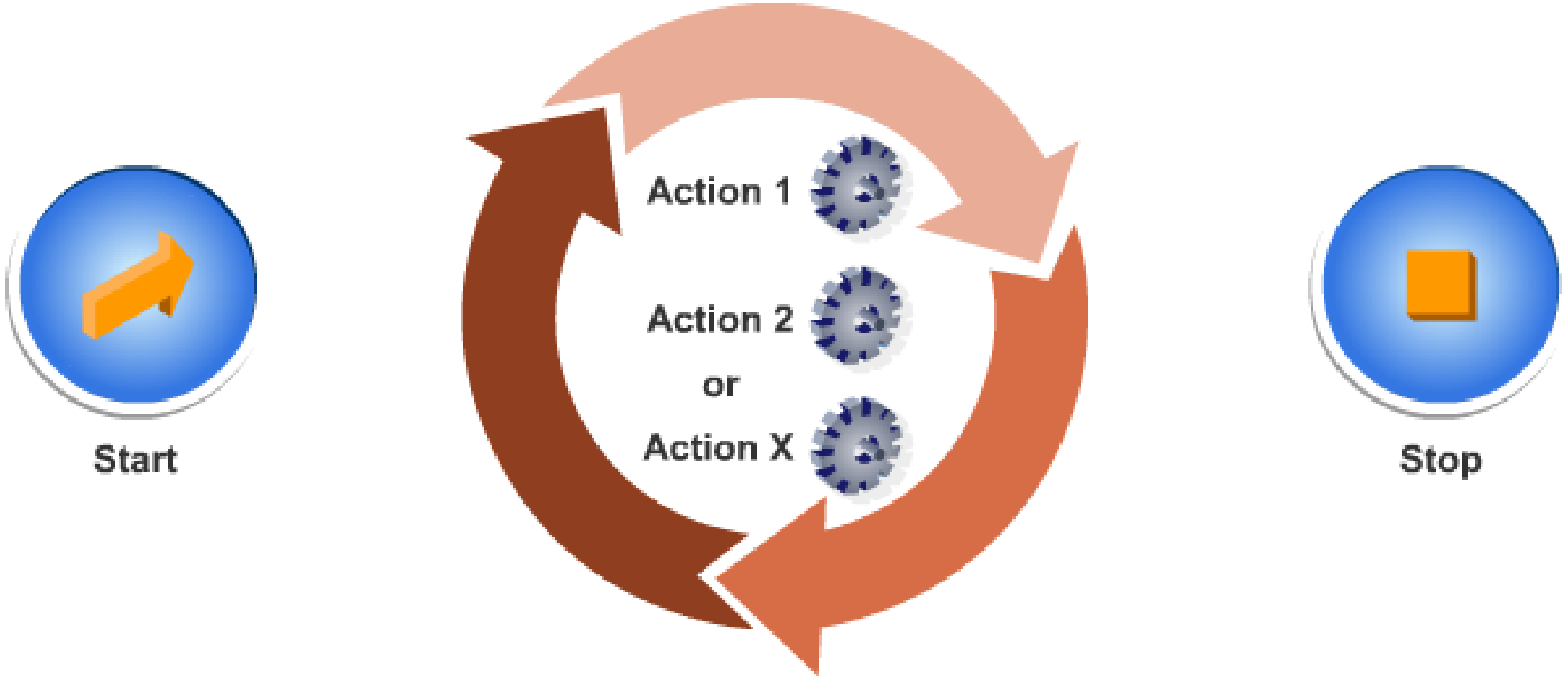
State Transition Diagram



State Machines

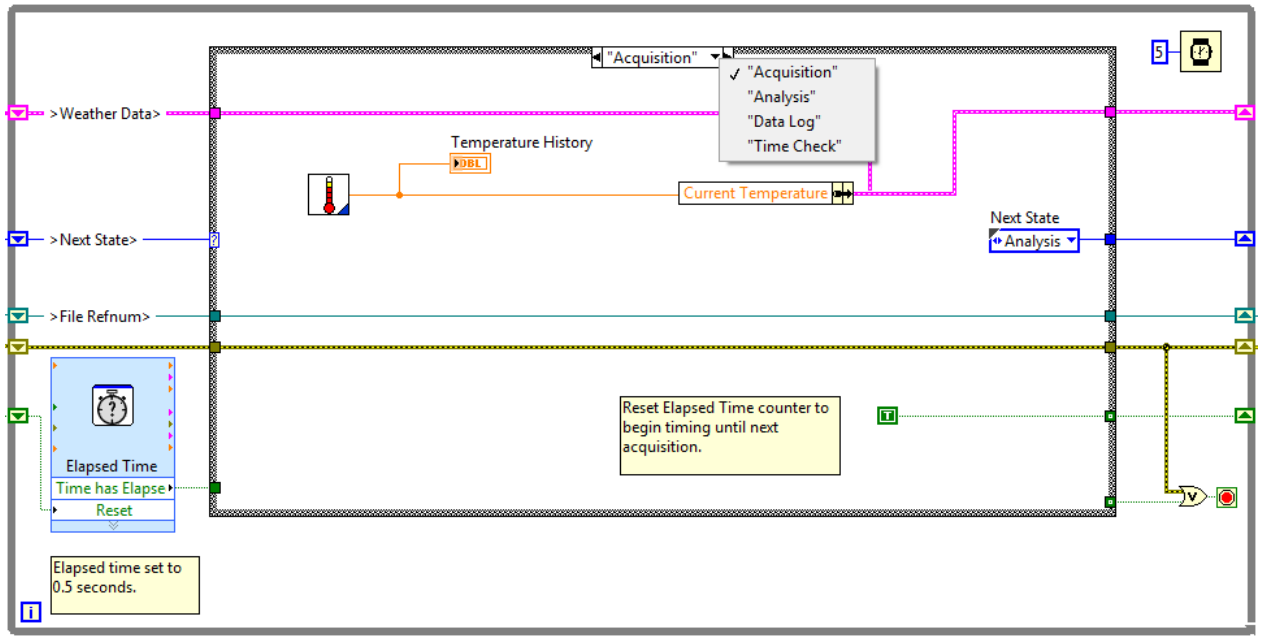
- Determine when to use a state machine.
  - What is a State Machine?
  - When to Use a State Machine?
  - Building a State Machine
  - Event-Based State Machine
  - Simple State Machine Project Template

What is a State Machine?

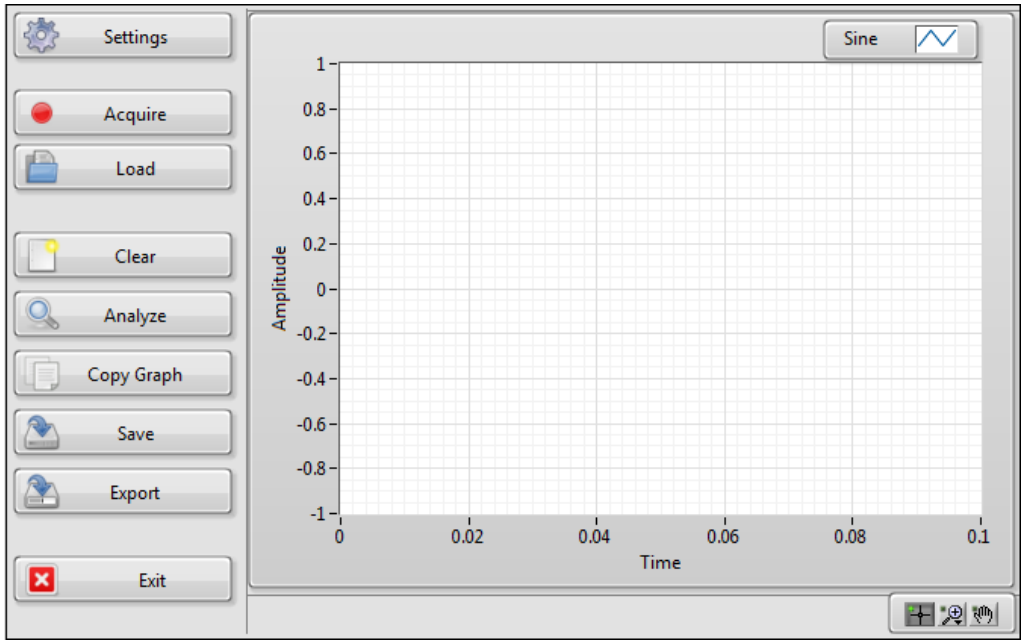


When to Use a State Machine

Sequential Process



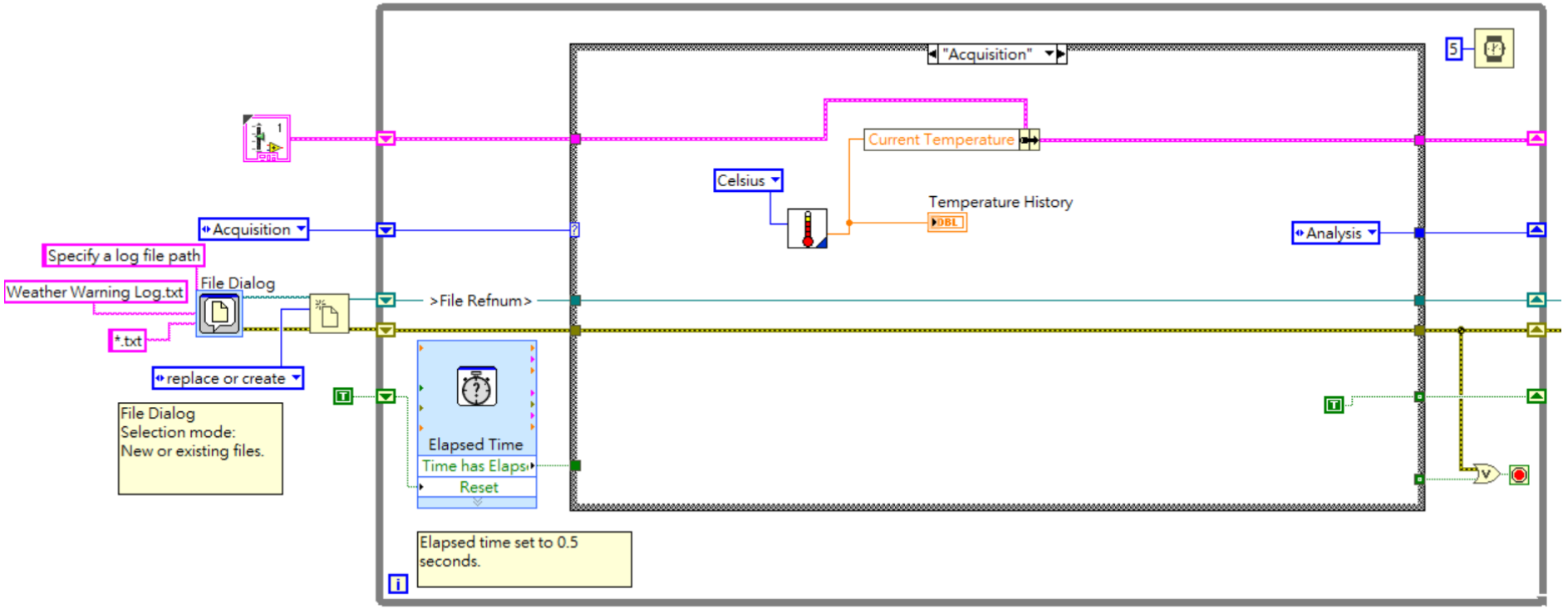
UI-Driven Process



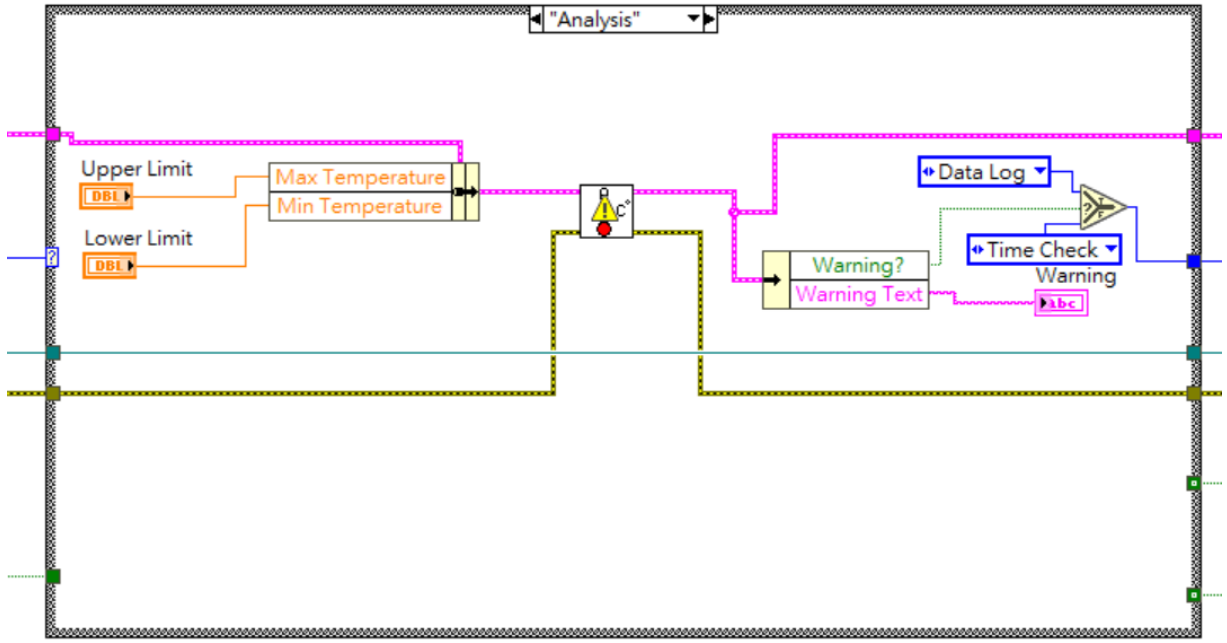
Exercise

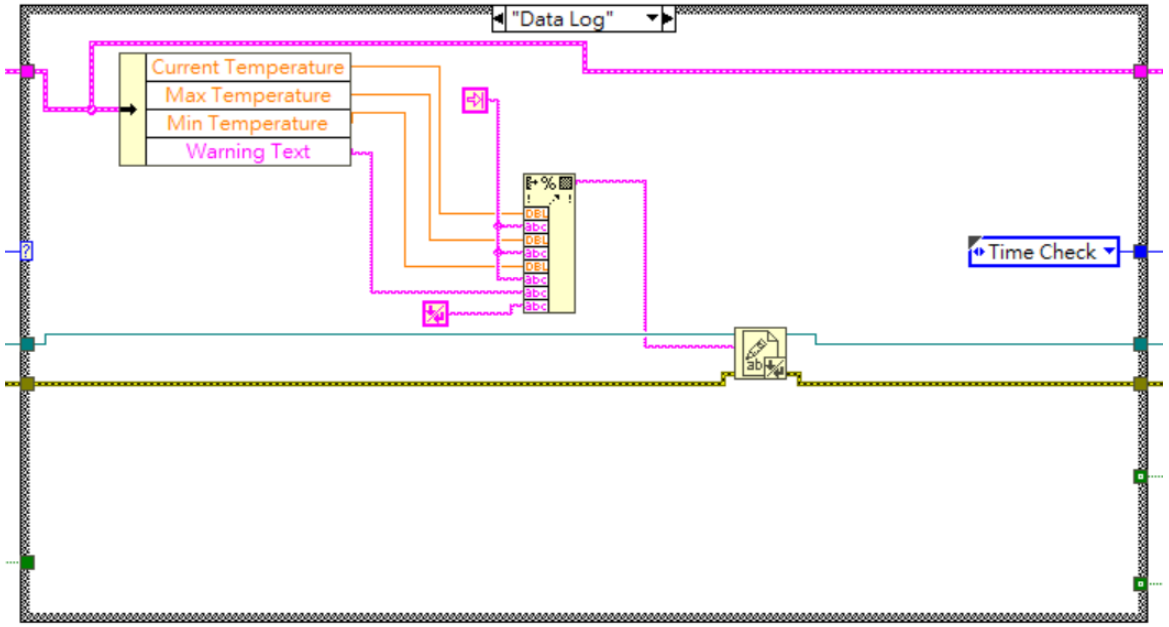
- 打開Weather Station UI.vi
- 放入Weather Data.ctl
- 建立Enum，包含Acquisition, Analysis, Data Log, Time Check，存為Typedef
- 設定Acquisition，放入Thermometer (demo).vi，取得Current Temperature，下一個State為Analysis
- 設定Analysis，放入Temperature Warning (SubVI)，連接Upper limit、Lower limit、Warning?和Warning Text，Warning?為True進入Data Log，False進入Time Check
- 設定Data Log，將前述資訊寫入檔案
- 設定Time Check，Time has elapsed 為True進入Acquisition，False進入Time Check，連接Stop button
- Run vi

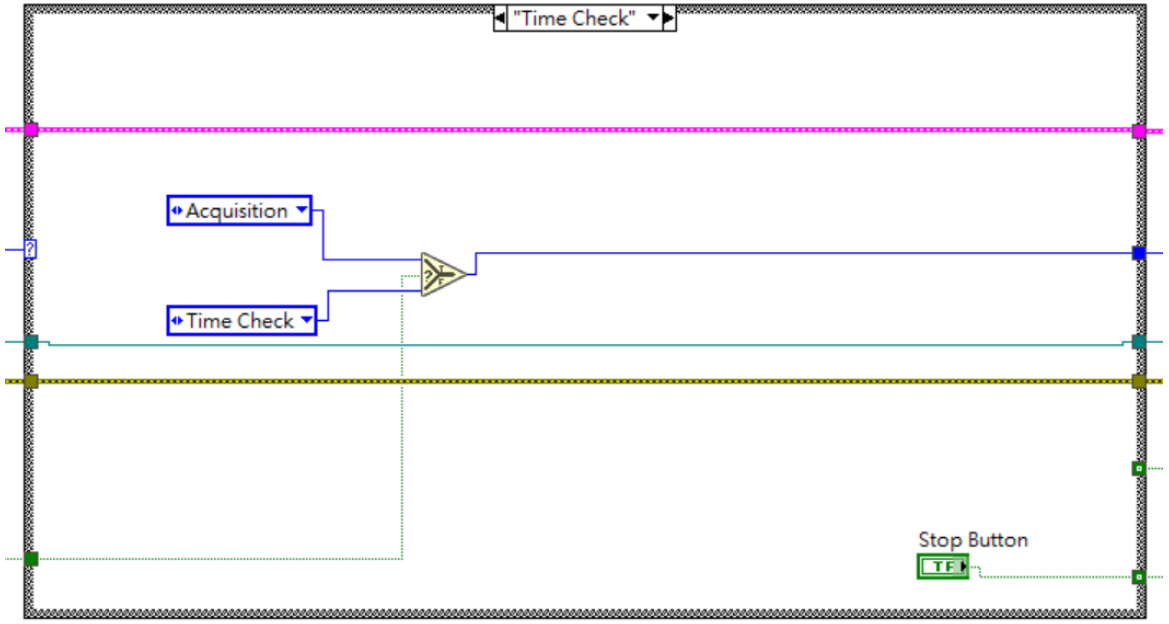
Exercise



Exercise







Thank you for kind attention~

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