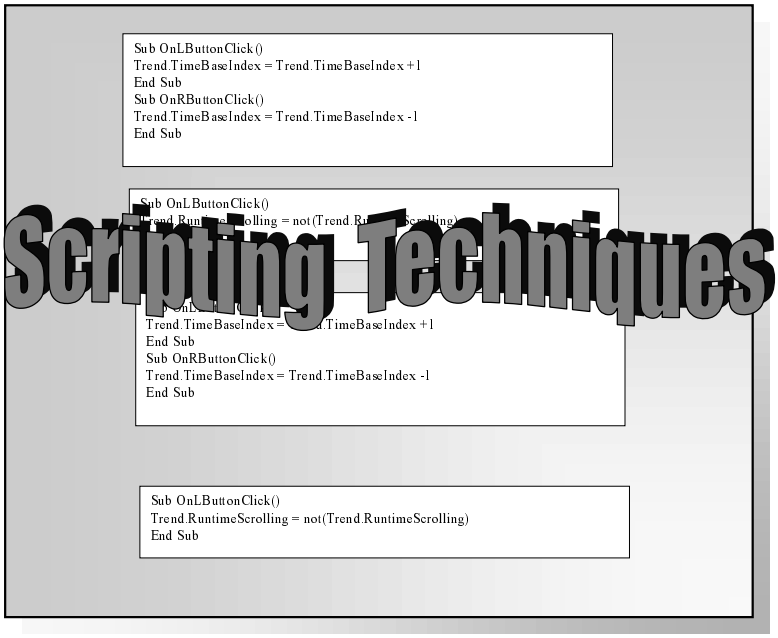
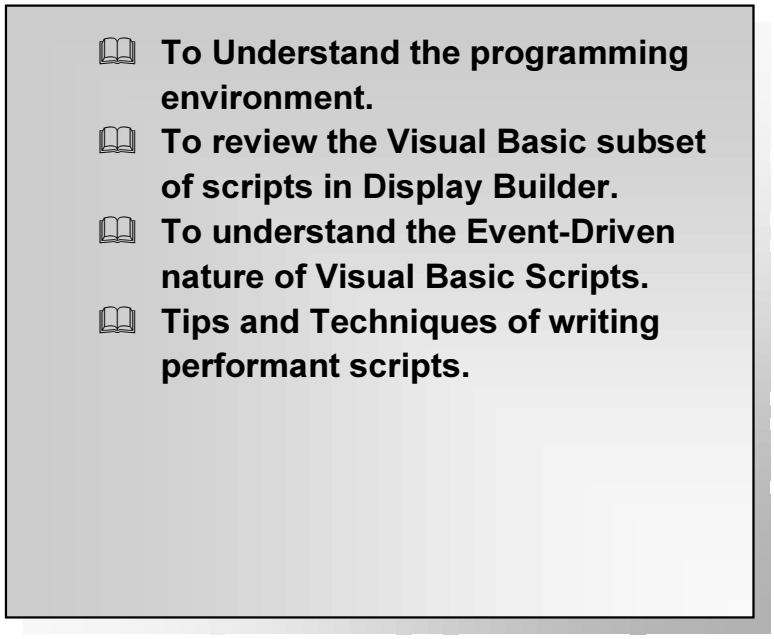

Sub OnLButtonClick()
Trend.TimeBaseIndex = Trend.TimeBaseIndex +1
End Sub
Sub OnRButtonClick()
Trend.TimeBaseIndex = Trend.TimeBaseIndex -1
End Sub
Sub OnLButtonClick()
Trend.TimeBaseIndex = Trend.TimeBaseIndex +1
End Sub
Sub OnRButtonClick()
Trend.TimeBaseIndex = Trend.TimeBaseIndex -1
End Sub
Sub OnLButtonClick()
Trend.RuntimeScrolling = not(Trend.RuntimeScrolling)
End Sub
Scripting Techniques

♦ Objectives



Main Idea

The topics covered in this module help you understand how to program using the Display Builder and how to script performant displays .

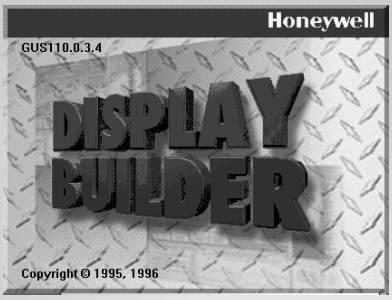
Objectives

At the end of this module you will be able to do the following:

- To Understand the programming environment.
- To review the Visual Basic subset of scripts in Display Builder.
- To understand the Event-Driven nature of Visual Basic Scripts.
- Tips and Techniques of writing performant scripts.

.

Introduction



- The Display Builder uses a subset of Microsoft Visual Basic.

Display Builder

- When Honeywell introduced the Global User Station and the Display Builder to replace the TDC Picture Editor, it opened an entire new world of programming. This, of course, brings Microsoft Technology into the playing field.
- This “new” programming skills is none other than Visual Basic.
- Release 120 of the Display Builder is compliant to Visual Basic 5.

Event-Driven Programming

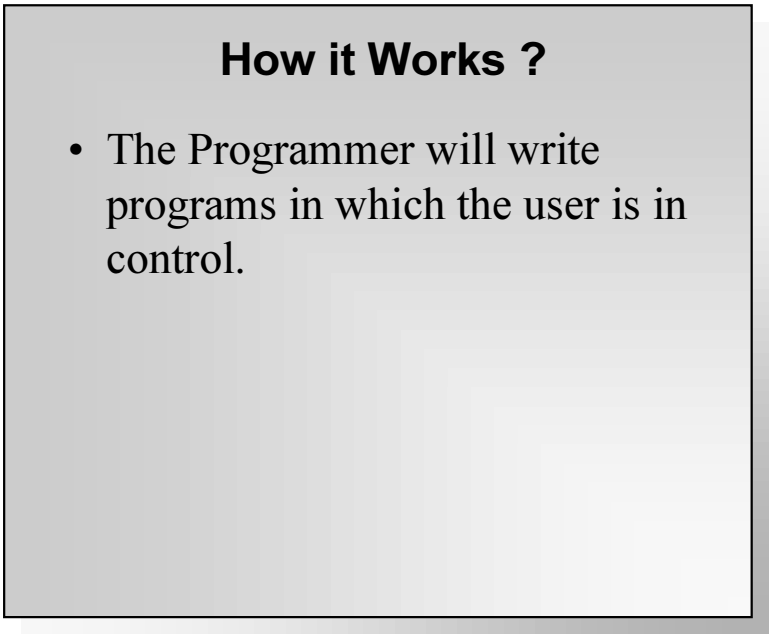
What is Event Programming ?

- Event-driven programming is like playing goalie in a soccer game. Like a goalie, a program responds to events like keyboard actions, a mouse click or touch screen actions.

Event-Driven Programming

- Visual Basic was created to be driven by events; these events can be called subroutines. The Display Builder is a subset of Visual Basic and thus is considered to be an event-driven programming environment.

Event-Driven Programming



Event-Driven Programming

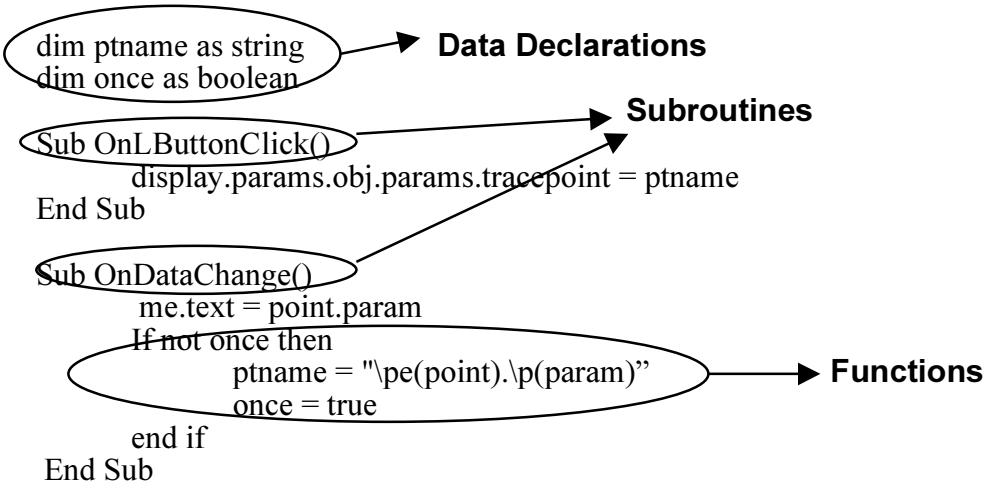
- The user is in control because it is the user who will write the programs which will execute based on events.
- Some programs will execute based on the input from the process that it will monitor and some will execute based on user events.
- In this model of programming, it is the occurrence of events that control the flow of the program.
- In GUS, the user can use events like key press or mouse clicks or touch screen. Process changes are also considered events.

Event-Driven Programming

What is a *Script* ?

- A *Script* is defined as a collection of functions, subroutines and data declarations.
  - A Script will always be associated with a Display Object.
- A *Display Object* is defined as a primitive element in a display, or the display itself.

Example of a Script

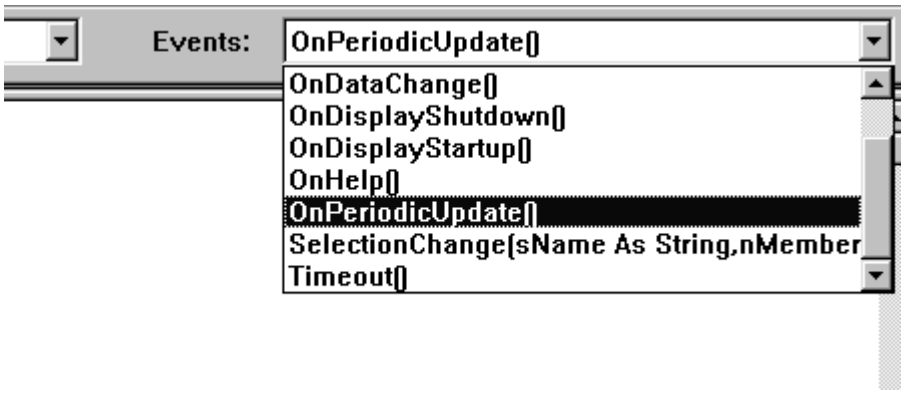


Event-Driven Programming

What is an *Event* ?

- Each Display Object has a predefined set of events that the Visual Basic script responds to.
- Events are directed to display objects by display runtime
- Display Objects handle events by executing subroutines in their associated scripts.

Examples of Events



Event-Driven Programming

Types of *Events*

- Events that result from operator interaction with the display via mouse clicks, keyboard actions or touch screen.
- Events that relates to process data access, e.g. process changes, alarm monitoring etc.

Notes

- Kinds of events that a display object (in Display Builder) can handle are defined by Honeywell.

  •User events result from operator interaction (mouse clicks, etc.)

  •System events would be a process value changing or the periodic update of the display

Script Access Display Element by Object Paradigm

- Properties of objects are available to scripts by the use of object paradigm.

Example

Objects are characterized by their properties. To change the characteristics of an object you must use the object.property syntax. For example, text is a property of a text object ; this text object has been renamed and is now called DisplayText. If the user wants to set the text property of the text object by a click event then the script will be:

```
Sub LButtonClick()

    DisplayText.text = "REACTOR ONE"

End Sub
```

Script Access Process Data by Object Paradigm

- All of Honeywell programming tools use the syntax:

  `TAGNAME.PARAMETER`

- In the Display Builder the syntax is in a similar format:

  `LCN.TAGNAME.PARAMETER`

Notes

Event-Driven Programming

Common Types of Events

- OnDataChange
- OnPeriodicUpdate
- OnLButtonClick (User Events)
- OnDisplayStartup
- OnDisplayShutdown

Notes

OnDataChange Events

- This subroutine will execute when a change is detected in the value of one or more of the data items it references.
- LCN and DDB reads are from cache (stored scanned values). All writes are immediate to the LCN and DDB.

Notes

OnDataChange Events

Guidelines for Usage

- Do not solicit user input.
- Avoid writes to the LCN and DDB.
- Use Basic Script variable (private or public) to display parameters instead of LCN parameters or DDB as often as possible.
- Do not use a public or private variable as an index value in the script. A change to the variable will not trigger the event.
- Avoid repetitive processing based on a condition. Avoid using the "Do Loop" or the "While Wend"
- Do not write to display parameters.

Notes

OnPeriodicUpdate Events

- This subroutine will execute periodically in half second steps.
- Reads are either cached or immediate. All writes are immediate to the LCN and DDB.

Notes

OnPeriodicUpdate Events

Guidelines for Usage

- Do not solicit user input.
- Avoid writes to the LCN and DDB. Use Basic Script variable (private or public or display parameter) whenever possible.
- Minimize reads of LCN and DDB Data.
- Do not use a public or private variable as an index value in the script. There will be a delay in showing the correct value.
- Avoid using the “Do Loop” or the “While Wend” statement.
- Use the subroutine to do simple animation only.
- Do not use the sleep statement.

Notes

User Events

- This subroutine will execute when a user initiates an event (like a mouse click) when the cursor is over the object.

Notes

User Events

Guidelines for Usage

- Do not use sleep statement.
- Use Basic Script variable (private or public) as much as possible.
- “For Next” “For Each”, “Do Loop”, “While Wend” statements may take time. Use an indicator for long process so the user knows that the statements are being processed.

Notes

User Events Events

Guidelines for Usage (cont.)

- Using a dialog box for user inputs stops all user event processing for that display.(This is because dialog boxes are modal).Instead use the following:
  1. Data Entry Box.
  2. Embedded Display.
  3. Invoking another window (using Safeview).

Notes

OnDisplayStartup Events

- This subroutine will execute when the display starts.

Notes

OnDisplayStartup Events
Guidelines for Usage

- Do not duplicate code that appears in OnDataChange scripts.
- Use Basic Script variable (private or public or display parameter) as much as possible.
- Minimize reads and writes to the LCN and DDB. Limit data access to initializing display DDB items.
- Do not solicit user input. All OnDisplayStartup processing will be stopped until the dialog box is closed.

Notes

OnDisplayStartup Events
Guidelines for Usage (cont.)

- Avoid repetitive processing based on a condition. Avoid using the “Do Loop” or the “While Wend”.
- Do not use the sleep statement.
- Do not use modal dialog boxes.

Notes

OnDisplayShutDown Events

- This subroutine will execute when the display is about to be closed

Notes

OnDisplayShutDown Events
Guidelines for Usage

- Be aware that a hidden dialog box that has not been closed will prevent the shutdown scripts from executing. Do not use modal dialog boxes for this event.

Notes

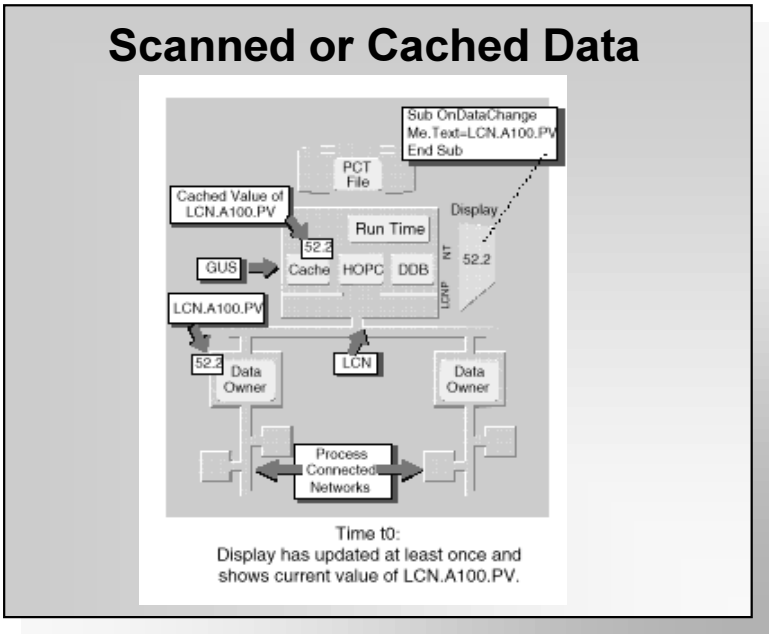
How Data is Accessed By GUS Displays

There are Three ways in which data is read from or written to the LCN.

- Scanned or Cached
- Immediate read
- Immediate write

Notes

Data Access



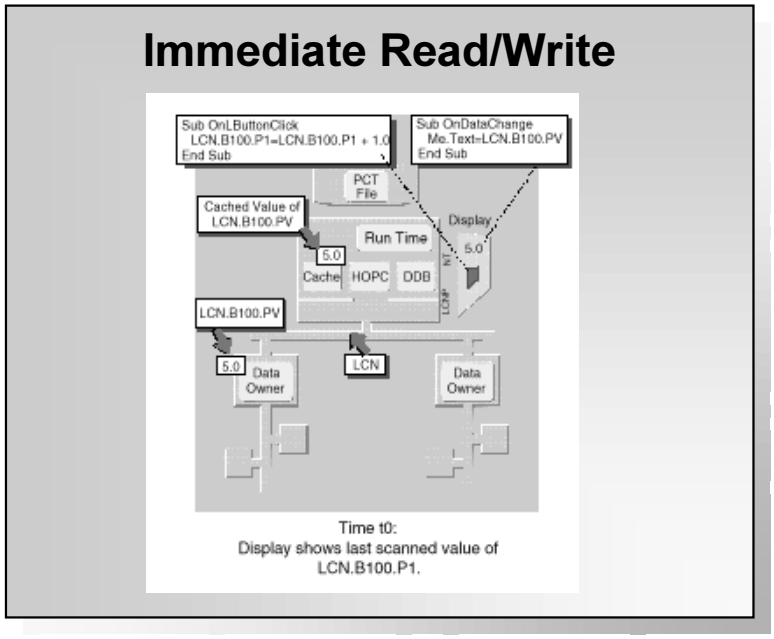
Scanned or Cached Data

Using the **H**oneywell **O**LE for **P**rocess **C**ontrol (HOPC) mechanism, scanned data is read from the LCN and stored in the GUS resident cache memory.

Gus runtime then reads these values, when required.

At each scanned cycle, the newly read value is compared to the previously read value and a DataChange event is generated if the values are different.

Data Access

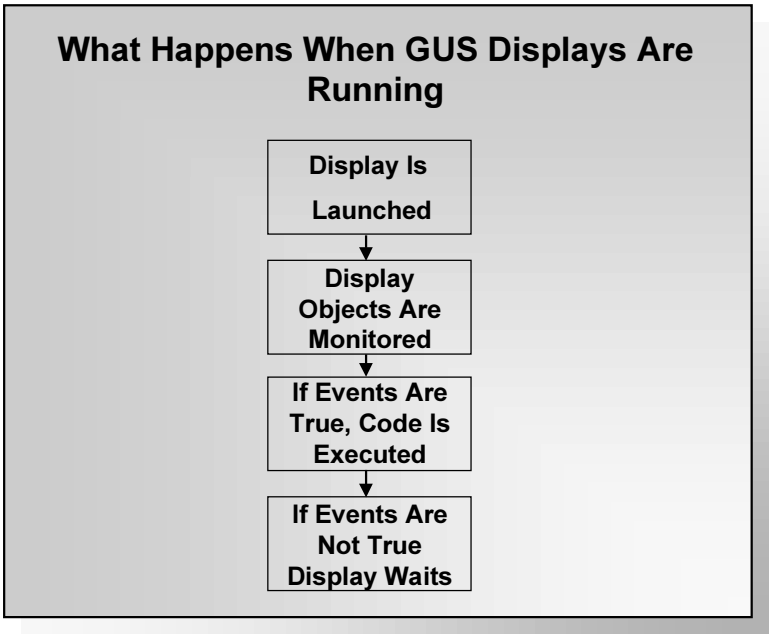


Immediate Read/write

Data referenced by subroutines that are executed because of a user event will read and write to/from the LCN immediately.

These immediate reads/writes do not affect the value in the cache memory. The HOPC server updates cache memory according to scan rates defined in the collection tables.

Data Access



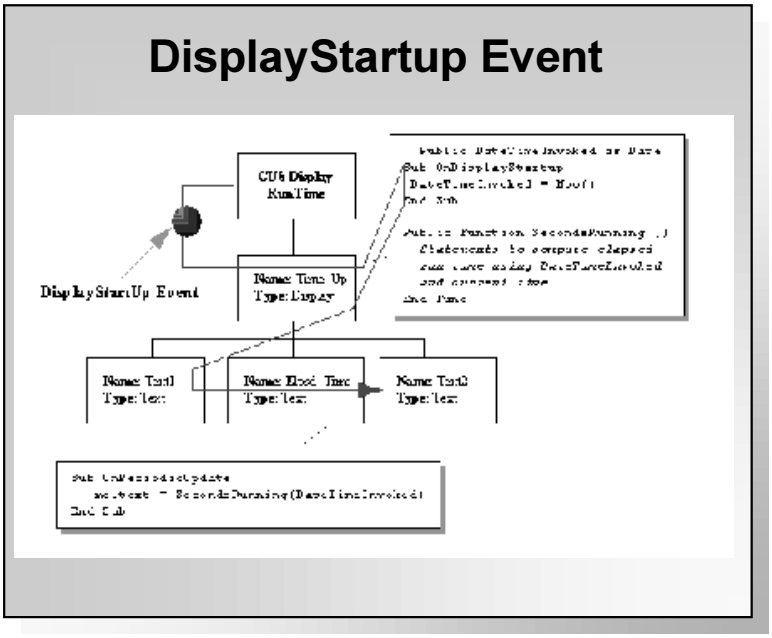
What Happens When a GUS Display Runs

Conceptually, events propagates through the objects in the display looking for scripts to service.

The rules that governed this are:

1. User events propagate from container objects to container objects.
2. When the system event finds a subroutine to handle it, the subroutine is executed and the event continues until all the objects have been visited.

Data Access



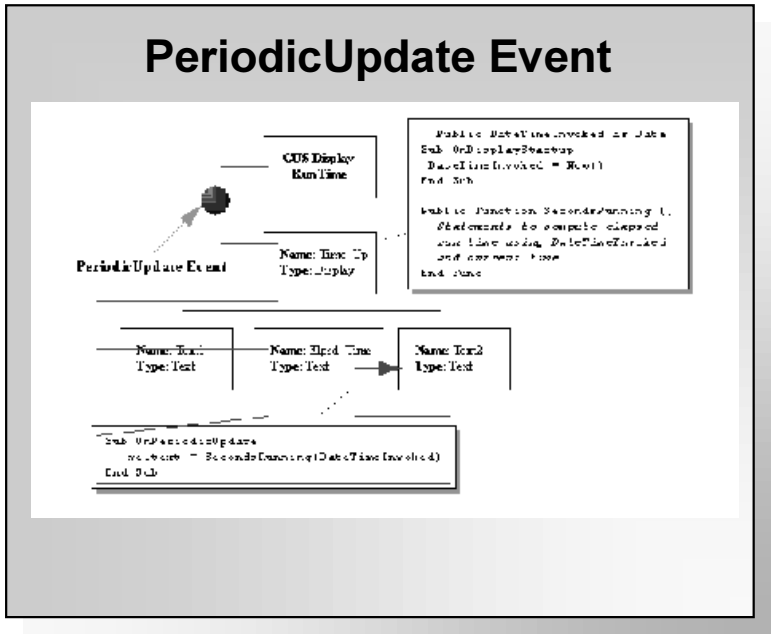
DisplayStartup Event

When a display is invoked, the GUS runtime (GPB.EXE) generates a DisplayStartup event. Scripts associated with this event will be executed.

What to use this event for ?

1. Initialize the display.
2. Initialize an embedded display.
3. Initialize an object.

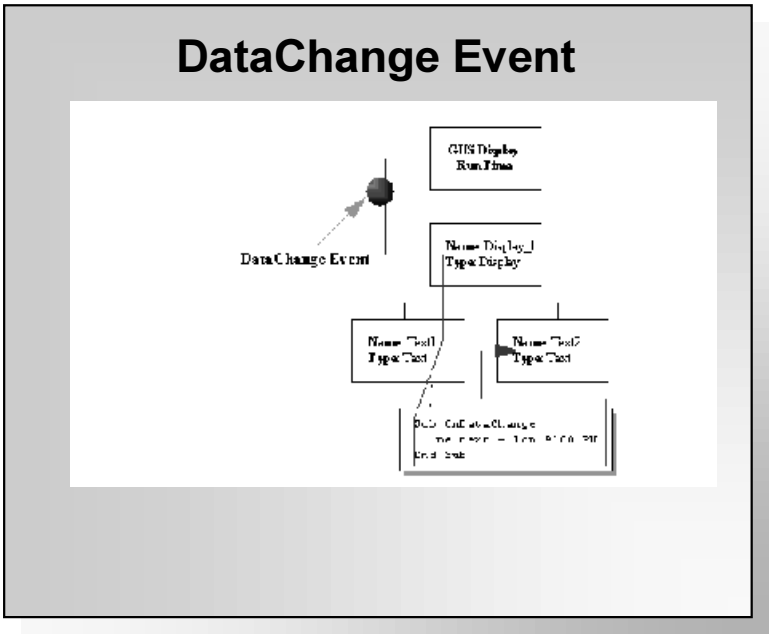
Data Access



PeriodicUpdate

GUS Runtime generates a PerodicUpdate event every half second.

Data Access

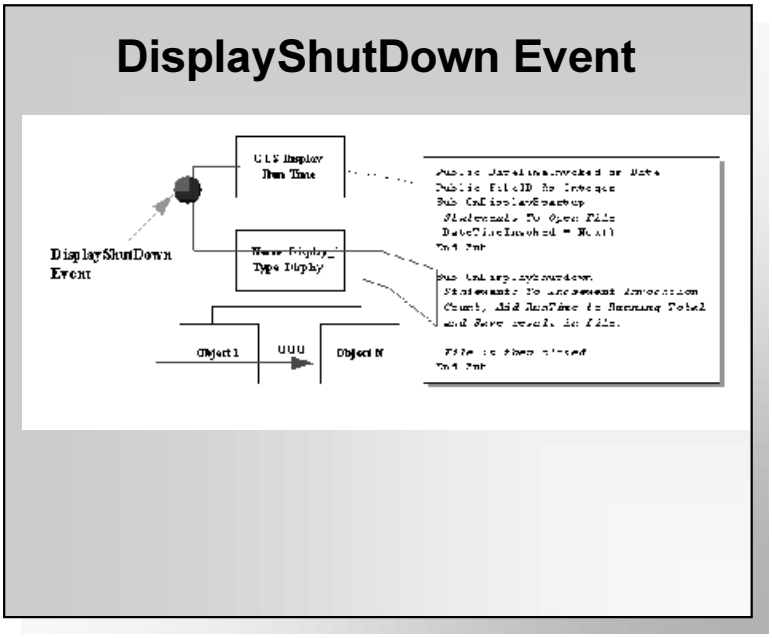


DataChange Event

A reference to an LCN point.parameters will cause the GUS Runtime to detect a change in value and send an OnDataChange event to the display.

This will follow the OnDisplayStartUp event at display invoke time.

Data Access



DisplayShutDown Event

Closing a display will cause GUS Runtime to send an OnDisplayShutDown event to the display.

What to use this event for ?

Use to perform any cleanup required before display is closed.

Order of Subroutine Execution

System Events are executed in the following order:

1. All DisplayStartUp subroutines for a container will execute before member objects.
2. All DisplayShutDown subroutines for member objects will be executed before the DisplayShutDown subroutines for the container.
3. Embedded displays Startup routines will be executed first.
4. Embedded displays Shutdown routines will be executed first.

Data Access

Indirect Access

- A Display can reference the parameters of LCN points by "going through" another variable that contains the name of that variable.
- The variable must be of type "ENTITY"

Indexed Access

LCN points can have arrays of values.
Each array is accessed by an index.

Example

```
Sub OnDataChange
  Me.Text = LCN.FIC200.Level(1)
End Sub
```

External and Internal Names

LCN data points have two names :

- External name - used by the human interface and is represented by a string (“TIC201”)
- Internal name - used by the system to identify points and is represented by a binary number.

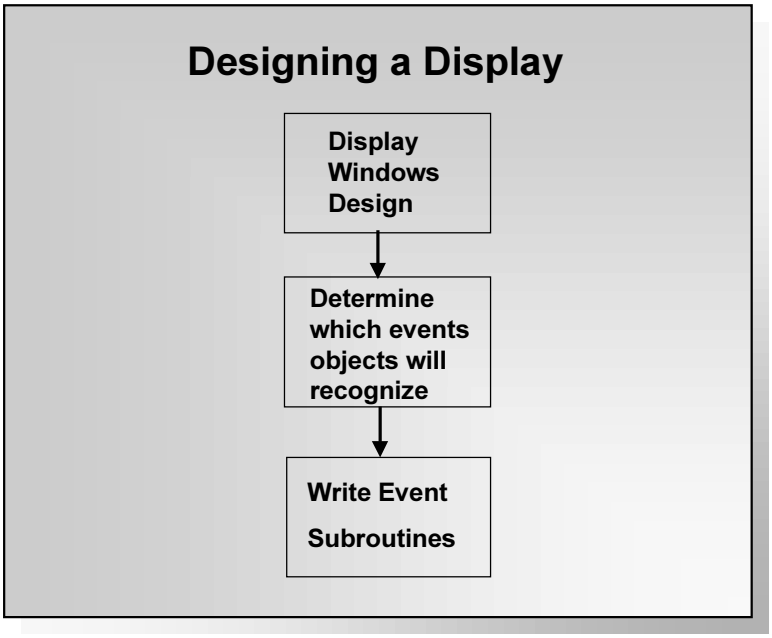
On Creating Code

1. Writing Code Is Easy.
2. Writing Good Code Is Hard.
3. So…. What Is Good Code ?

   – ***Think of The Following Attributes***:
   - The Code Works (No Bugs).
   - The Code Is Documented.
   - The Code Is Maintainable.
   - The Code Runs Quickly.

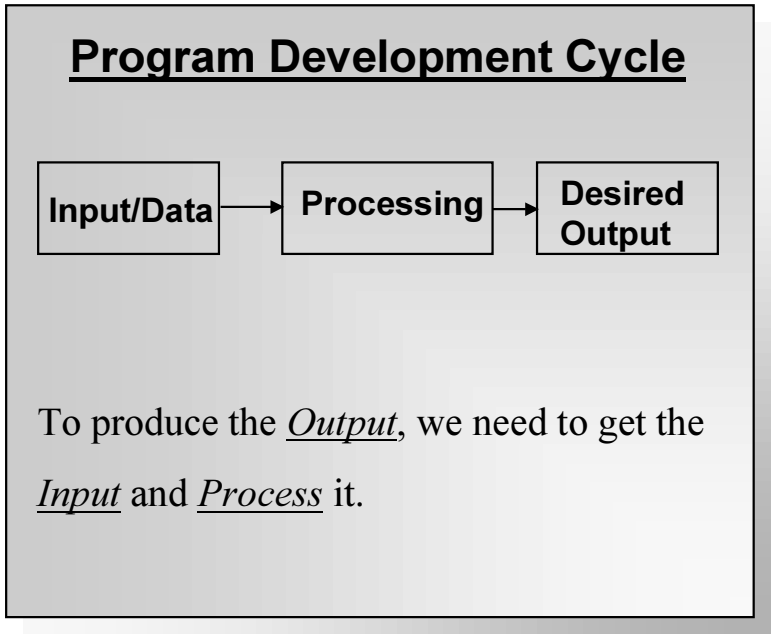
On Writing Code

There are many ways to skin a cat. The guidelines that we will discuss are just that -- GUIDELINES. You may not agree with some of them, so don't follow them; however we have found that these guidelines are useful in the job of writing good code.



Designing a Display

1. Decide what the graphics (window) that the user sees will look like.
2. Determine which events the objects on the graphics (window) should recognize.
3. Write the code for the events (subroutines).

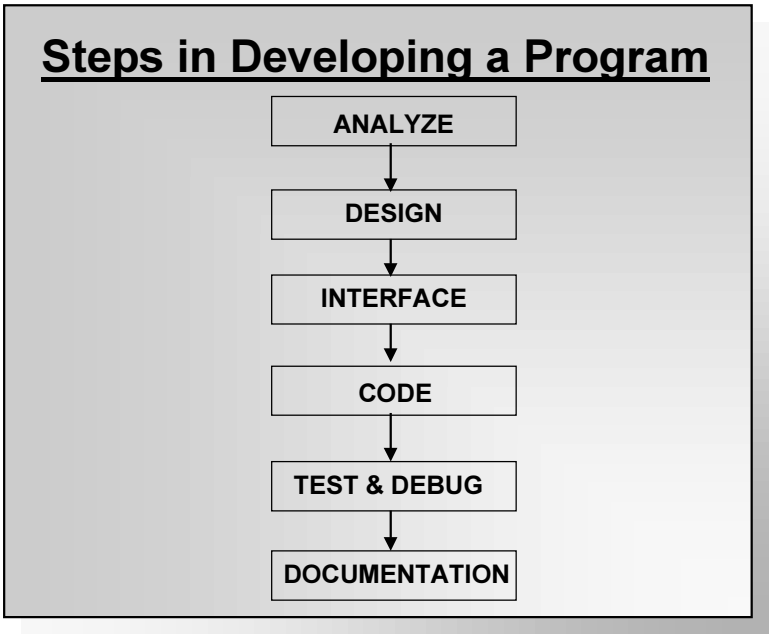


Program Development Cycle

The first step in writing instructions is to determine what the output should be.

The second step is to identify the data or input.

The last step is to determine how to process the input to obtain the desired output.



Steps in Developing a Program

1. **ANALYZE:** Define the problem.

   Know what the program should do, i.e. what is the output ? Have a clear idea of the relationship between the data (input) and the desired output.

2. **DESIGN:** Plan the solution to the problem.

   Plan and create an ALGORITHM, which is a logical sequence of precise steps that solves the problem.

3. **INTERFACE:** Select the objects (Text boxes, buttons, dialog boxes, list boxes etc.)

   Determine how the input will be obtained and how the output will be displayed. Create objects to receive the input and display the output. Have buttons to allow user to control the program.

4. **CODE:** Translate the algorithm into programming language.

   During this stage the program is written in Visual Basic (which is the language for the Display Builder) and entered into the computer.

5. **TEST & DEBUG:** Locate and remove any bugs (errors) from the program.

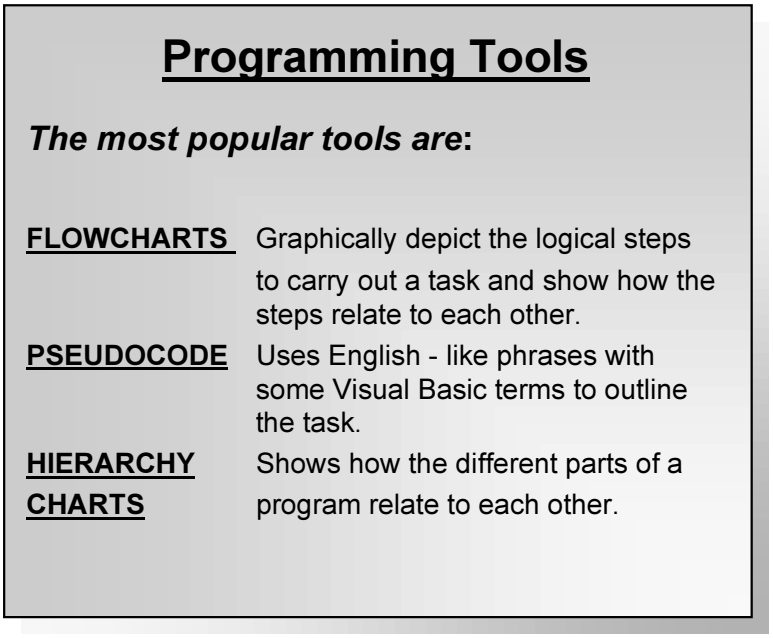
   Other types of errors can only be detected when the program is executed.

6. **DOCUMENTATION:** Organizes all materials that describes the program.

   Why Document ?

   1. Allows another person/programmer to understand the program at a later date.
   2. Detailed description of what the program does and how to use the program - instruction manual.

   This should take place as the code is developed.



Programming Tools

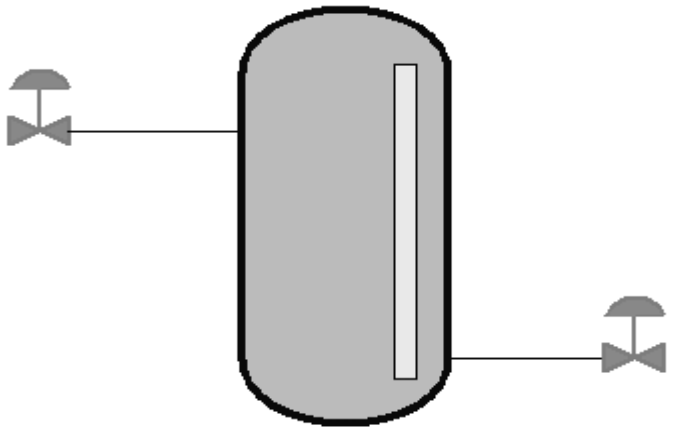
Many programmers may not use these tools. However, you should at least plan your code to enable you to arrive at working logical steps to solve the task.

Example

Task: Create a tank with a level indicator. High and low alarms must be shown using recommended scheme :

- color changes\blinking objects

Example



Example Solution

1. ANALYZE:

What is the output ?

- Monitor the level in the tank
- Alert the operator of low and high alarm
- Know the alarm limits.

2. DESIGN:

Steps in monitoring tank (Algorithm)

- Display PV for level controller.
- Generate alarm when limits have been exceeded.
- Agree on type of alarm.

3. INTERFACE

Select The objects (shape of tank, level indicator)

Example Solution cont.

4. CODE:

Translate the Algorithm to Display Builder script.

5. TEST & DEBUG:

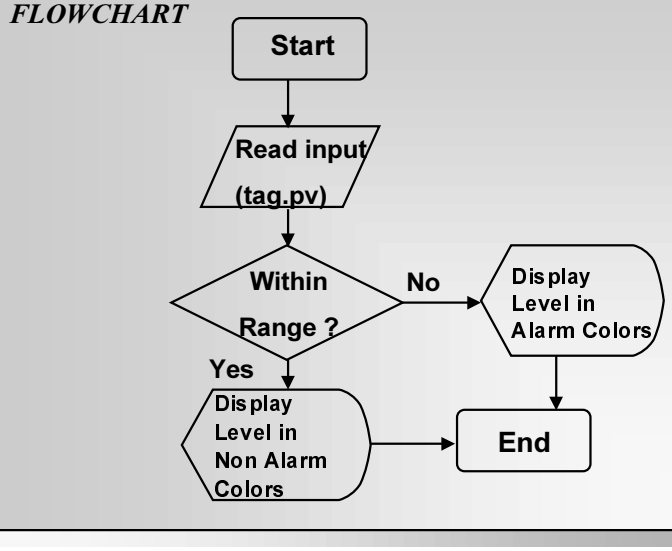
Run and test program by manually generating alarms.

6. DOCUMENT:

Document the program as you create the code.

Example Solution cont.

FLOWCHART



Example Solution cont.

CODE: (This script can be on the object that will display the color change)

```
Sub OnDataChange()
  Select Case LCN.LI24741.PV   'Read the LCN Value
  Case 0 to 10       'Selecting the low alarm range
   Me.fillcolor = Tdc_yellow
  Case 11 to 500     'Selecting the normal range
   Me.fillcolor = Tdc_green
  Case 501 to 532   'Selecting the high alarm range
   Me.fillcolor = Tdc_red
End Select
End Sub
```

Reference

Display Authoring Tutorial

Guidelines for Performant Displays Section 1 page 11-34