

**Chương 6****NGÔN NGỮ LẬP TRÌNH LabVIEW****6.1 Giới thiệu về LabVIEW**

LabVIEW (Laboratory Virtual Instrument Electronic Workbench) của hãng National Instruments, là một ngôn ngữ lập trình bằng đồ họa sử dụng các biểu tượng (icons) thay cho các dòng lệnh để tạo ra các ứng dụng. LabVIEW sử dụng lập trình dòng dữ liệu - dữ liệu xác định sự thực hiện của chương trình. Chính bởi hình thức này mà LabVIEW đem đến một cách nhìn trực quan nhất về phương pháp phân tích, xử lý và thiết kế hệ thống và đặc biệt hữu ích trong thu thập cũng như xử lý tín hiệu.

LabVIEW có khả năng thiết lập mạng truyền thông chuyên dụng cho việc nhận và truyền dữ liệu giữa các máy tính trong công nghiệp, được tích hợp đầy đủ cho việc truyền tin với các phần cứng như là GPIB, VXI, PXI, RS-232, RS-485, và các thiết bị thu nhận dữ liệu... Vì vậy mà LabVIEW có các đặc trưng được xây dựng bên trong phù hợp với việc kết nối các ứng dụng.

LabVIEW có các thư viện đầy đủ dùng cho thu thập, phân tích hiển thị và lưu trữ dữ liệu. LabVIEW cũng có các công cụ phát triển phần mềm truyền thống. Người sử dụng có thể tạo các điểm dừng, chạy mô phỏng chương trình và chạy từng bước cả chương trình để đơn giản hoá việc gỡ rối và viết chương trình.

**6.1.1 Ngôn ngữ lập trình G (Graphic)**

Ngôn ngữ lập trình G là một ngôn ngữ đồ họa và được sử dụng cho phần mô tả các kết nối và mối quan hệ giữa các đối tượng trong biểu đồ khối và đây là nền tảng của LabVIEW, ngôn ngữ G khác với ngôn ngữ lập trình dạng text theo nhiều cách.

Trước hết, ngôn ngữ dạng text được thực thi một cách tuyến tính, nghĩa là từng lệnh một. Lệnh kế tiếp sẽ được thực thi ngay sau khi lệnh trước được hoàn thành, nghĩa là chỉ có một lệnh duy nhất được thực thi ở một thời điểm. Ngôn ngữ G có những đặc tính khác biệt. Một khối đồ họa được thực thi chỉ khi các input của nó được định nghĩa hoặc được xác định. Điều này nghĩa là nó có khả năng tạo ra một biểu đồ có các nhánh song song, vì vậy ta nhận thấy sự thực thi các hoạt động cùng một lúc.

Ngôn ngữ G hoàn toàn là một ngôn ngữ lập trình và cung cấp nhiều tính năng thông dụng chẳng hạn như vòng lặp For, While, Case structure và một nét mới mà nó không phải đặc trưng cho ngôn ngữ dạng text được gọi là Sequence structure, nó cho phép thực thi tuyến tính những phần nào đó của chương trình.

**6.1.2 Sự khác biệt giữa LabVIEW và ngôn ngữ lập trình dạng text**

- Ngôn ngữ dạng text (C, pascal, Matlab. .) là sử dụng những dạng lệnh dạng text, của LabVIEW là lập trình dạng đồ họa.

- Lập trình dạng tuần tự (C, pascal, Matlab...) còn LabVIEW lập trình dạng song song.
- Dây dẫn biểu diễn cho dãy ước lượng
- Ngôn ngữ dạng text sử dụng những dòng lệnh để xác định lệnh thực thi chương trình, còn LabVIEW thì sử dụng dạng dữ liệu để xác định lệnh thực thi.

### 6. 1. 3 Những điểm giống nhau giữa LabVIEW và ngôn ngữ lập trình dạng text

- Algorithm → code
- Biên dịch (trong LabVIEW “được ẩn”)
- Input → algorithm → output (LabVIEW: input/output có thể can thiệp thông qua font panel và algorithm có thể can thiệp thông qua diagram)

### 6. 1. 4 Thuận lợi và bất lợi

#### \* Thuận lợi :

- Xây dựng các dụng cụ điều khiển và hiện thị giống như thật, nhanh chóng, dễ dàng
- Lập trình khá dễ dàng cho những công việc rất phức tạp chẳng hạn như sự tự động hóa của những thí nghiệm phức tạp
- Tất cả khả năng của LabVIEW đều “phơi bày” trên desktop
- Một thuận lợi quan trọng khác là khả năng thực thi những script mà được gửi từ xa từ bất kỳ ứng dụng có quyền truy suất vào **NI DataSocket server** (giao thức TCP/IP có thể sử dụng tốt)

#### \* Bất lợi :

- Bạn đã quen với lập trình bằng câu lệnh ở các ngôn ngữ khác rồi
- Bạn có thể dễ bị rối trong mớ biểu đồ (Nếu bạn ko đủ kiên nhẫn hay công việc giải quyết được xây dựng sai lúc đầu)

### 6. 1. 5 Ứng dụng của LabVIEW

Phần mềm LabVIEW đang ngày càng phát triển và trở nên chuyên dụng cho các hệ thống đo lường. Phần mềm này giúp cho việc thiết kế các thiết bị đo ảo cho các phòng thí nghiệm dựa trên cơ sở là các thiết bị thực tế, người sử dụng hoàn toàn có thể thao tác dễ dàng trên các thiết bị ảo này giống như đối với các thiết bị thực tế. LabVIEW có khả năng thích ứng với các phần cứng và các phần mềm khác nhau nên nó rất hữu ích trong việc thu thập phân tích và xử lý số liệu nên rất tiện dụng cho các phòng thí nghiệm. Với các ứng dụng hữu ích, LabVIEW đã xâm nhập vào hầu hết các lĩnh vực từ những công việc chỉ đơn giản là mô phỏng đến những công việc phức tạp như:

- Trong tự động hoá: dẫn đường cho pilot của máy công cụ để tạo các chi tiết, kiểm tra nhiệt độ của motor trong dây chuyền lắp ráp.
- Trong quân sự: kiểm tra khả năng truyền và nhận của ăngten radar.

- Trong ngành sản xuất gang thép: cải thiện khả năng tự động hoá trong việc tạo ra các thành phẩm, chi tiết cơ khí.
- Trong ngành năng lượng: thu thập và điều khiển quá trình nạp năng lượng.
- Trong ngành vũ trụ: điều khiển và thu thập dữ liệu của thiết bị viễn thám.
- Trong viễn thông: sử dụng các thiết bị ảo qua mạng để thu thập các số liệu tức thời tại một nơi khác.
- Trong thông tin vô tuyến: tự động kiểm tra các thiết bị truyền phát.
- Trong y học: thu thập và xử lý, phân tích độ rung và các chuyển động bất thường.

Theo tài liệu Lview online, thì LabVIEW có ứng dụng cụ thể trong công nghiệp như:

- Kiểm tra, đo kiểm và phân tích tín hiệu trong kỹ thuật (Đo nhiệt độ, phân tích nhiệt độ trong ngày)
- Thu thập dữ liệu (Data Acquisition), (Thu thập các giá trị áp suất, cường độ dòng điện ...)
- Điều khiển các thiết bị (Điều khiển động cơ DC) .
- Phân loại sản phẩm (Dùng chương trình xử lý ảnh để phân biệt sản phẩm bị lỗi, phế phẩm...)
- Báo cáo trong công nghiệp (Thu thập, phân tích dữ liệu và báo cáo cho người quản lý ở rất xa thông qua giao thức truyền TCP/IP trong môi trường mạng thernet
- Giao tiếp máy tính và truyền dẫn dữ liệu qua các cổng giao tiếp (Hỗ trợ hầu hết các chuẩn giao tiếp như USB, PCI, COM, RS-232, RS-485).

## 6. 2. Lập trình LabVIEW căn bản

### 6.2.1 Cách thức hoạt động của LabVIEW

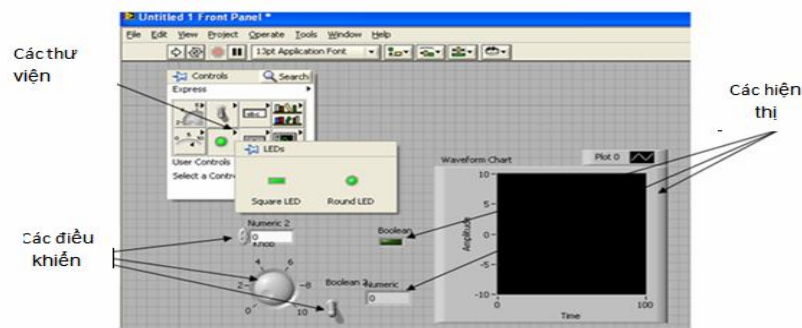
Những chương trình LabVIEW thiết kế dụng cụ, thiết bị, bộ thí nghiệm được gọi là những thiết bị ảo (Virtual Instruments – VIs), bởi vì hình dạng và cách hoạt động giống với những thiết bị vật lý, chẳng hạn như các bộ thí nghiệm điều khiển nhiệt độ, điều khiển mực chất lỏng, máy nghiệm dao động, máy hiện sóng. Mỗi VI sử dụng những hàm mà tín hiệu đầu vào từ giao diện người dùng hoặc từ những nguồn khác và hiển thị ra thông tin đó hoặc di chuyển tới những file khác hoặc computer khác.

### 6.2.2 Giới thiệu dụng cụ ảo

Chương trình labview được gọi là dụng cụ ảo hay là các VI. Một VI chứa 3 thành phần sau:

- Front panel – là giao diện người sử dụng
- Block diagram – chứa đoạn mã đồ họa mà nó định nghĩa chức năng của VI đó.
- Icon và connector panel – Chỉ định một VI mà được sử dụng trong một VI khác. (kết nối và biểu tượng để làm chương trình con). Một VI được sử dụng trong VI khác thường gọi

là subVI. Một subVI tương ứng với chương trình con (subroutine) trong ngôn ngữ lập trình dạng text.



**Hình 6.1 Các dụng cụ điều khiển và hiện thị**

### 6. 2. 3 Front Panel

Font Panel một giao diện với người. Bạn xây dựng font panel bằng cách sử dụng những bộ điều chỉnh dữ liệu (Controls) và dụng cụ hiện thị dữ liệu đã được xử lý (Indicators)

- Control là các đối tượng được đặt trên Front Panel để cung cấp dữ liệu cho các khối trong chương trình. Nó tương tự như đầu vào cung cấp dữ liệu.
- Indicator là đối tượng được đặt trên Front Panel dùng để hiện thị kết quả, nó tương tự như bộ phận đầu ra của chương trình.

Những bộ điều khiển và dụng cụ hiện thị này được đặt ở trong **control palette** của labview.

Các bộ điều khiển là những nút xoay (knobs), nút nhấn (push buttons), bộ phím bấm (dials) và các cơ cấu input khác.

Dụng cụ hiển thị là những ô text, đồ thị, biểu đồ, LEDs, . . vv.

### 6. 2. 4 Block Diagram

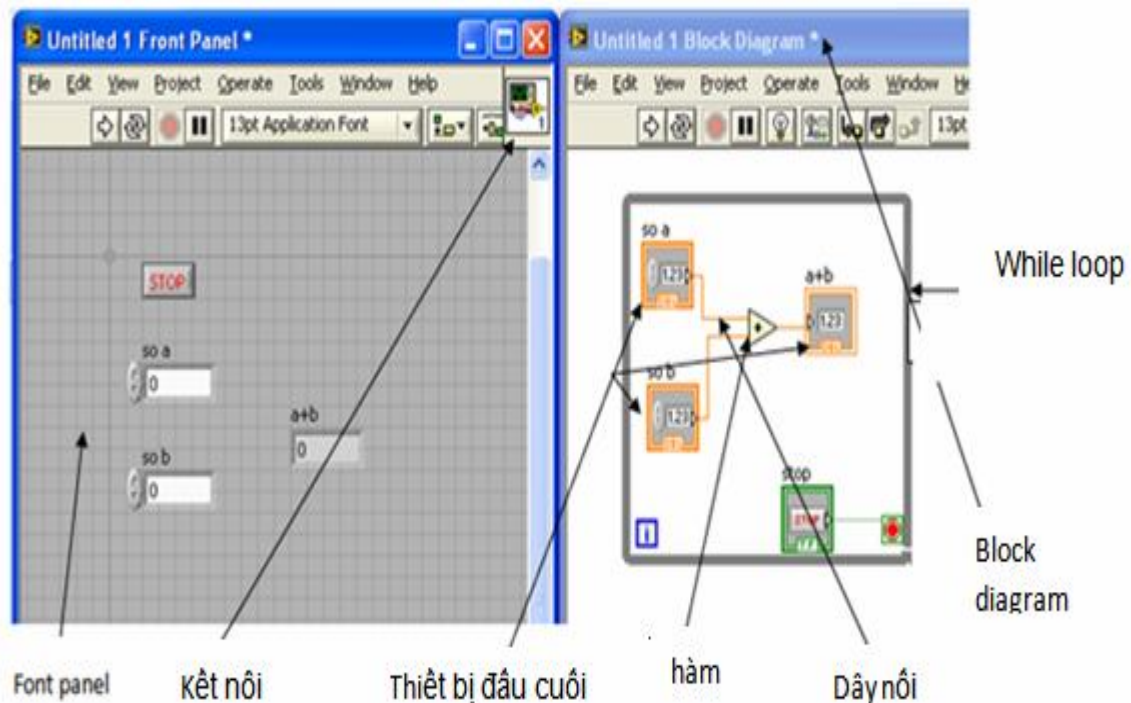
Block diagram chứa mã source đồ họa, được biết như là mã G hoặc mã block diagram, giúp cho VI chạy. Block diagram sử dụng các biểu tượng hàm đồ họa để kết nối các điều khiển với các hiện thị để thành chương trình phục vụ mục đích của người lập trình (các điều khiển và hiện thị trong block diagram tương ứng với các điều khiển và hiện thị ở trên font panel) khi ta đặt trên font panel một điều khiển hoặc hiển thị thì sẽ xuất hiện trong block diagram một khối điều khiển hoặc hiển thị tương ứng

Trong Block Diagram gồm các thiết bị đầu cuối (Terminal), Nút (Node) và các dây nối (wire)

- Terminal: Là mã biểu tượng cho các điều khiển hay hiện thị trong font panel. Nó chứa dữ liệu của các điều khiển hoặc hiển thị nên nó dùng để thay đổi dữ liệu giữa font panel và block diagram. Khi bạn nhập dữ liệu từ các điều khiển trong font panel nó sẽ được truyền đến thiết bị đầu cuối.

- Nodes: Là các phần tử thực hiện chương trình trong block diagram, nó có đầu vào và đầu ra. thực hiện hoạt động khi VI chạy, chúng tương tự như các lệnh toán tử, hàm và các chương trình con trong các ngôn ngữ lập trình thông thường.
- Wires: Là các dây nối dữ liệu giữa các node với các điều khiển và hiển thị để truyền dữ liệu. mỗi loại dây có một loại dữ liệu riêng. màu dây, loại dây phụ thuộc vào loại dữ liệu của chúng.


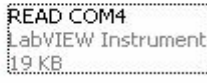
Dòng dữ liệu (data flow) đi trong dây dẫn từ những bộ điều khiển tới các khối lệnh để thực hiện và tới các hiển thị



**Hình 6.2 Giới thiệu về Block Diagram**

### 6. 2. 5 Icon & Connector

Xây dựng Icon và connector để sử dụng trong VI khác

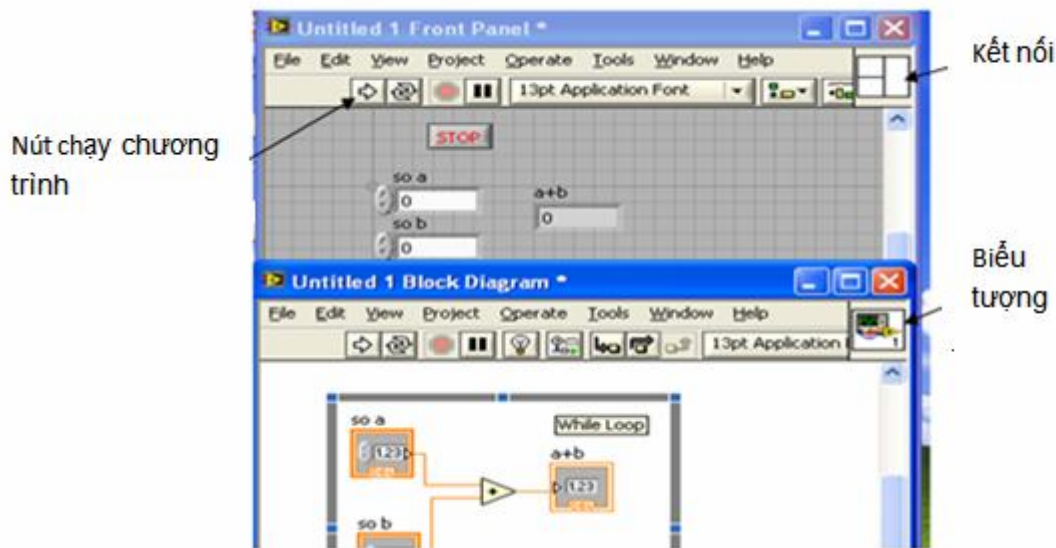
- Icon (biểu tượng)   : là biểu tượng của VI hay của một chương trình, một chương trình có một biểu tượng riêng hoặc theo người lập trình tạo chương trình này có thể được sử dụng làm chương trình con trong chương trình khác. Khi một VI được sử dụng như chương trình con trong một VI khác, khi đó VI đó được gọi là SubVI.

Bạn có thể xây dựng một kết nối như hình dưới đây để sử dụng VI như một sub VI



- Connector (bộ nối) : Là một phần tử của Terminal dùng để nối các đầu vào và đầu ra của các VI. Khi sử dụng một VI làm sub VI thì ta phải qui định những đầu vào và đầu ra. Đầu vào được kết nối với các điều khiển để nhận dữ liệu, đầu ra kết nối với những khối lệnh

hoặc các thiết bị hiện thị khác, Icon và connector đều nằm Trên góc phải của font panel và block diagram



Hình 6.3 Icon và connector

Khi muốn tạo connector và Icon thì ta đưa chuột vào icon mặc định trên góc phải của font panel nhấp phải chuột và chọn edit icon, nhấp chuột phải vào Icon và chọn show connector để chọn kết nối đầu vào và đầu ra. Ta có thể thay đổi được Icon và connector

## 6. 2. 6 Môi trường labview:

**6.2.6.1 Getting tarted window:** Getting tarted xuất hiện khi bạn chạy labview. Sử dụng window này để tạo các VI, mở các file, tìm ví dụ, mở tài liệu để học labview, mở help để đọc những thông tin cơ bản. Getting tarted sẽ mất khi bạn mở một ví dụ hoặc một file hay bạn tạo một VI mới.


- **Các menu:** Các menu nằm ở phía trên của một VI bao gồm các mục phổ biến tới các giao tiếp khác cũng như open, copy, paste, .... vv
- **Thanh công cụ:** Sử dụng các nút nhấn trên thanh công cụ để chạy, dừng VI và gỡ rối chương trình



Hình 6.4 Các menu và các thanh công cụ



Hình 6.5 Menu và thanh công cụ

- **Context help window:** Context help hiển thị thông tin cơ bản về những dự án labview khi bạn di chuyển con trỏ chuột trên mỗi dự án. khi đã đọc thông tin cơ bản trên context help mà bạn muốn tìm hiểu chi tiết hơn thì click vào **detailed help** hoặc click vào biểu tượng hình dấu hỏi như hình dưới đây: 

Để sử dụng context help chọn: **help/show context help**

Sử dụng labview palletet, dụng cụ, menu để xây dựng front panel và block diagram của VIs. labview bao gồm 3 palettes: Control palettes, Function palette, Tools palettes

- Controls palette: Chỉ sử dụng trong front panel, control palette bao gồm những điều khiển và hiển thị, bạn sử dụng để xây dựng front panel. Để truy xuất controls palette ta vào: **view /controls palette**





Hình 6.6 Controls palette

- Functions palette: Chỉ sử dụng trong block diagram, functions palette bao gồm các VI và các hàm lệnh. Bạn sử dụng để xây dựng block diagram. . . nếu bạn biết tên của các hàm mà không biết nó nằm ở đâu thì nhập tên của hàm đó vào tìm kiếm. để truy xuất functions palette: chọn **view/Functions palette**



Hình 6.7 Functions palette

#### 6.2.5.2 Tools Palette (dụng cụ)

Tools palette được dùng trong front panel và block diagram. Các công cụ này ta chọn chúng để thực hiện một chức năng riêng như: nối dây, di chuyển các khối, điều khiển thử, nhập số, ghi nhãn cho các điều khiển. . . v. v

LabVIEW tự động chọn những công cụ cần thiết hiển thị trên Front panel và Block diagram. Các chức năng công cụ có thể lựa chọn ở chế độ đặc biệt theo con trỏ chuột. Sử dụng các công cụ để tạo và hiệu chỉnh các đối tượng trong Front panel và Block diagram.











Để hiển thị Tools palette, vào **View\Tools palette** hoặc nhấn phím **shift +click phải chuột**.  
**Hiện như hình dưới**





Hình 6.8 Tools palette

**Chức năng cụ thể của các công cụ cơ bản:**

-  Operating tool.
-  Positioning tool.
-  Labeling tool.
-  Wiring tool.
-  Object Pop-up Menu tool.
-  Scroll tool.
-  Breakpoint tool.
-  Probe tool.
-  Color copy tool.
-  Color tool.

Hình 6.9 Chức năng tool palette

- Operating tool: Là dụng cụ dùng để điều khiển giá trị của các điều khiển khi chưa chạy chương trình
- Positioning tool: Là dụng cụ để di chuyển các phần tử điều khiển hay để chọn các dụng cụ.
- Labeling tool: Là dụng cụ để ghi nhãn cho các điều khiển hay nhập giá trị của các điều khiển hay các hiển thị.
- Wiring tool: Là dụng cụ dùng để nối dây giữa các phần tử điều khiển hay với các lệnh.
- Scroll tool: Dụng cụ để kéo dự án đi lên, xuống, qua trái, phải.
- Color tool: Dụng cụ để thay đổi màu của các điều khiển và hiển thị.

#### 6.2.5.2 Một số bộ điều khiển và hiển thị trong controls palette

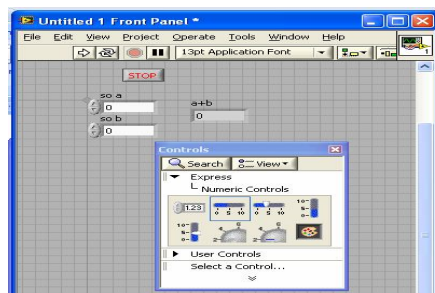
Khi tạo VI mới ta phải lấy các điều khiển hoặc hiển thị trong controls palette

##### + Numeric control:

Vào font panel nhấp phải chuột hoặc chọn trên thanh công cụ (chọn view/control palette. rồi từ control ta tìm đến các thư viện để lấy các khối lệnh

Hình 6.10 ta muốn lấy các điều khiển số, ta vào controls /numeric controls và chọn các điều khiển mình muốn. các phần tử điều khiển số dùng để điều khiển số, thay đổi giá trị điều khiển số bằng những các sau :

- Sử dụng công cụ hoạt động hoặc công cụ nhấn click bên trong ô hiển thị số và thay đổi từ bàn phím.
- Sử dụng công cụ hoạt động click vào nút tăng hoặc giảm của một điều khiển số, để tăng hoặc giảm giá trị.



**Hình 6.10 chọn điều khiển số**

#### + Boolean Controls

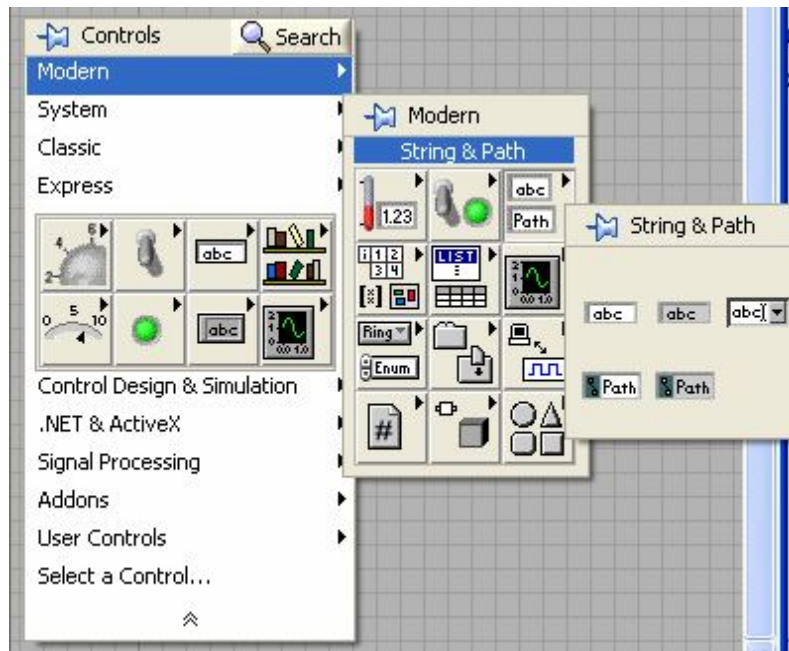
Bộ công cụ này cung cấp 2 giá trị là True và False. Khi thực hiện chương trình người sử dụng sử dụng chuột để điều khiển giá trị của thiết bị. Việc thay đổi giá trị của các thiết bị chỉ có tác dụng khi các thiết bị đó được xác lập ở chế độ là các Control. Còn nếu ở chế độ là các Indicator thì giá trị không thay đổi vì chúng chỉ là các thiết bị hiển thị



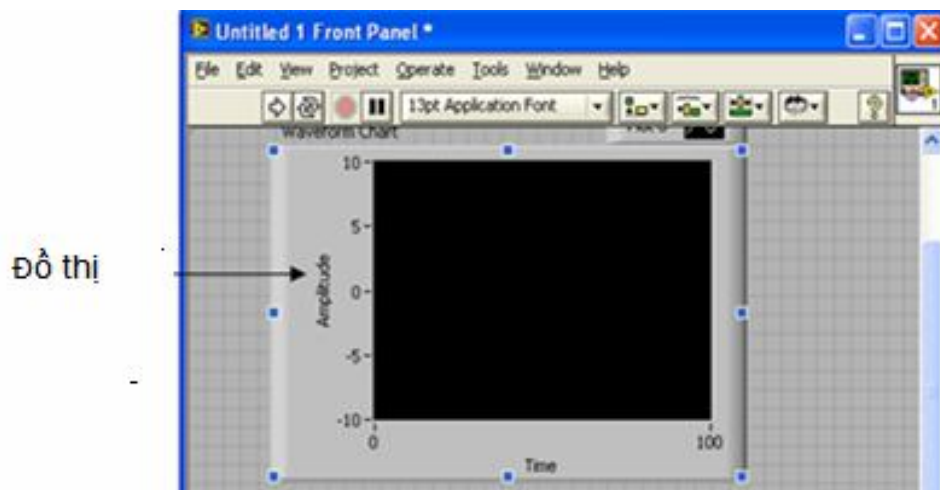
**Hình 6.11 Boolean Controls/Indicators**

#### + String Controls/Indicators:

Các điều khiển này dùng để nhập và hiển thị các ký tự, số. . vv. nó cũng có thể được xác lập ở chế độ đầu vào hay đầu ra muốn sử dụng các phần tử này. Ta vào controls/modern/string&path/string control hoặc string indicator. Như hình 6.12



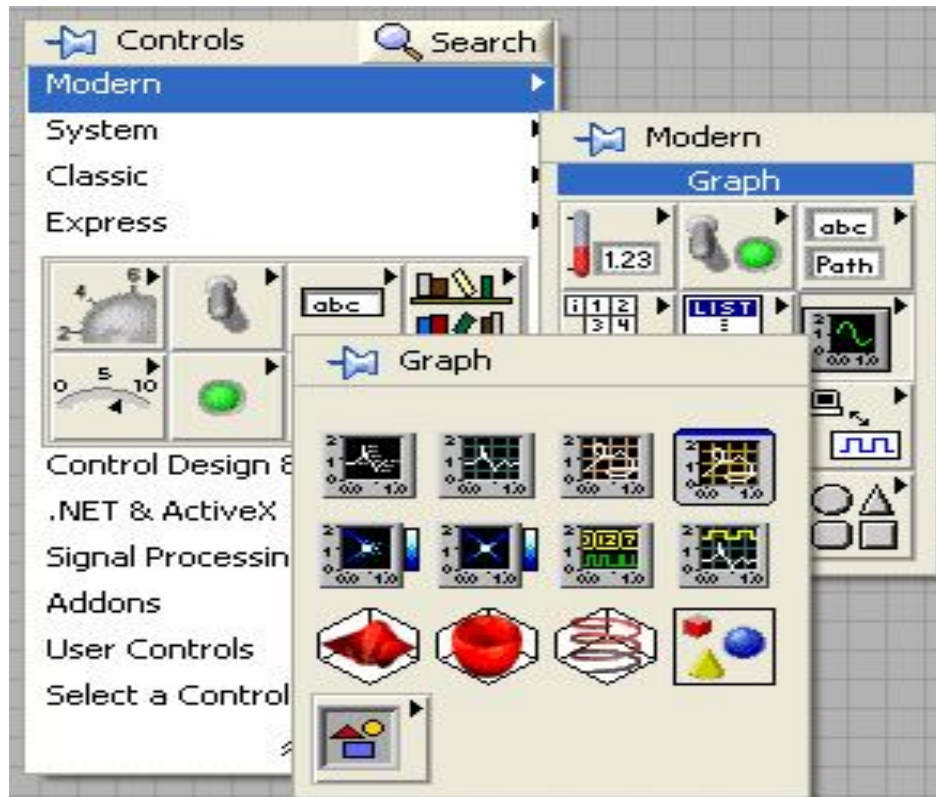
Hình 6.12 Lấy string control, string indicator



Hình 6.13 Đồ thị

+ **Đồ thị, Biểu đồ:** là những hiện thị dùng để biểu diễn dữ liệu số ở dạng biểu đồ hoặc đồ thị

Để sử dụng biểu đồ và đồ thị, từ font panel ta chọn control/modern/graph/chọn loại biểu đồ và đồ thị thích hợp



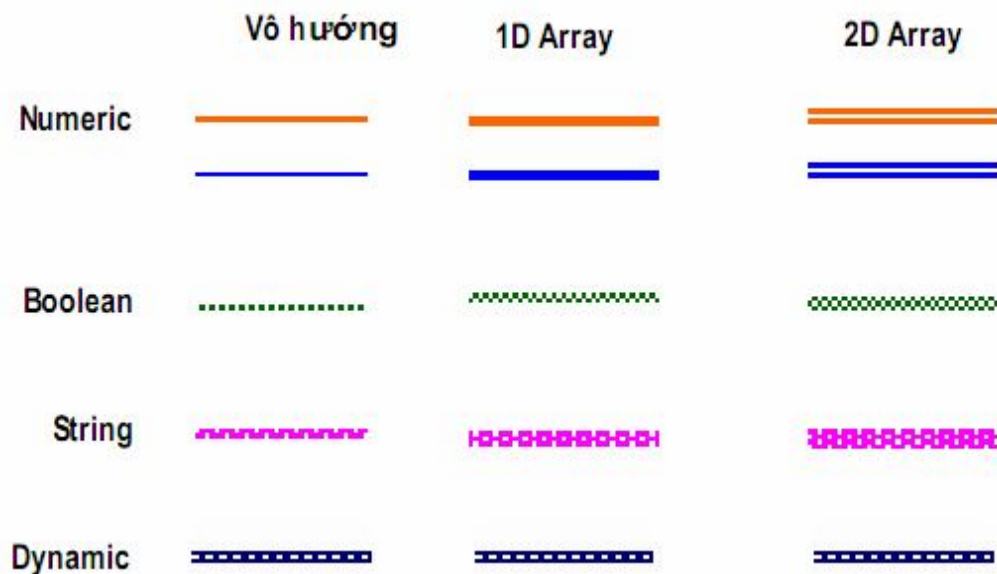
Hình 6.14 Chọn đồ thị và biểu đồ

#### 6.2.5.4 Các kiểu và cấu trúc dữ liệu :

- a) Tất cả các kiểu dữ liệu “tiêu chuẩn” đều có: numeric, Boolean, string
- b) Các cấu trúc dữ liệu “tiêu chuẩn”: array (list, table)
- c) Các cấu trúc dữ liệu “nâng cao” : cluster, chart, ring, ...

#### Các Function

- a) Các kiểu function “tiêu chuẩn” có sẵn: numeric, boolean, string, array, logical, file I/O
- b) Các function “nâng cao”: time, communication, DAQ, instrumentation control, data analysis, ...



**Hình 6.15** Các dạng dây nối trên Block Diagram

Các loại dây nối theo từng kiểu dữ liệu sẽ được tự động cập nhật, khi bạn chọn kiểu dữ liệu, ví dụ như khi bạn chọn một điều khiển mảng thì kiểu dây mảng sẽ được tự động xuất hiện khi bạn nối mảng đó tới các hàm hoặc hiện thị

### 1 Dữ liệu:

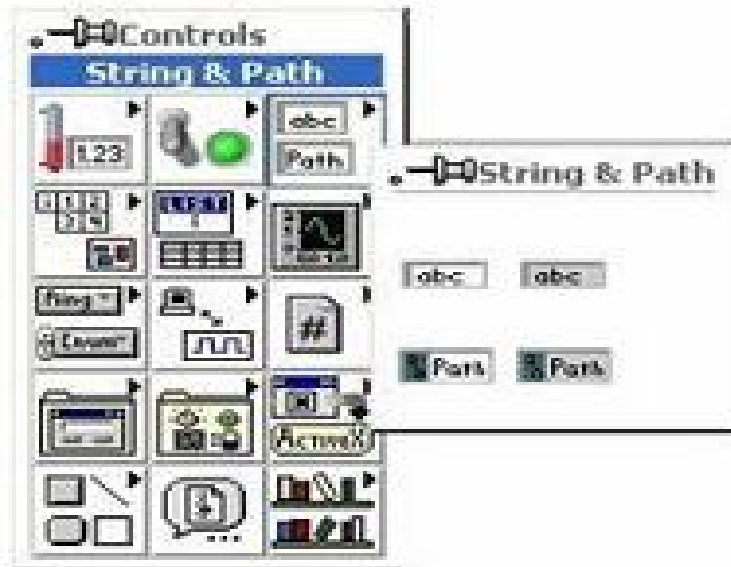
Ngôn ngữ lập trình LabVIEW hỗ trợ cho tất cả các dạng dữ liệu. Các kiểu dữ liệu dạng Boolean, bytes, string, array, file, text, cluster và dạng số có thể được chuyển đổi một cách dễ dàng sang các dạng cấu trúc. Sau đây chúng ta xem xét một vài dạng dữ liệu

#### + Variables (biến)

Trong quá trình lập trình cần thiết phải sử dụng tới các biến. Thông qua các biến, người lập trình có thể thực hiện được việc xử lý và thay đổi dữ liệu một cách thuận lợi. Trong LabVIEW, các biến được sử dụng dưới 2 dạng là biến toàn cục (global variables) và các biến cục bộ (local variables).

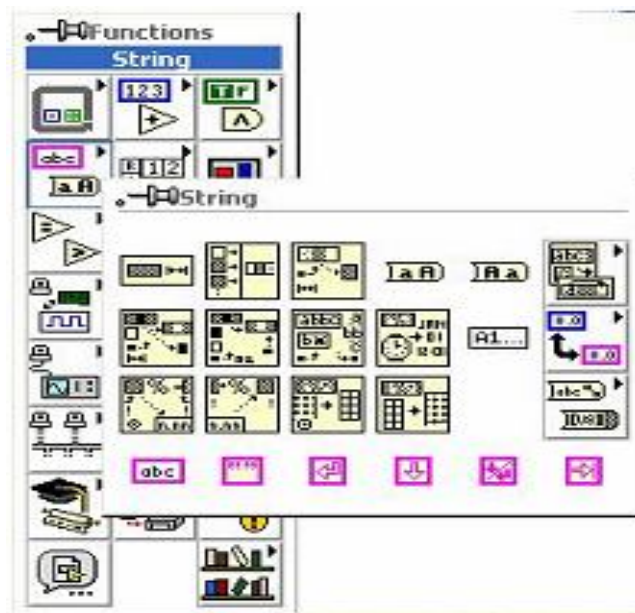
- Global variables (biến toàn cục): Biến toàn cục được sử dụng để thực hiện công việc truyền và lưu trữ dữ liệu giữa một vài VIs. Biến toàn cục được coi là một đối tượng trong LabVIEW.
- Local variable: Cho phép người sử dụng đọc hoặc viết thông tin lên 1 trong những thiết bị điều khiển hoặc thiết bị hiển thị trên Front Panel. Để tạo một biến cục bộ, chọn Local Variable từ bảng Structs & Constants.

Kiểu string (chuỗi) là một tập hợp các ký tự. Ta có thể sử dụng chuỗi tham gia tính toán xử lý. Để lựa chọn các ô text lưu trữ dữ liệu kiểu string ta chọn từ ô “String & path” từ control palette có hình như sau (hình 2. 19)



Hình 6.16 String

LabVIEW cung cấp cho người sử dụng các công cụ để thao tác trên LabVIEW như việc tìm kiếm, xác định độ dài của chuỗi ký tự, chuyển dãy ký tự thành dạng viết hoa và ngược lại, so sánh chuỗi, thay một số ký tự trong chuỗi này sang một số chuỗi khác. các hàm được cung cấp trong menu “string” trên Function Palette có hình 2. 20



Hình 6.17 Function / String

Không chỉ vậy có thể thực hiện việc chuyển đổi từ dạng string sang các dạng khác như dạng số, dạng mảng (array). Trong LabVIEW, đôi khi người sử dụng có thể cần phải tạo ra hoặc hiển thị một số ký tự trong bảng mã ASCII mà không thể hiển thị được vd: Esc, tab.... Để giải quyết vấn đề đó, LabVIEW đã cung cấp một số từ đại diện cho các từ đó, khi cần thể hiện các ký tự đó ta chỉ cần gõ vào những từ đại diện cho chúng trên Front Panel. Sau đây là một số từ thay thế (Escape Code)



**Bảng 6. 1 - Escape Code :**

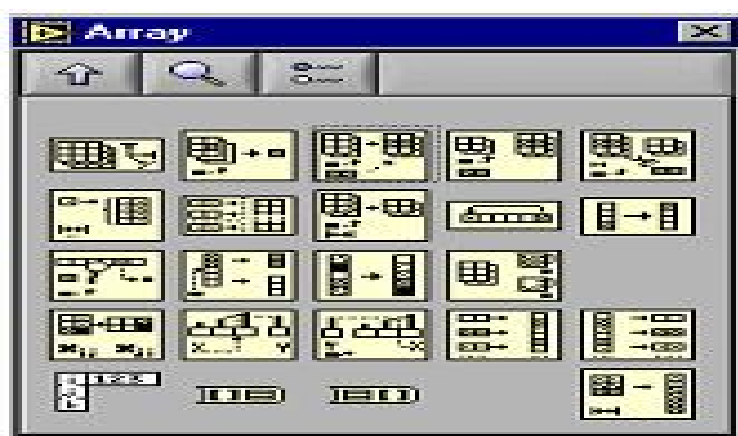
| Escape code | Từ được thay (Interpreted As)    |
|-------------|----------------------------------|
| \00-\xFF    | Giá trị dạng Hex của ký tự 8 bit |
| \b          | Backspace                        |
| \f          | Formfeed                         |
| \n          | Newline                          |
| \r          | Return                           |
| \t          | Tab                              |
| \s          | Space                            |
| \\          | BackSlash (dấu chéo ngược)       |

### + Array

Trong việc lập trình đôi khi chúng ta cần xử lý một số các dãy số mà mỗi phần tử trong đó có thể được xử lý như từng thành phần riêng biệt, vì vậy mà ta cần tới mảng. Các mảng ở đây có thể là mảng 1 chiều (một cột hoặc 1 vector), mảng 2 chiều, mảng 3 chiều. LabVIEW hỗ trợ cho người lập trình có thể tạo ra các mảng của các mà trong đó chứa dữ liệu kiểu numeric, string, boolean... và rất nhiều các dạng dữ liệu khác. Các mảng thường được tạo ra bởi các vòng lặp. Việc sử dụng vòng lặp tốt cho các ứng dụng bởi vì nó xác định một vùng bộ nhớ xác định từ khi nó bắt đầu.

Một số hàm sử dụng thao tác trên mảng:

Việc sử dụng các hàm trên bảng Function >> Array để tạo và điều khiển các mảng



**Hình 6.18 Array**

Trên đó bao gồm các hàm sau

Array Constant

Array Max & Min

Array Size

Array Subset



Array To Cluster  
 Build Array  
 Cluster To Array  
 Decimate 1D Array  
 Delete From Array  
 Index Array  
 Initialize Array  
 Insert Into Array  
 Interleave 1D Arrays  
 Interpolate 1D Array  
 Replace Array Subset  
 Reshape Array  
 Reverse 1D Array  
 Rotate 1D Array  
 Search 1D Array  
 Sort 1D Array  
 Split 1D Array  
 Threshold 1D Array  
 Transpose 2D Array

**Chạy và gỡ rối chương trình:** Để chạy được chương trình bạn phải nối dây tất cả các sub Vis, cấu trúc, hàm, với đúng loại dữ liệu. cho những thiết bị đầu cuối. Bạn có thể sử dụng show context help để kiểm tra hàm và loại dữ liệu. Khi bạn nối dây mà dây bị gãy có nghĩa là không đúng loại dữ liệu.

Chạy VI: bạn có thể chạy VI khi nút run trên thanh công cụ không bị gãy như hình dưới đây:



Khi chạy đang chạy chương trình nút run sẽ như hình dưới:



Khi một VI bị lỗi và sẽ không chạy được thì nút run sẽ bị gãy như hình sau:



Khi VI bị lỗi thì chúng ta phải khắc phục hết lỗi mới có thể chạy VI được. Chúng ta khắc phục như sau: click vào nút chạy và sau đó chọn show error và xem nó bị lỗi gì và tìm cách khắc phục

**Tìm kiếm ví dụ có sẵn trong labview:** Từ font panel bạn chọn **help/find examples** hoặc từ cửa sổ **Getting started** chọn **find examples**. Sau đó tìm những ví dụ bạn cần quan tâm để mở.

+ **Màu của dự án:** Có thể thay đổi màu của nhiều công cụ và dự án sử dụng công cụ màu từ tools palette. click phải trên dự án muốn thay đổi màu để thay đổi



**Hình 6.19 Công cụ để thay đổi màu**

Bạn sử dụng công cụ này để thiết kế dự án của bạn thật đẹp với các màu sắc bạn thích. Bạn có thể thay đổi màu mặc định của các dự án bằng cách chọn **tools /color**

## 6.2.5 CÁC HÀM THÔNG DỤNG TRONG LABVIEW

### Các cấu trúc điều khiển luồng chương trình

Trong bất cứ ngôn ngữ lập trình nào, ta cũng hay thường gặp và làm việc với các phần tử điều khiển luồng chương trình, đó là các cấu trúc (Structures). Các cấu trúc điều khiển luồng chương trình trong một VI có năm cấu trúc là: For Loop, While Loop, Case Structure, Sequence Structure và Formula Node.

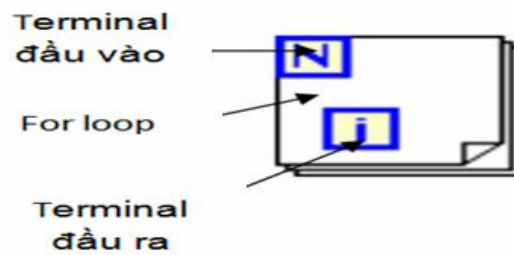
Các cấu trúc đó thực hiện tự động khi dữ liệu đầu vào của chúng có sẵn và thực hiện các công việc theo ý muốn cho tới khi hoàn thành thì mới cung cấp dữ liệu tới các dây nối dữ liệu đầu ra. Tuy nhiên, mỗi cấu trúc thực hiện theo các quy tắc riêng (Sub Diagram) của nó. SubDiagram là tập hợp của các Node, Wire và Terminal bên trong đường viền của Structure. Mỗi cấu trúc For Loop và While Loop có một SubDiagram. Cấu trúc Case và Sequence có thể có nhiều SubDiagram, nhưng chỉ có một SubDiagram có thể được thực hiện tại một thời điểm. Cách xây dựng các SubDiagram cũng giống như việc xây dựng các Block diagram mức đầu.

Việc truyền dữ liệu vào và ra các Structure thông qua các Terminal mà được tự động tạo ra ở nơi dây nối đi qua đường viền của cấu trúc, các Terminal này được gọi là các đường ống (Tunnel).

Các cấu trúc điều khiển chương trình trong LabVIEW nằm ở **Functions \programming\Structures**.

#### - For Loop.

For Loop là một cấu trúc lặp thực hiện sơ đồ bên trong nó một số lần định trước. khi thực hiện hết số vòng lặp đã được định trước thì vòng lặp For mới đưa ra kết quả thực hiện sơ đồ bên trong



Hình 6.20

Để truy cập For Loop ta chọn Menu: Functions\programming \ Structures \ For Loop. For Loop có hai Terminal đó là:



Count Terminal (Terminal đầu vào): Chỉ ra số lần vòng lặp phải thực hiện. để cài đặt số lần lặp ta click phải vào N và chọn constant và nhập số vào

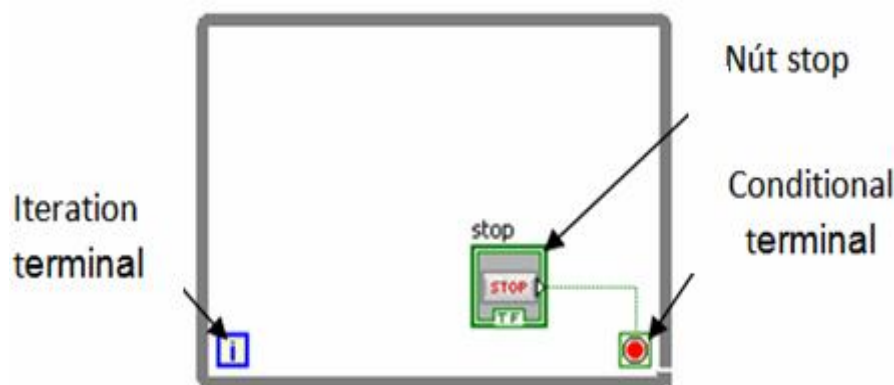


Iteration Terminal (Terminal đầu ra): Chứa số lần vòng lặp đã thực hiện. Giá trị của i thay đổi từ 0 tới n-1.

#### - While Loop

While Loop là một cấu trúc lặp thực hiện sơ đồ bên trong nó cho đến khi giá trị Boolean đưa tới Conditional Terminal (một Terminal đầu vào) là trùng với điều kiện được thiết lập để thực hiện vòng lặp.

Để truy cập While Loop ta chọn Menu: Functions\programming\ Structures \ While Loop. Khi ta lấy một vòng lặp while thì có một nút stop được đính kèm để dừng vòng lặp khi điều kiện vào là đúng. Bạn click phải vào đường biên của vòng lặp để chọn điều kiện là: stop if true hoặc continue if true từ menu đổ xuống.



Hình 6.21 While Loop

VI kiểm tra Conditional Terminal tại cuối mỗi vòng lặp, do đó While Loop luôn thực hiện ít nhất một lần. Iteration Terminal là một Terminal đầu ra mà đưa ra số lần vòng lặp thực hiện được. Việc tính sự lặp luôn được bắt đầu từ 0.

Vì vậy, nếu vòng lặp chạy một lần thì Iteration Terminal đưa ra kết quả 0. Việc thực hiện vòng lặp có thể được xác định thông qua Conditional Terminal. Tại Conditional Terminal. Ta thường sử dụng vòng lặp while khi muốn chương trình trong vòng lặp làm việc liên tục

Để truy cập while Loop ta chọn Menu: Functions\programming \ Structures \ while Loop.

+ **Thanh ghi dịch:** Sử dụng thanh ghi dịch khi bạn muốn đưa dữ liệu từ lần hoạt động trước tới

lần hoạt động tiếp theo một thanh ghi dịch như hình dưới:



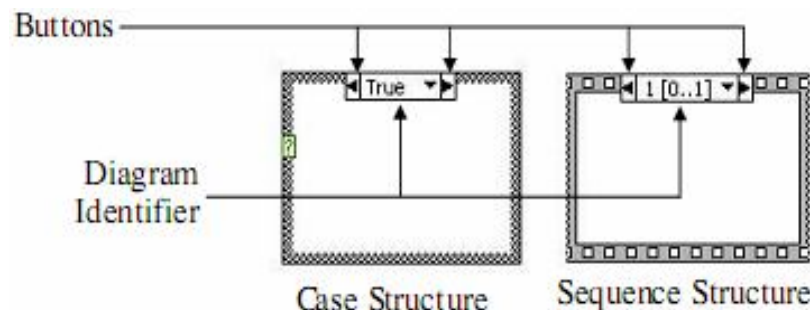
Để tạo một thanh ghi dịch bạn click phải vào đường biên của một vòng lặp và chọn **add shift resistor** từ menu đổ xuống

### - Case & Sequence Structures

Case Structure và Sequence Structure là các cấu trúc thực hiện một SubDiagram bên trong nó. Case Structure thực hiện SubDiagram dựa trên một giá trị đầu vào mà được bộ chọn gọi. Sequence Structure thực hiện các SubDiagram một cách tuần tự theo thứ tự mà chúng xuất hiện.

Cả Case Structure và Sequence Structure có thể có nhiều SubDiagram, nhưng chỉ có một SubDiagram được nhìn thấy tại một thời điểm. Tại đường viền bên trên của mỗi cấu trúc là một cửa sổ hiển thị SubDiagram mà gồm có sơ đồ định danh (Diagram Identifier) ở giữa và nút tăng (Increment), nút giảm (Decrement) ở hai bên. Diagram identifier chỉ ra SubDiagram hiện thời được hiển thị.

Đối với Case Structure thì diagram identifier là một danh sách các giá trị để lựa chọn SubDiagram. ví dụ ta có thể chọn true hoặc false Ta thường sử dụng cấu trúc case như hàm if trong ngôn ngữ lập trình dạng text Đối với Sequence Structure thì diagram identifier là khung (Frame) trong Sequence (từ 0 tới n-1). Đối với sequence là cấu trúc tuần tự ví dụ ta có thể thêm từng phần của cấu trúc bằng cách nhấp phải vào đường biên của cấu trúc và chọn add frame before hoặc add frame after. Khi thực hiện nó sẽ thực hiện hết phần này rồi đến phần khác tuần tự cho đến hết. Case Structure và Sequence Structure được minh họa ở hình 2. 25

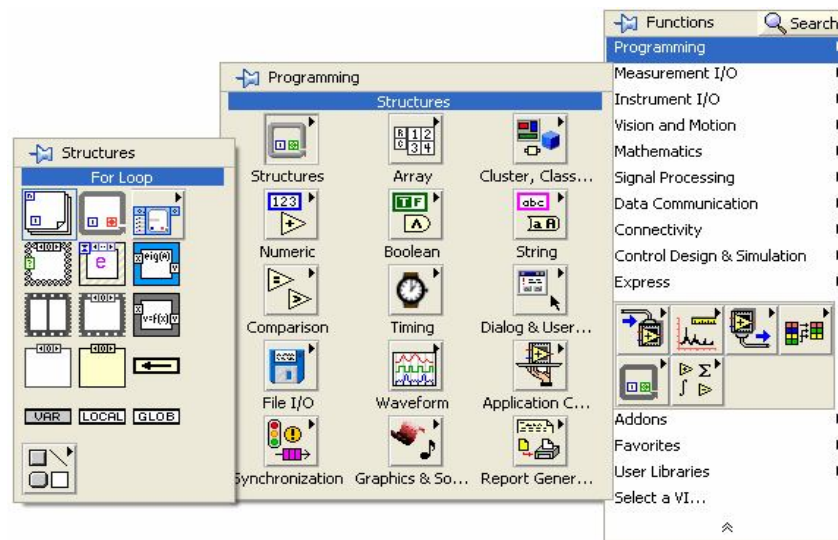


**Hình 6.22 Case Structure và Sequence Structure**

Khi nhấn vào nút tăng (bên phải) hoặc nút giảm (bên trái) sẽ làm hiển thị SubDiagram kế tiếp hoặc trước đó tương ứng. Nhấn nút tăng khi ở SubDiagram cuối cùng làm hiển thị SubDiagram đầu tiên còn nhấn nút giảm khi ở SubDiagram đầu tiên làm hiển thị SubDiagram cuối cùng. Để

truy cập Case Structure và Sequence Structure ta chọn Menu: **Functions \programming\ Structures \ Case**

Hoặc **Functions \programming\ Structures \ Sequence**



**Hình 6.23 chọn cấu trúc**

#### - Formula Node

Formula Node là cấu trúc dùng để thực hiện các dãy công thức toán học bên trong nó.

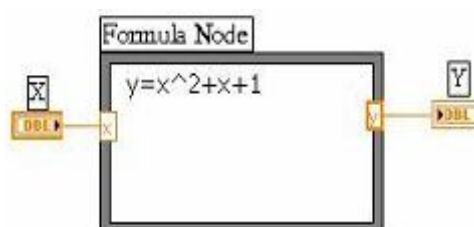
Khi dùng Formula Node ta có thể vào các công thức trực tiếp trong một Block diagram nhờ chuột ở chế độ Labeling Tool. Để tạo các đầu vào và các đầu ra, bấm vào đường viền của Formula Node và chọn Add Input (Add Output), đánh vào tên biến vào bên trong hộp. Các công thức được đánh vào bên trong đường viền của Formula Node, mỗi công thức trình bày phải kết thúc bằng dấu chấm phẩy (;).

Ví dụ minh họa Formula Node có dạng như hình 6. 24.

Để truy cập Formula Node ta chọn:

Functions \programming\ Structures \ Formula Node

Các toán tử và hàm có thể dùng bên trong Formula Node được liệt kê trong cửa sổ Help của Formula Node.



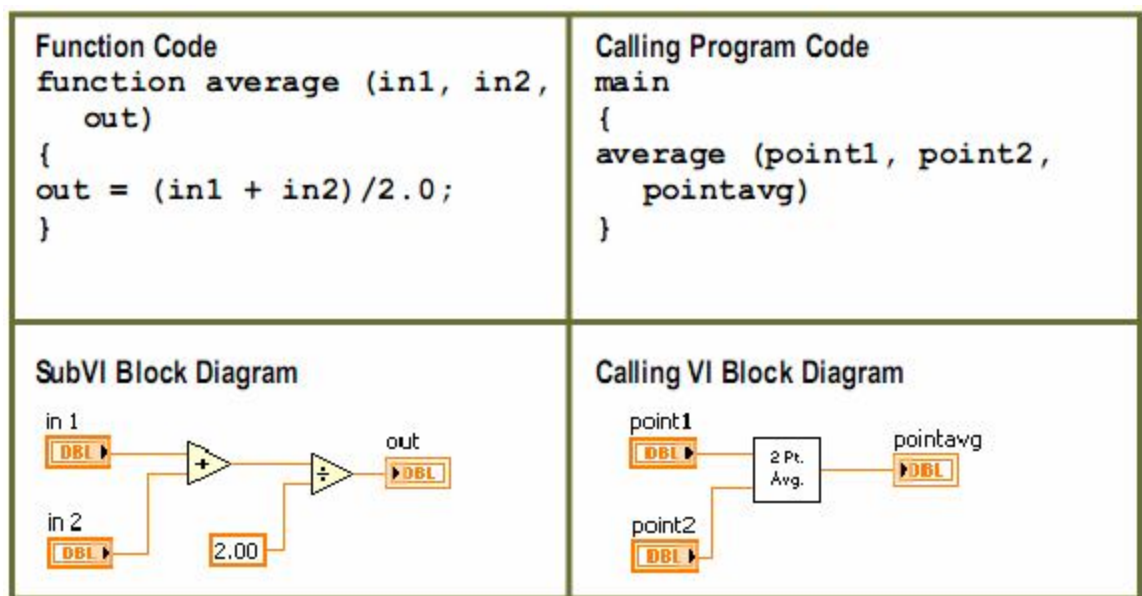
**Hình 6.24 Formula Node**

#### - SubVIs

Sau khi xây dựng một VI và biểu tượng và ô vuông liên kết (connector panel) của nó, thì ta có thể sử dụng nó trong một VI khác. Một VI bên trong một VI khác được gọi là subVI. Một subVI tương đương với một thủ tục con trong ngôn ngữ lập trình dạng text (hàm Record trong Pascal, Procedure trong Visual Basic). Sử dụng subVI có những thuận lợi sau:

- Chia ra từng Module rõ ràng
- Dễ dàng để debug (dò lỗi)
- Không phải tạo lại code
- Đòi hỏi ít bộ nhớ

Dưới đây là ví dụ về sự tương ứng của subVI với hàm thủ tục con trong ngôn ngữ lập trình dạng text.



**Hình 6.25 Tương ứng chương trình con dạng text**

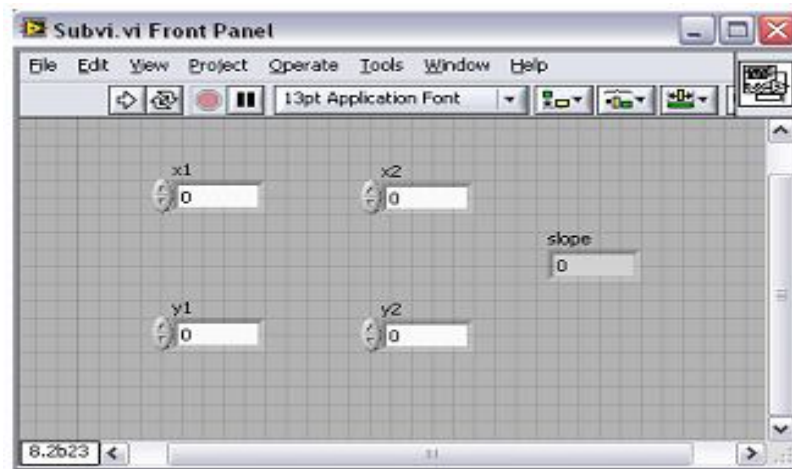
- **Ví dụ về Icon và connector panel**

Sau khi xây dựng một VI, ta phải tạo icon và connector pane cho nó để có thể sử dụng được như một subVI. Mỗi VI đều có một icon, nó nằm ở góc trên bên phải của cả 2 cửa sổ Front panel và Block diagram.

Để hiểu rõ cách tạo subVI và icon, connector pane của nó thì ta làm ví dụ sau:

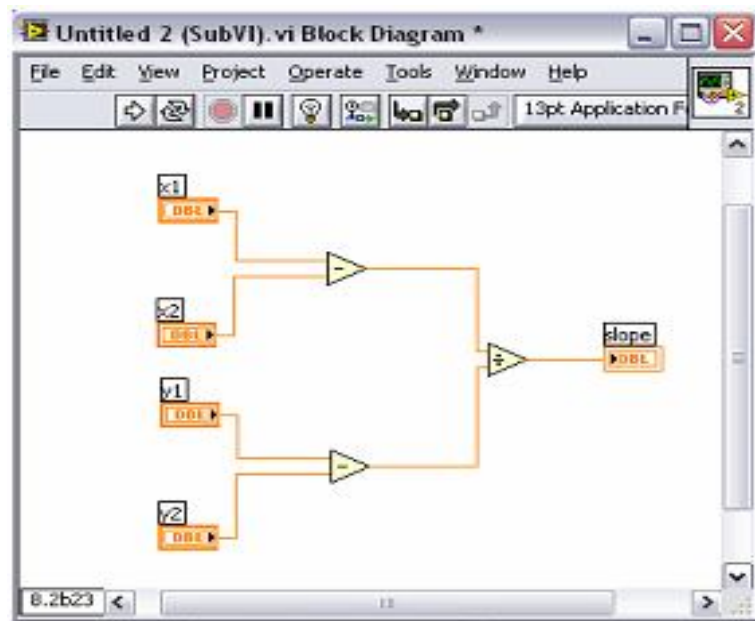
Tạo giao diện Front panel của VI giống hình bên dưới.





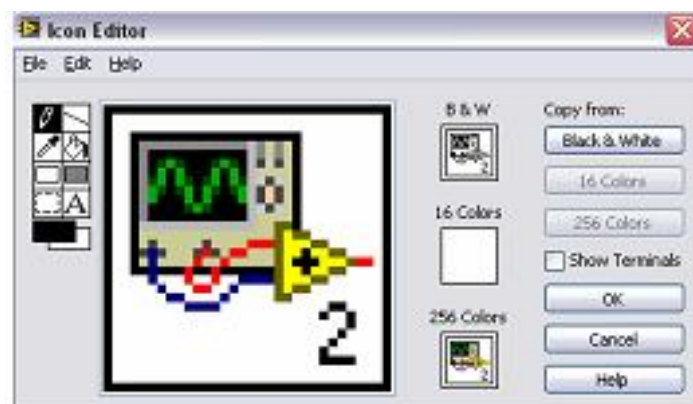
**Hình 6.26 Front Panel**

Và trong block diagram ta liên kết như sau:



**Hình 6.27 Block Diagram**

Để tạo icon thì ta click phải vào biểu tượng ở góc phải bên trên của Front panel hoặc Block diagram. Sau đó bảng Icon Editor xuất hiện như hình dưới, trong đó có các công cụ dùng để vẽ tương tự trong Paint.

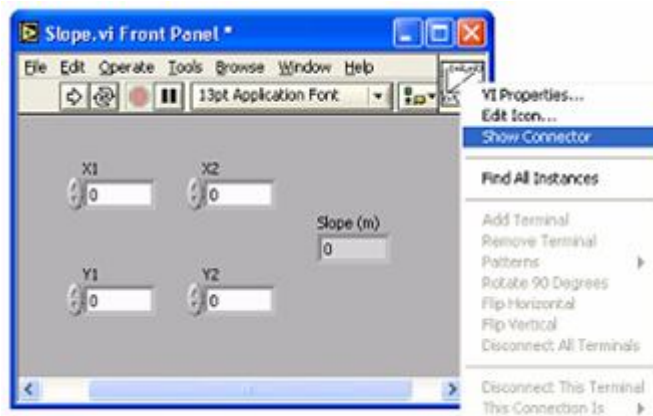




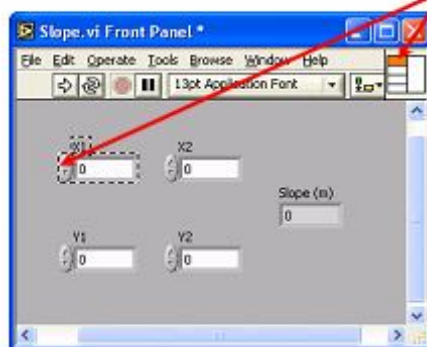
Giả sử ta vẽ Icon như sau :



Bây giờ ta tạo các Connector Panel. Click phải lên biểu tượng Icon, chọn Show Connector

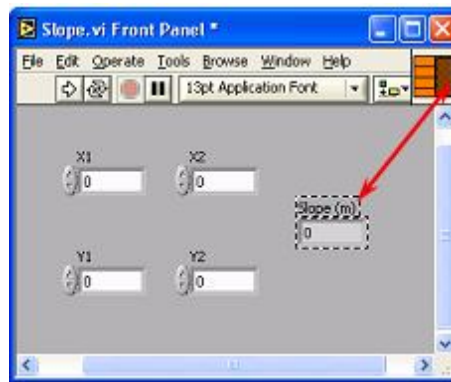


Sau đó tạo Connector : Khi tạo kết nối ta phải chọn chuột ở chế độ writing tool



Click ô vuông  
sau đó click  
vào X1

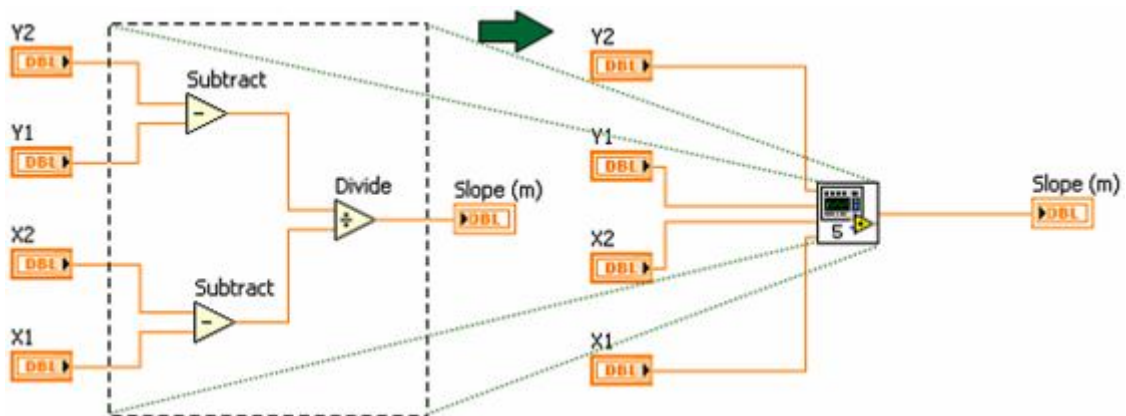
Làm tương tự, cho đến Connector cuối cùng như hình vẽ bên dưới.



Sau đó save lại với tên slope. vi. khi muốn sử dụng sub VI này Có thể click phải vào block diagram của VI sau đó vào Function palette / Use a VI... Rồi trở tới tập tin slope. vi hoặc ta có thể kéo icon của slope. vi đang mở sang block diagram của VI mới.



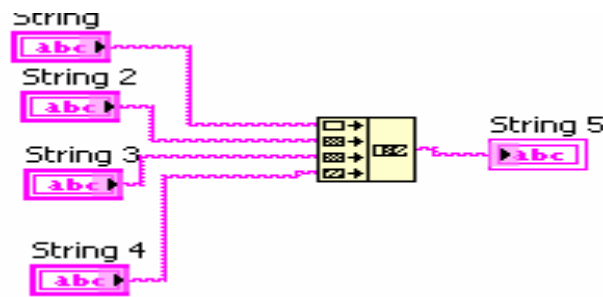
Ta có thể tạo một SubVI option bằng cách dùng chuột trái chọn vùng muốn tạo SubVI, sau đó vào Edit chọn Create a SubVI



Hình 2. 31 Chọn vùng để tạo chương trình con

- **NHÓM DỮ LIỆU:**

+ Ta có thể nhóm dữ liệu dùng **string**, string thông dụng trong các ô nhập dữ liệu để cho thuận tiện hoặc do mục đích nhiều lúc ta phải sử dụng gom nhiều string để cho ra một string mà nội dụng không thay đổi. giả sử bạn muốn kết hợp 4 string thành một string để truyền dữ liệu ra công com thì bạn có thể dùng **concatenate strings** để nhóm chúng lại thành một string để truyền ra. để sử dụng hàm nhóm string, từ block diagram bạn chọn **function/programming/string/concatenate strings**



Hình 2. 32 Nhóm dữ liệu

Ngoài string bạn có thể nhóm dữ liệu với **arrays**. hoặc **clusters**

- **ARRAYS**

Một array (mảng) là tập hợp những thành phần có cùng kiểu dữ liệu. Một array có thể có một hoặc nhiều chiều và mỗi chiều có thể chứa tối đa  $2^{31}$  phần tử. Array trong LabVIEW có thể thuộc bất cứ loại dữ liệu nào ngoại trừ bạn không thể tạo một array của các array, chart và graph. Ta có thể truy nhập bất cứ phần tử nào của một array nếu biết chỉ số của phần tử đó. Chỉ số phần tử được đánh số từ 0 tới N-1, trong đó N là số phần tử của array. Nên nhớ là phần tử đầu tiên được đánh số 0, phần tử thứ 2 được đánh số 1, và cứ như thế tiếp tục như thế...

| Chỉ số phần tử      | 0  | 1  | 2  | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
|---------------------|----|----|----|-----|-----|-----|-----|-----|-----|-----|
| Array có 10 phần tử | 12 | 32 | 82 | 8.0 | 4.8 | 5.1 | 6.0 | 1.0 | 2.5 | 1.7 |

Hình 2.

34 Arrays

+ **Tạo array điều khiển và array chỉ thị**

Ta khá dễ dàng để tạo một array điều khiển và chỉ thị bằng cách tạo một vỏ array (array shell) chứa một đối tượng dữ liệu (có thể là dạng Boolean, string, numeric hay cluster).

**Bước 1:** Để tạo một vỏ array thì ta vào Controls palette \ programming\Arrays, Matrix & Clusters \ Array.

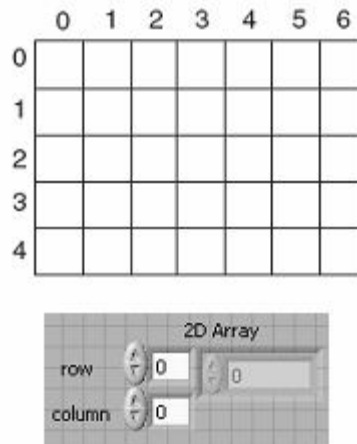


**Bước 2:** Vào Controls palette rồi chọn các kiểu dữ liệu muốn tạo array, sau đó để vào trong vỏ array.



### 2D Array

Đối với một 2D array thì cần hai chỉ số phần tử để có thể truy xuất phần tử trong array một chỉ số theo cột và một chỉ số theo hàng. Muốn thêm n chiều vào array thì click phải lên array chọn Add Dimension



Hình 2. 34 2D Arrays

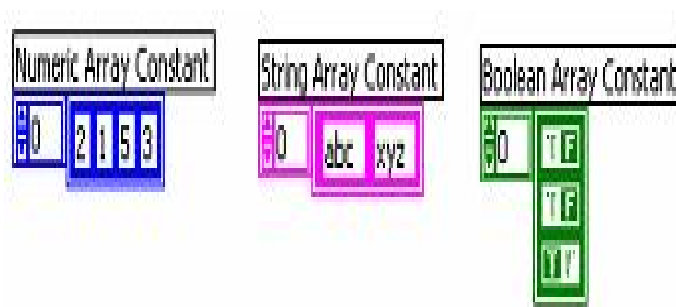
#### + Tạo một Array hằng số

Ta có thể tạo một Array hằng số trong Block Diagram bằng cách tạo một vỏ Array chứa đối tượng dữ liệu giống như làm trong Font Panel.

**Bước 1 :** Để tạo vỏ Array thì ta vào Function Palette \ programming/Array \ Array Constant



**Bước 2 :** Từ Function Palette ta kéo một đối tượng dữ liệu vào trong vỏ Array, có thể là dạng Numeric, Boolean, String.

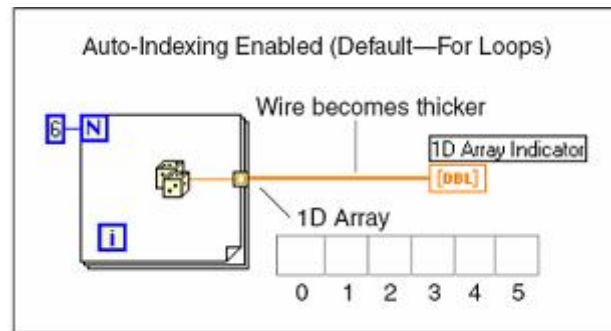


Hình 2. 35 Tạo array

- Tạo Array bằng vòng lặp

Vòng lặp For và While có thể lặp array và thêm các phần tử vào array đó. Chức năng này được gọi là Auto-indexing. Mỗi vòng lặp sẽ tạo thêm một phần tử kế tiếp trong array. Ta nhận thấy là dây nối

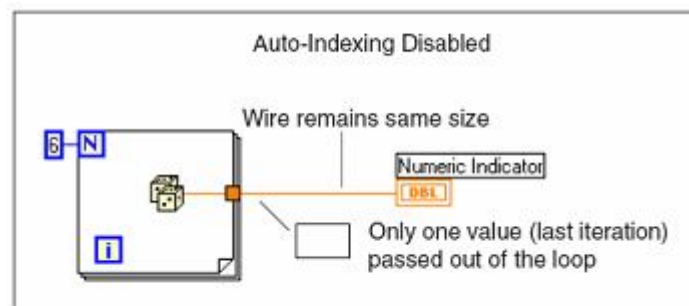
giữa hình vuông màu cam ở cạnh vòng lặp với array sẽ dày hơn bình thường, có thêm xem ví dụ hình bên dưới.



**Hình 2. 36 Tạo array bằng For loop**

Nếu bạn chỉ cần giá trị cuối cùng của array khi vòng lặp kết thúc mà không cần tạo array, thì bạn tắt chức năng Auto-Indexing, bằng cách click phải lên hình vuông màu cam và chọn Disable Indexing, khi đó dây dẫn nối với array sẽ trở lại kích thước bình thường.

Ví dụ bên dưới cho kết quả cuối cùng của hàm Random Number (0-1) sau khi kết thúc vòng lặp.

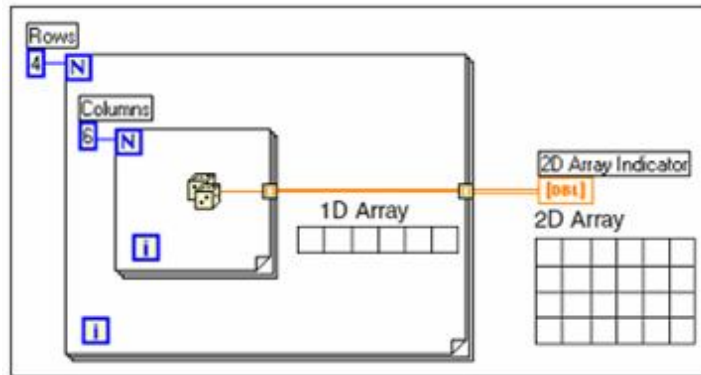


**Hình 2. 37 Tạo số ngẫu nhiên**

**Chú ý:** LabVIEW để chế độ Auto-Indexing cho vòng lặp For, còn đối với vòng lặp While thì mặc định là Disable Indexing.

#### + Tạo Array 2D bằng vòng lặp

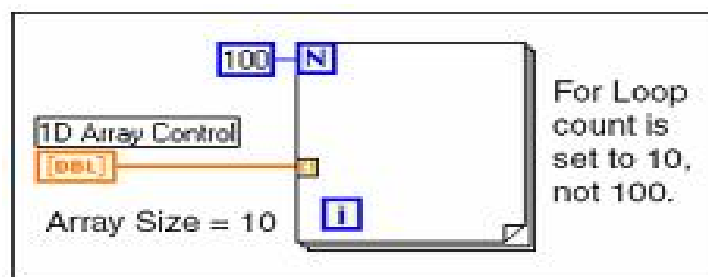
Ta sử dụng 2 vòng lặp For, cái này bên trong cái kia để tạo array 2D. Vòng For bên ngoài tạo các phần tử dòng, vòng For bên trong tạo các phần tử thuộc cột. Ví dụ hình bên dưới là tạo một array 2D với các phần tử là các số ngẫu nhiên từ 0-1.



Hình 2. 38 Tạo mảng 2D

#### + Sử dụng Auto-Indexing để chỉ định số lần lặp của vòng lặp For

Khi bạn bật chức năng Auto-Indexing của một array đi vào vòng For, thì LabVIEW sẽ tự động chỉ định số vòng lặp tương ứng với kích thước array, cho dù ta cho hằng số lặp vào. Nếu có hơn một array đi vào vòng For thì số lần lặp được lấy từ kích thước của array nhỏ hơn. Trong ví dụ hình bên dưới thì số lần lặp của vòng For là 10 chứ không phải 100.

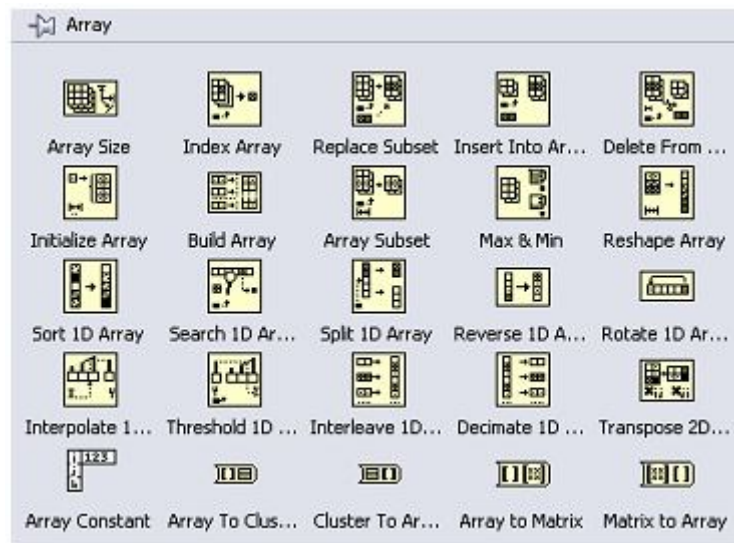


Hình 2. 39 Mảng điều khiển

#### + Các hàm thao tác với Array

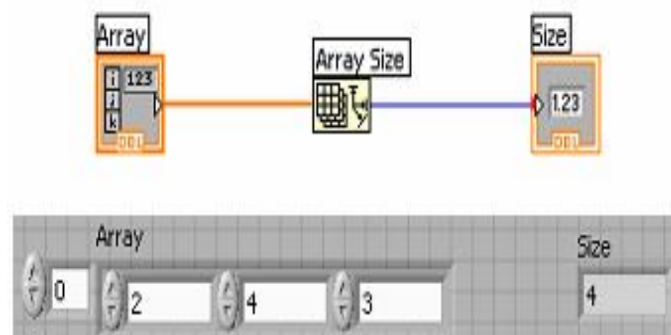
LabVIEW có nhiều hàm thao tác trên Array, được tìm thấy ở Function pallette \ Array.

Vài hàm thông dụng sẽ được thảo luận ở đây.



Hình 2. 40 Function pallette \ Array

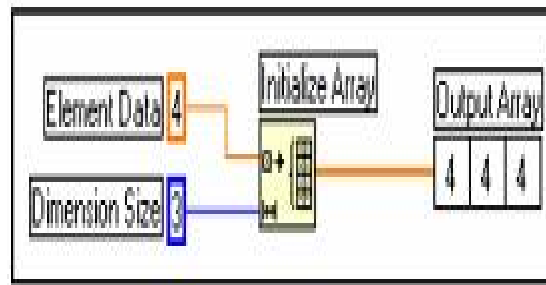
- **Array size:** Hàm này sẽ cho biết số phần tử của array. Nếu array có N chiều thì Array size sẽ trả về một dãy gồm N phần tử, trong đó mỗi phần tử sẽ là số phần tử của mỗi chiều của array đó



Hình 2. 41 Xác định kích thước mảng

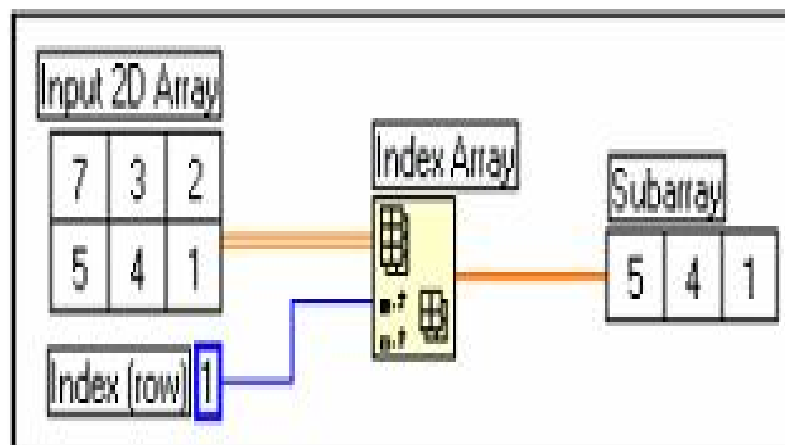
- **Initialize Array:** hàm này sẽ tạo một array có kích thước của chiều chứa giá trị phần tử. Bạn có thể chỉnh kích thước dài ngắn của Initialize Array tương ứng nếu bạn muốn có nhiều chiều. Ví dụ bên dưới là tạo ra một array 1D có ba phần tử với giá trị bằng 4.





Hình 2. 42 Tạo mảng

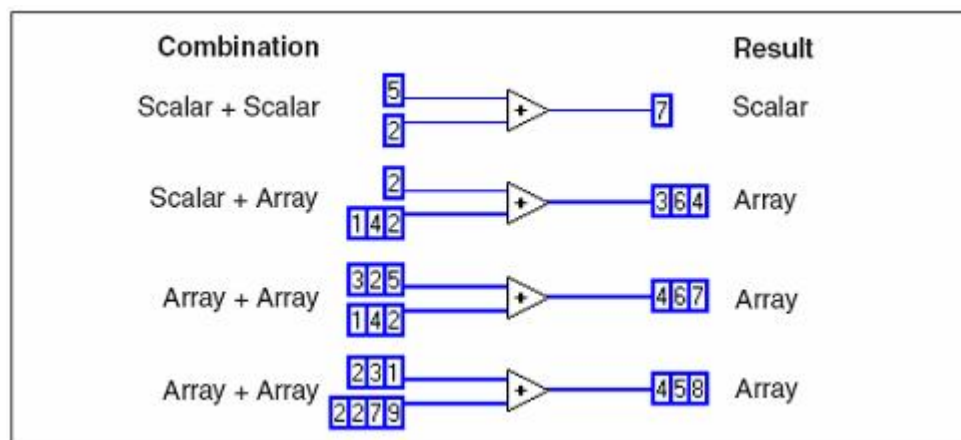
Nếu array đi vào hàm Index Array là array có N chiều thì biểu tượng hàm Index Array sẽ tự động thay đổi như hình dưới khi ta nối dây. Lúc này hàm sẽ trích xuất một hàng hay một cột của array, tùy thuộc vào ta đặt chỉ số Index nối ở hàng hay ở cột, như hình bên dưới ta đặt chỉ số ở Index row.



Hình 2. 43 Mảng truy xuất hàng cột

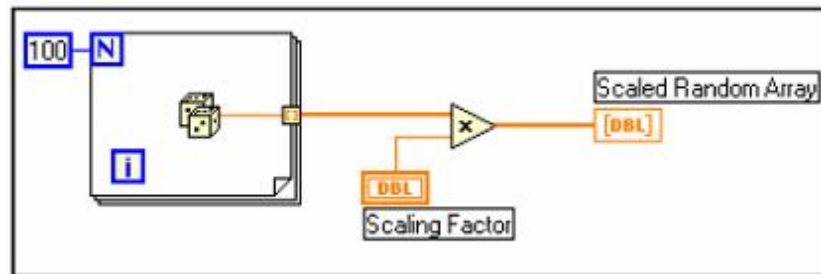
- **Sự đa hình thái (POLYMORPHISM)**

Những hàm toán tử trong LabVIEW có nhiều hình thái, nghĩa là đầu vào các hàm này có cấu trúc dữ liệu khác nhau – scalar (vô hướng) và array (mảng). Ví dụ như bạn có thể cộng một số vô hướng vào một mảng hoặc cộng hai mảng với nhau, như bảng bên dưới



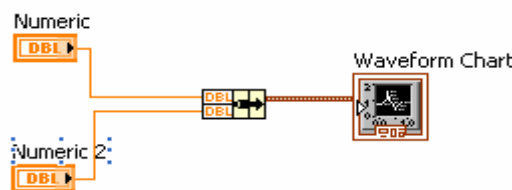
**Hình 2. 44 Các hình thức cộng mảng**

Thêm ví dụ bên dưới, sau mỗi vòng lặp thì giá trị đi ra vòng lặp sẽ được nhân với hệ số tỷ lệ (Scaling Factor), sau đó xuất vào array.



**Hình 2. 45 Tạo mảng bằng For loop**

Bó dữ liệu với **clusters**: Khi ta muốn một đồ thị mà hiện thị 2 hoặc nhiều thông số dữ liệu khác nhau thì ta phải sử dụng hàm bó dữ liệu. nó có tác dụng lấy dữ liệu từ 2 hoặc nhiều đầu vào và kết hợp lại một đầu ra như mảng để đồ thị có thể hiểu và hiện thị 2 hoặc nhiều thông số khác nhau



**Hình2. 46 Bó dữ liệu**

Để sử dụng chức năng bó dữ liệu trên. Từ **Function/programming/ clusters/ bundle**

- **Các loại biểu đồ**

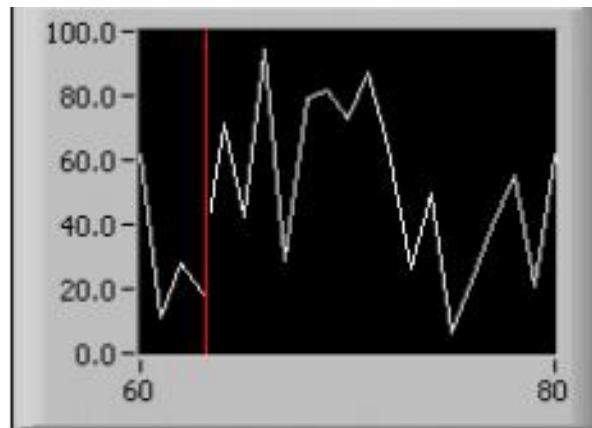
Charts (biểu đồ) và graphs (đồ thị) trong LabVIEW cho phép hiển thị dữ liệu ở dạng đồ họa. Charts không những vẽ biểu đồ mà còn cập nhật dữ liệu mới vào đồ thị. Graphs thì vẽ đồ thị của các dãy giá trị được khởi tạo trước theo cách truyền thống mà không giữ lại dữ liệu được khởi tạo trước đây. Trong phần này bạn sẽ được giới thiệu cách sử dụng Charts và Graph, những tính năng đặc biệt của chúng, và 3D graph, waveform wave. Cuối cùng là bạn sẽ làm quen với kiểu dữ liệu dạng sóng (waveform data), một công cụ trình diễn dữ liệu theo thời gian (time-based data).

## + CHARTS

### . Waveform charts

Một biểu đồ đơn giản chỉ là vẽ các giá trị trục Y so với giá trị trục X. Thông thường thì trục Y là trục dữ liệu, còn trục X là trục thời gian. Waveform chart (được tìm thấy trong Modern \ Graph

của Controls palette) là một bộ chỉ thị số đặc biệt, có thể hiển thị nhiều dòng một hay nhiều dữ liệu cùng một lúc. Thông thường waveform chart được đặt trong vòng lặp. Trong một chart, giá trị trục Y đại diện cho dữ liệu dữ liệu mới, giá trị trục X đại diện cho thời gian (thông thường, giá trị Y được khởi tạo trong một vòng lặp, vì vậy giá trị X là thời gian của một vòng lặp). Ta có thể thiết lập các thông số cho biểu đồ như các trục x, y, các đường nét hiển thị, các đường giống của biểu đồ...vv. Bằng cách nhấp phải chuột chọn từ menu hoặc vào property để cài đặt các thông số cần thiết.

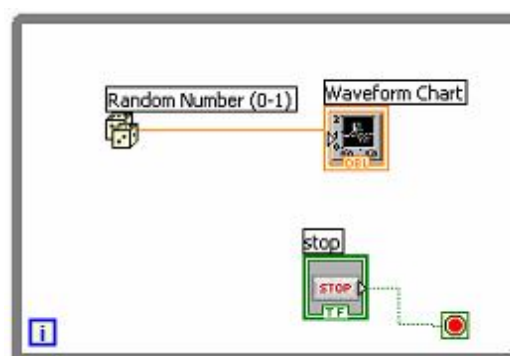


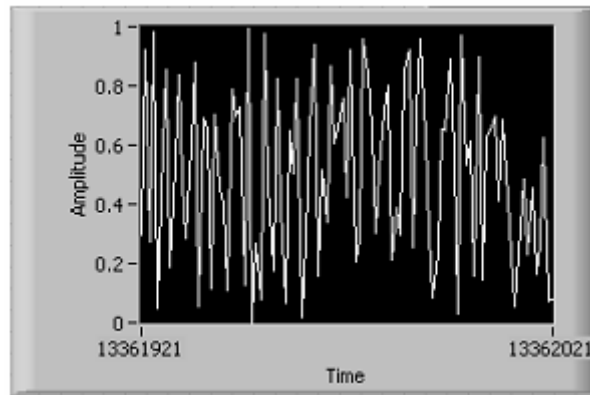
**Hình 2. 47 Waveform charts**

Waveform chart (biểu đồ dạng sóng) có 3 chế độ hiển thị: Strip chart, scope chart, sweep chart. Có thể thay đổi các chế độ hiển thị trong khi bằng cách click phải lên chart rồi vào Advanced \ Update Mode.

### . Single-Plot Charts

Ví dụ sử dụng chart để vẽ đồ thị của hàm random

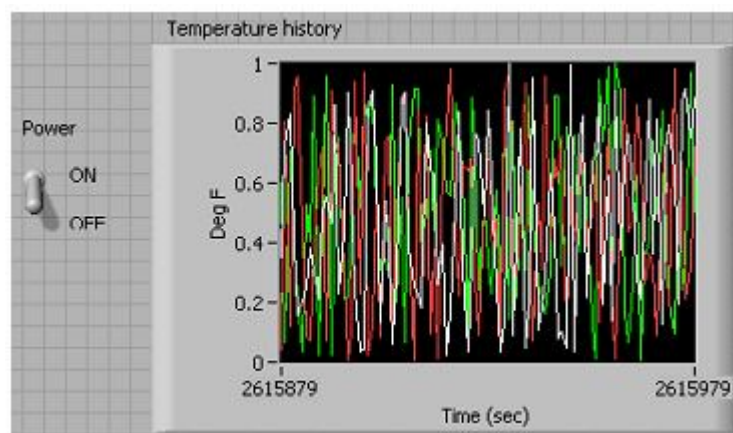
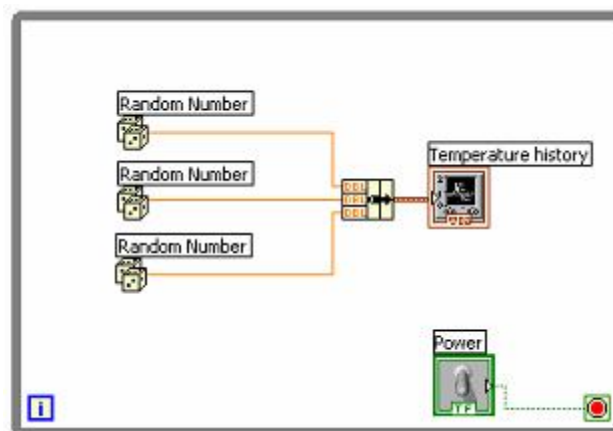




**Hình 2. 48 Single-Plot Charts**

### . Multiple-Plot Chart

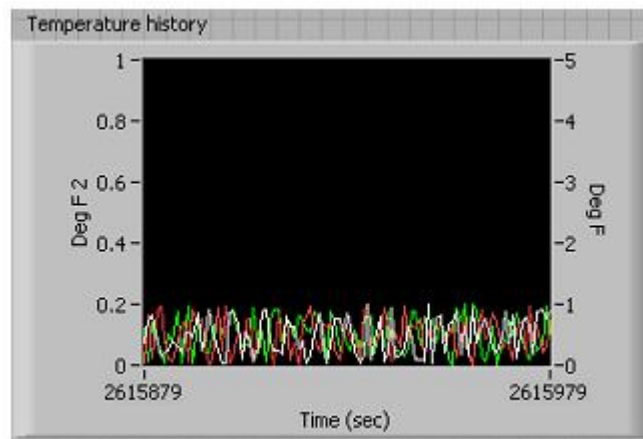
Waveform chart có thể hiển thị được nhiều đường dữ liệu. Tuy nhiên, chúng không thể nối nhiều dây từ nhiều nguồn vào 1 chart được, vì vậy ta phải sử dụng function Bundle (Programming \ Cluster & Variant) để “bó” các dây lại rồi chỉ truyền 1 dây tới chart. Bundle có thể được kéo dài ra tương ứng với số nguồn kết nối vào.



**Hình 2. 49 Multiple-Plot Chart**

### . Multiple Y Scales

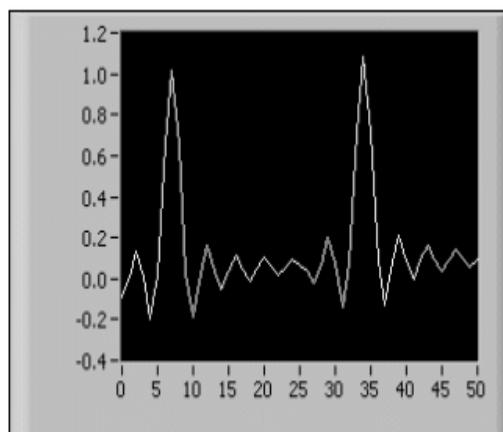
Nếu bạn đã có một Multiple-Plot Chart, thì đôi khi bạn muốn thể hiện những giá trị ở những cột giá trị khác nhau, những dòng dữ liệu có giá trị chênh lệch khá xa thì nên thể hiện ở 2 cột giá trị có tỷ lệ khác nhau. Để tạo thêm một trục Y thì ta click phải lên trục Y của chart và chọn Duplicate Scale, sau đó để di chuyển nó qua bên kia thì click phải lên trục vừa mới tạo chọn Swap Sides. Để xóa trục đó thì click phải chọn Delete Scale.



**Hình 2. 50 Multiple Y Scalse**

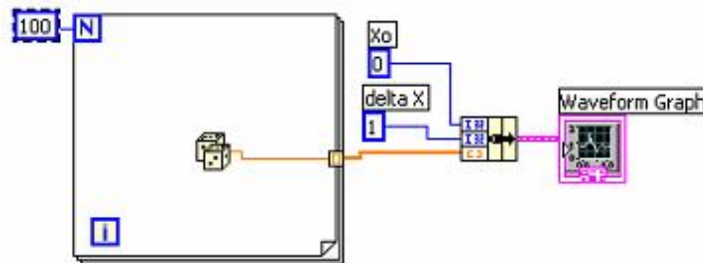
## + GRAPHS

Không giống như Chart, graph vẽ đồ thị của dãy dữ liệu đã được khởi tạo trước và không có khả năng thêm những giá trị mới vào dữ liệu được khởi tạo trước đó. LabVIEW cung cấp vài kiểu graph linh động như: waveform graphs, XY graphs, intensity graphs, 3D graphs, digital waveform graphs, và vài graph khác (Smith plots, Polar charts, Min-Max, and distribution plots). Chúng ta sẽ tìm hiểu về waveform graphs, XY graphs, và sau đó là intensity graphs, 3D graphs. Waveform và XY graphs nhìn giống nhau ở màn hình fron panel nhưng thực ra chúng có chức năng khác nhau.



**Hình 2. 51 Single-Plot Waveform Graphs**

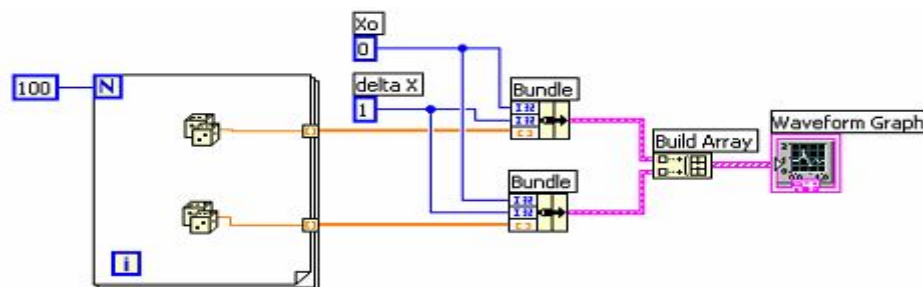
Đối với Single-Plot Wave form graph, thì ta có thể đơn giản kết nối một dãy giá trị Y tới terminal Waveform graph (hình bên dưới). Phương pháp này giả thiết là giá trị ban đầu của X là 0, và giá trị  $\Delta X = 1$ .



Hình 2. 52 Single-Plot Waveform Graphs

#### + Multiple-Plot Waveform Graphs

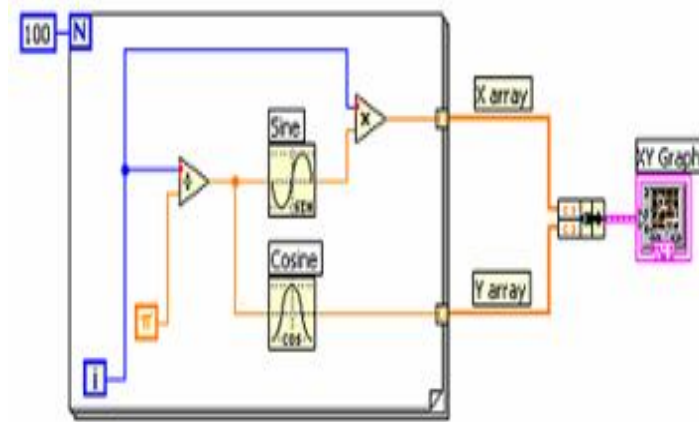
Bạn có thể biểu diễn nhiều đường trên cùng một graph bằng cách tạo một array của các kiểu dữ liệu. Như ví dụ hình bên dưới thì ta sẽ xây dựng một array 2D bằng cách sử dụng Bundle Array, theo mặc định thì LabVIEW sẽ vẽ 2 đường cho 2 dòng của array 2D. Nếu dữ liệu của bạn được xếp theo cột thì bạn phải chuyển vị thành hàng hết, có thể dùng terminal Transpose 2D Array trong Function palette.



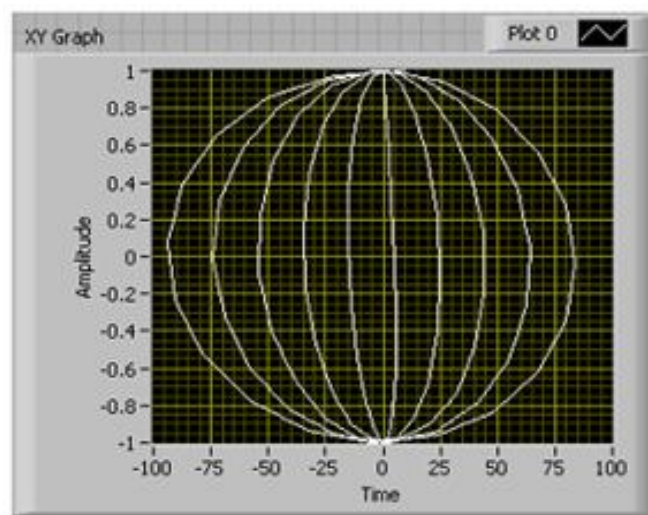
Hình 2. 53 Multiple-Plot Waveform Graphs

#### + Graphs :

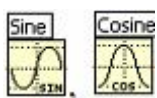
XY graphs (trong Controls palette \ Modern \ Graph) biểu diễn cho các kiểu dữ liệu khác nhau. Nếu bạn muốn vẽ một hàm hay muốn khảo sát các dữ liệu ở khoảng thời gian không đều. Ví dụ như ta vẽ hình bên dưới.



Hình 2. 54 Sơ đồ khối sau khi xây dựng xong



Hình 2. 55 XY Graphs

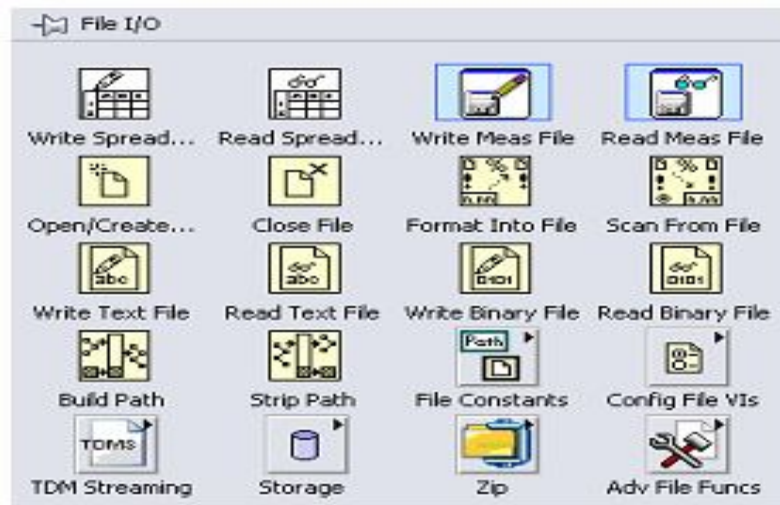


: Hai terminal Sine và Cosine được tìm thấy trong Function palette \ Express \ Arith & Comparison \ Math \ Trigonometric.

#### + FILE INPUT/OUTPUT

File I/O lưu trữ và nhận thông tin từ file hoặc đĩa. LabVIEW có nhiều hàm thao tác, đặt trong Function palette \ File I/O

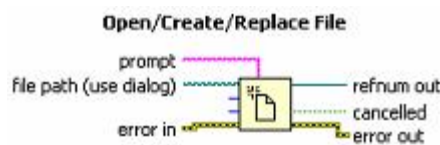




Hình 2. 56 Function palette \ File I/O

+ Các hàm cơ bản của file input và output

- **Open / Create / Replace File** : Mở một file, tạo file mới, hoặc thay thế một file hiện có một cách tương tác thông qua việc sử dụng hộp dialog



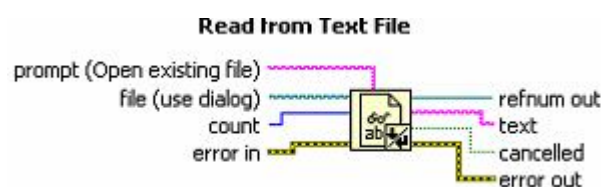
Hình 2. 57 Open / Create / Replace File

- **Write to Text File**: Ghi một chuỗi các ký tự hoặc một array chuỗi thành những hàng vào trong một file. Nếu một đường dẫn được nối với input file (use dialog) thì hàm sẽ mở hoặc tạo file đó trước khi ghi hoặc thay thế những nội dung trong đó. Nếu một file refnum nối với file (use dialog) thì hàm sẽ viết vào file hiện tại



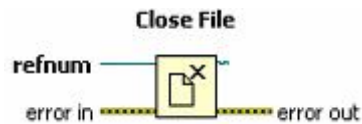
Hình 2. 58 Write to Text File

- **Read from Text File**: Đọc các chuỗi ký tự của một file cụ thể



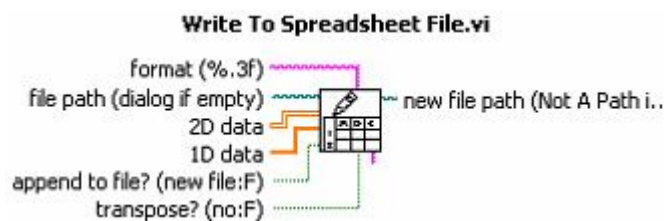
**Hình 2. 59 Read from Text File**

- **Close File:** Đóng một file đã mở được chỉ định bởi refnum.



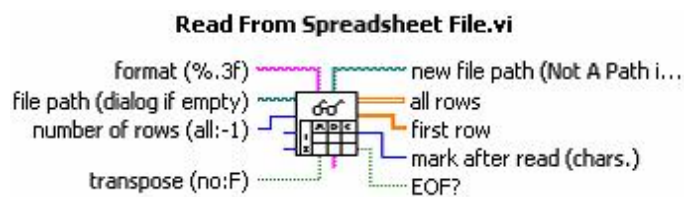
**Hình 2. 60 Close File**

- **Write to Spreadsheet File. vi:** Chuyển các array 1D, 2D kiểu số thành kiểu string dạng text, sau đó ghi các chuỗi này vào một file mới. Ngoài ra, bạn có thể chuyển vị dữ liệu hàng thành cột, cột thành hàng. Chú ý khi ghi file ta nên viết thêm đuôi định dạng. cvs hoặc. xls ngay sau tên file để Excel có thể đọc được.



**Hình 2. 61 Write to Spreadsheet File. vi**

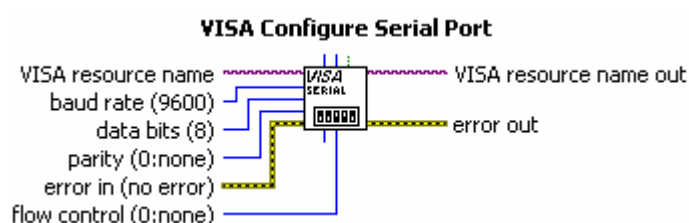
- **Read From Spreadsheet File. vi:** Đọc một số lượng dòng hoặc cột cụ thể từ một file text kiểu numeric, bắt đầu tại một vị trí offset cụ thể nào đó, và chuyển dữ liệu thành 2D Array. Bạn có thể lựa chọn chuyển vị Array



**Hình 2. 62 Read From Spreadsheet File. vi**

+ Các hàm truyền thông nối tiếp trong LabVIEW :

- . **VISA Configure Serial Port VI :** dùng để cài đặt các thông số khi truyền thông nối tiếp



**Hình 2. 63 visa configure SeriPort**

- VISA resource name : Chọn thông số cổng Com.
- Baud rate : Chọn tốc độ baud. Mặc định là 9600.
- Data bits : Số bit dữ liệu vào. Mặc định là 8 bit
- Parity : Bit chẵn lẻ - dùng khi được truyền - nhận. Mặc định là 0
- Error in : Thông báo lỗi trước khi chức năng này hoạt động. Mặc định là No Error.
- Flow control : Điều khiển lưu lượng truyền dữ liệu. Mặc định là 0
- VISA resource name out : Là toàn bộ giá trị của VISA resource name
- Error out : Nếu lỗi ngõ vào chứa thông tin lỗi trước khi VI chạy thì lỗi ngõ ra sẽ mang thông tin lỗi đó.

**VISA Write Function**

VISA Write Function : Ghi dữ liệu từ Write Buffer tới thiết bị hay giao diện được chỉ bởi VISA resource name

**Hình 2. 64 visa write**

- VISA resource name : Chọn thông số cổng Com.
- Write buffer : Chứa dữ liệu sẽ được ghi tới thiết bị.
- Error in : Thông báo lỗi trước khi chức năng này hoạt động. Mặc định là No Error.
- VISA resource name out : Là toàn bộ giá trị của VISA resource name
- Return count : Chứa những byte thực sự được ghi ra
- Error out : Nếu lỗi ngõ vào chứa thông tin lỗi trước khi VI chạy thì lỗi ngõ ra sẽ mang thông tin lỗi đó.

**VISA Read Function:**

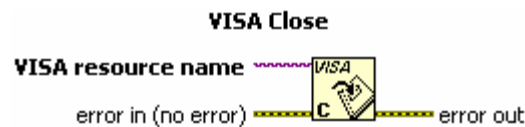
VISA Read Function : Đọc dữ liệu từ Write Buffer tới thiết bị hay giao diện được chỉ bởi VISA resource name và ghi dữ liệu ra read buffer.

**Hình 2. 65 Visa read**

- VISA resource name : Chọn thông số cổng Com.
- Byte count : Số byte đã đọc vào.
- Error in : Thông báo lỗi trước khi chức năng này hoạt động. Mặc định là No Error.
- VISA resource name out : Là toàn bộ giá trị của VISA resource name.
- Read buffer : Chứa dữ liệu đọc từ thiết bị.
- Error out : Nếu lỗi ngõ vào chứa thông tin lỗi trước khi VI chạy thì lỗi ngõ ra sẽ mang thông tin lỗi đó.

**VISA Close :**

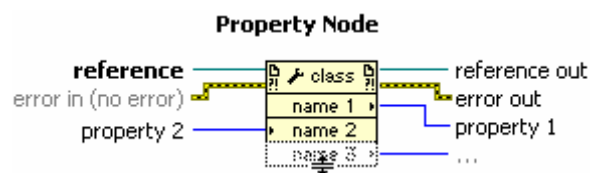
VISA Close : Đóng một bộ phận thiết bị hoặc một đối tượng sự kiện được chỉ rõ bởi VISA resource name



**Hình 2. 66 Visa close**

- VISA resource name : Chọn thông số cổng Com.
- Error in : Thông báo lỗi trước khi chức năng này hoạt động. Mặc định là No Error.
- Error out : Nếu lỗi ngõ vào chứa thông tin lỗi trước khi VI chạy thì lỗi ngõ ra sẽ mang lỗi đó

+ **Property Node:**

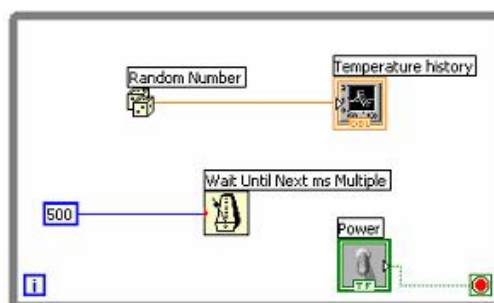


**Hình 2. 67 Property Node**

- **Reference** : Giá trị tham chiếu.
- **Property 2** : Là những ví dụ những thuộc tính mà bạn muốn ghi.
- **Reference out** : **Trả lại giá trị tham chiếu không thay đổi.**
- **Property 1** : Là những ví dụ những thuộc tính mà bạn muốn đọc.
- Error in : Thông báo lỗi trước khi chức năng này hoạt động. Mặc định là No Error.
- Error out : Nếu lỗi ngõ vào chứa thông tin lỗi trước khi VI chạy thì lỗi ngõ ra sẽ mang lỗi thông tin đó.

### Time for loop

Thông thường LabVIEW thực hiện vòng lặp rất nhanh. Đôi khi chúng ta muốn xem dữ liệu tại thời điểm nào đó, thì để cho vòng lặp chậm lại thì ta thêm vào 1 terminal Wait Until Next ms Multiple (Programming / Timing) , rồi tạo thêm 1 Numeric Constant kết nối với nó, đánh 500 (milisecond) vào. Như vậy, sau mỗi nửa giây là vòng lặp được lặp lại. Xem ví dụ hình bên dưới.



**Hình 2. 68 Time for loop**

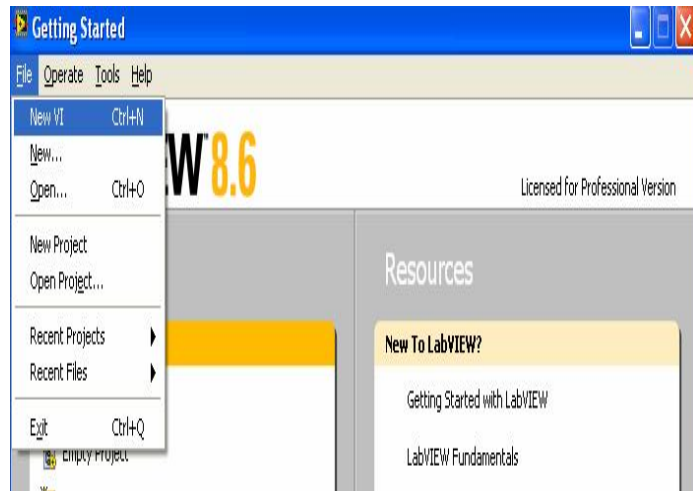
### 2. 2. 4 Một số phím tắt:

1. CTRL+H: Hiển thị context help
2. CTRL+E: Mở front panel hay block diagram
3. CTRL+F: Tìm các dự án
4. CTRL+L: Hiển thị bảng thông tin lỗi
5. CTRL+N: Tạo VI mới
6. CTRL+O: Mở một VI đã có sẵn
7. CTRL+W: Đóng một VI đang mở
8. CTRL+Q: Thoát labview
9. CTRL+? Hoặc F1: Hiển thị labview help
10. CTRL+Z: Xóa bỏ hành động vừa làm xong.
11. CTRL+Z+SHIFT: Làm lại hành động vừa làm
12. CTRL+C: Copy dự án đã chọn
13. CTRL+V: Dán dự án đã copy hoặc cắt trước đó
14. CTRL+R: Chạy VI.
15. CTRL+. Dừng VI
16. SHIFT+click phải: Hiển thị tools
17. ESC: Xóa bỏ lệnh thực thi hiện tại
18. CTRL+B: Xóa bỏ những dây bị lỗi (gãy)

## **2. 2. 5 Các ví dụ minh họa xây dựng một VI cụ thể trên LabVIEW:**

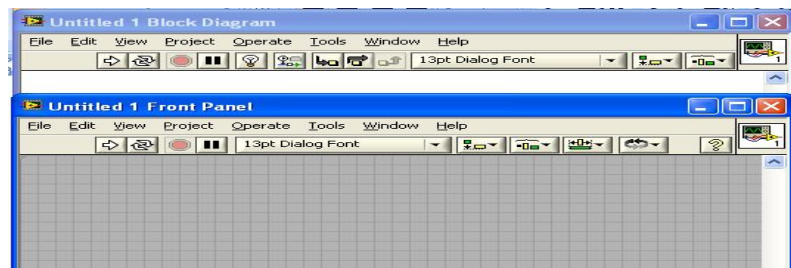
- Ví dụ đơn giản đếm sản phẩm khi đếm đủ số lượng đặt thì led sáng

**Bước 1 : start/all program/national instruments labview 8. 6**



Hình 2. 69 Getting started

**Bước 2:** Từ cửa sổ Getting started chọn **file/ new VI** ta được cửa sổ làm việc của Font panel và Block diagram như hình dưới

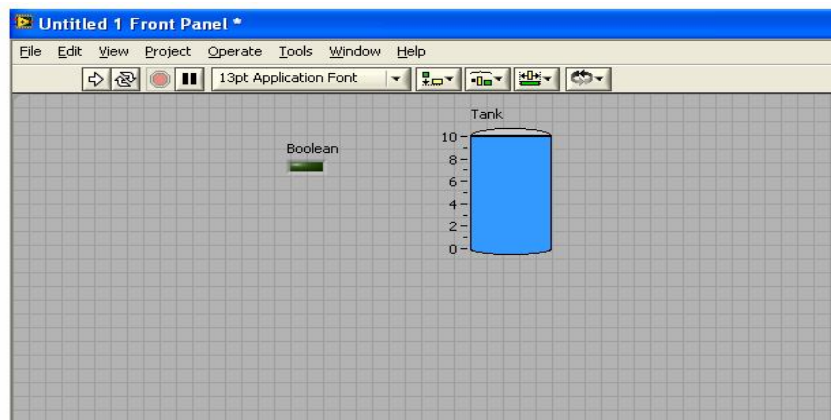


Hình 2. 70 Font Panel và Block Diagram của một VI

**\* Xây dựng Front Panel :**

**Bước 1.** Dùng chức năng controls để lấy và đặt các công cụ lên Font panel

- Controls/classic/classic number/tank
- Controls/modern/Boolean/square led
- Front panel của VI sẽ giống như minh hoạ dưới đây.



**Hình 2. 71 Font panel sau khi xây dựng xong**

**\* Xây dựng Block Diagram.**

- Mở Block diagram bằng cách từ font panel chọn **Windows / Show Diagram**
- Đặt các đối tượng được chọn dưới đây từ **Functions palette** lên cửa sổ Block Diagram.



increment: Function/programming/numeric/increment



geater or equal: Function/programming/comparsion/geter or equal



Numeric Constant (Functions \programming\ Numeric\constan) là hằng số. Dùng chuột ở chế độ Labeling Tool đặt giá trị của nó. Trong ví dụ này ta cần hai hằng số.

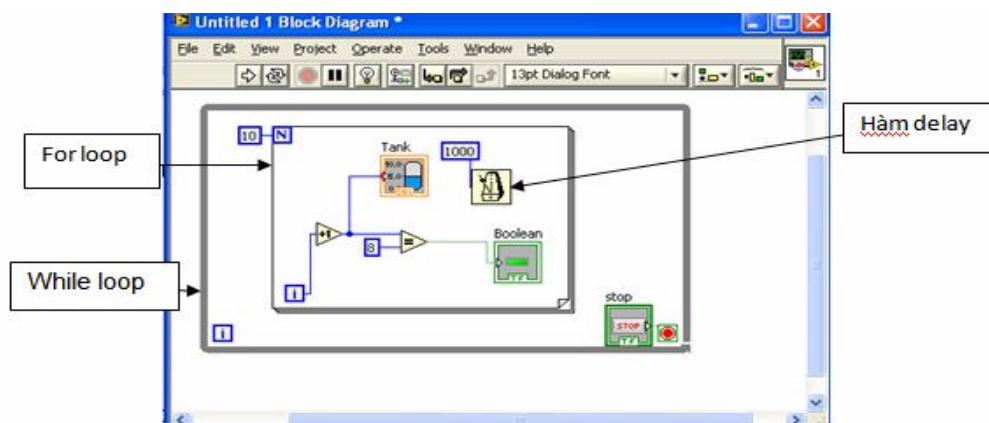
Ta chọn vòng lặp for để mô phỏng và tăng số sản phẩm

Ta vào **Functions/programming/structures/ for loop**

- Click phải chuột vào N chọn create/constan và nhập vào số vòng lặp muốn vòng lặp thực hiện ở ví dụ này chọn 10.

Các vòng lặp thực hiện rất nhanh nên để nhìn thấy kết quả ta thêm hàm delay vào và thêm hàm hàng số để định khoảng thời gian delay.

Dùng chuột ở chế độ Wring Tool nối tất cả các hàm lại với nhau. sau khi nối xong như hình dưới



**Hình 2. 72 Block diagram sau khi xây dựng xong**

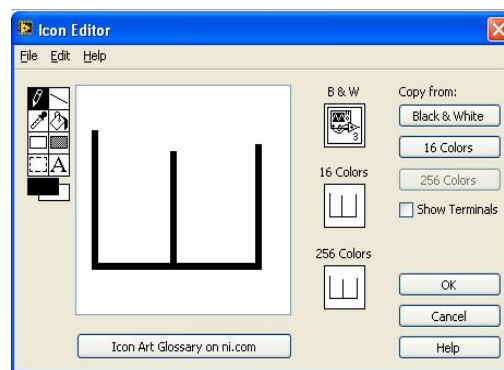
- Chọn File \ Save và lưu giữ VI với tên đếm sản phẩm
- Từ Font panel chạy VI bằng cách nhấn vào nút Run trên thanh công cụ hoặc ấn Ctrl+R.



- Đóng VI bằng cách chọn File \ Close.

### \* Xây dựng Icon của VI.

- Từ Front panel của đếm sản phẩm vi, bấm chuột phải vào Icon panel ở góc trên bên phải và chọn Edit Icon ... để vào Icon Editor, hoặc có thể bấm đúp chuột trái vào Icon panel để làm xuất hiện Icon Editor.
- Xoá Icon mặc định của VI. Dùng các công cụ ở bên trái của Icon Editor để vẽ Icon mới cho VI giống như hình.



**Hình 2. 73 Tạo Icon**


- Sau khi xây dựng xong Icon mới cho VI, bấm nút OK để đóng Icon Editor, Icon mới sẽ xuất hiện thay thế cho Icon mặc định.

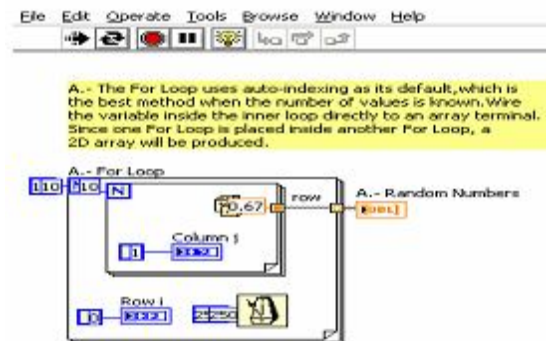
### \* Gỡ rối và sửa chương trình xây dựng trên LabVIEW:

Khi xây dựng chương trình trên LabVIEW, ta có thể tiến hành các thao tác sửa chữa, thay đổi một cách dễ dàng bằng cách thay đổi, dịch chuyển, loại bỏ các đối tượng một cách dễ dàng trên Front Panel và Block Diagram

#### - Gỡ rối chương trình:

Một chương trình không thể chạy được cũng như không thể biên dịch được khi chương trình đó còn lỗi. Khi chương trình có lỗi, nút Run sẽ xuất hiện nút gãy khi VI đang được soạn thảo và VI vẫn còn lỗi. LabVIEW cung cấp một số công cụ để theo dõi lỗi phát sinh trong quá trình lập trình. Nếu sau khi soạn thảo xong mà chương trình vẫn không chạy được thì mở hộp Error List (Window\ Show Error List) để tìm lỗi và xác định vị trí của lỗi. Nếu như khi chương trình đã chạy được nhưng kết quả không như ý muốn, ta có thể sử dụng công cụ theo dõi quá trình thực hiện chương trình, kết quả sau mỗi bước sẽ được hiện thì từ đó ta có thể xác định được thuật toán của ta có đúng hay không.

Để chọn chức năng này sau khi cho chương trình chạy, ta bấm vào nút  trên thanh công cụ của Block Diagram. Chức năng được minh họa trong hình

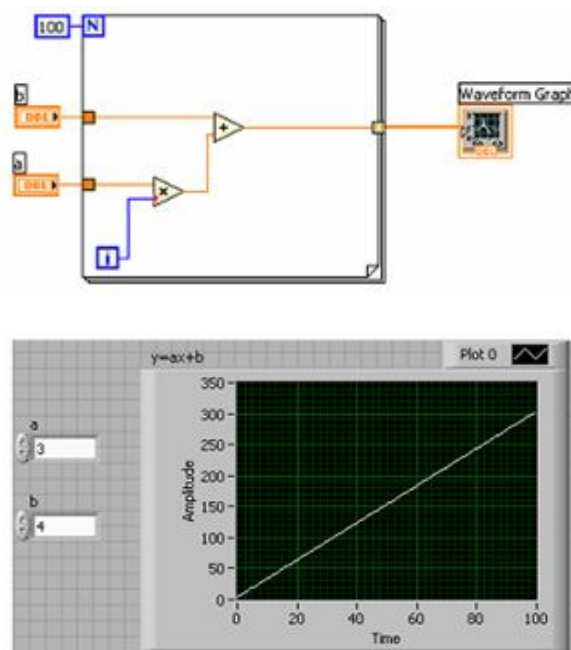


**Hình 2. 74 Kiểm tra dòng dữ liệu (kiểm tra giải thuật của chương trình)**

**Ví dụ: Vẽ đồ thị  $y = ax + b$ . Với  $a, b$  là số bất kỳ ta chọn**

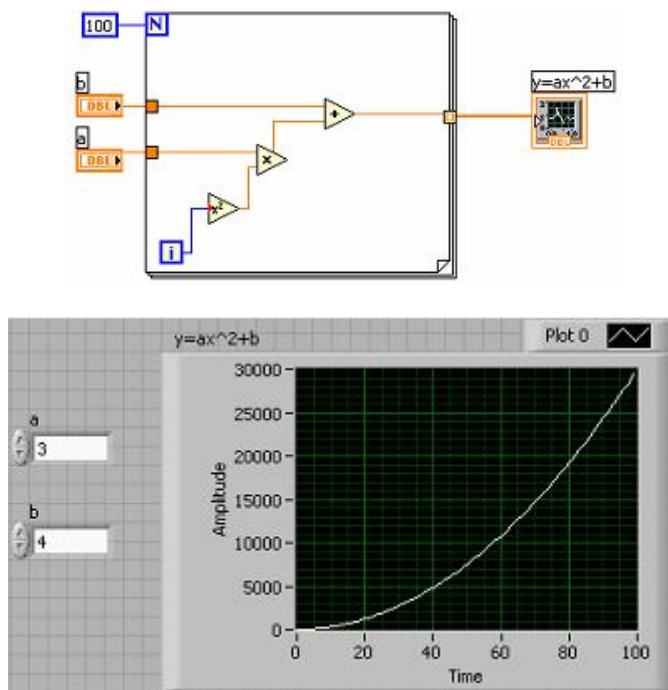
Ta xây dựng Front panel như sau:

- Lấy một đồ thị và 2 điều khiển số:



**Hình 2. 75 Front panel sau khi chạy chương trình**

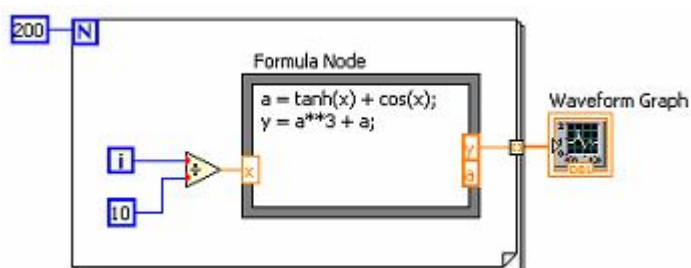
**Ví dụ: Vẽ đồ thị  $y = ax^2 + b$ .**



**Hình 2. 76 Block diagram. Front panel sau khi chạy chương trình**

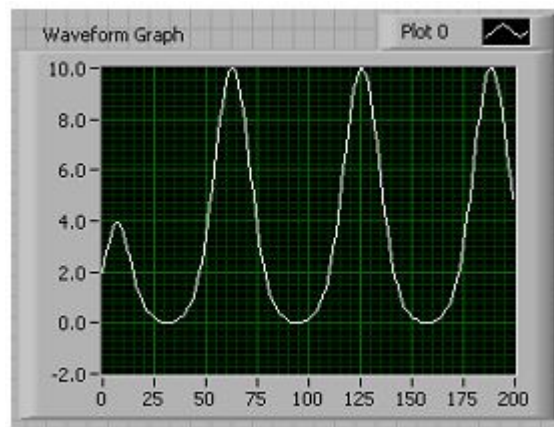
**Ví dụ : Vẽ đồ thị phương trình  $y = f(x)^3 + f(x)$ , với  $f(x) = \tanh(x) + \cos(x)$ .**

- Trong Front panel ta tạo một Waveform Graph.
- Sau đó, xây dựng block diagram như hình bên dưới. Lưu ý là ta vẫn phải tạo Output cho biến trung gian a. (ta phải nhấp phải vào đường biên và add output và đặt là a)



**Hình 2. 77 Block diagram**

- Nhấn Run, và trên đồ thị sẽ xuất hiện như sau:

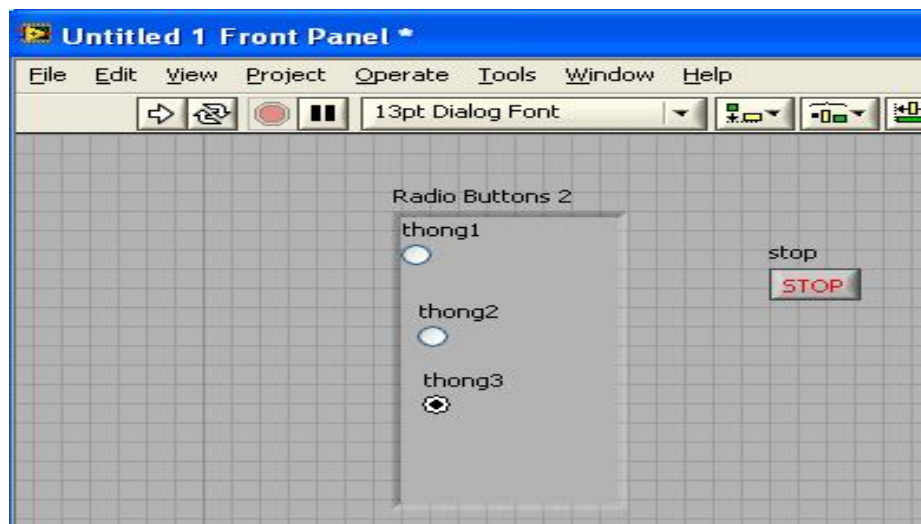


**Hình 2. 78 Khi chạy chương trình**

### Ví dụ : xuất hộp thoại thông báo chọn mục

Xây dựng font panel như hình dưới:

- Bạn lấy một radio buttons: Controls / modern / Boolean/radio buttons
- Lấy công cụ system radio button và đặt vào trong radio button ở ví dụ này sử dụng 3 system radio button : Control/classic/ classic Boolean/system radio
- Lấy 3 string để nhập nội dung cần thông báo: Control/modern/string& path/string control



**Hình 2. 79 Font panel sau khi xây dựng xong**

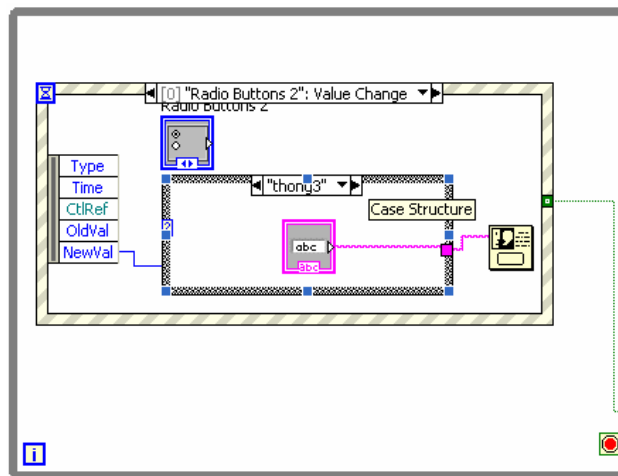
Xây dựng block diagram:

- Lấy một cấu trúc case: Function/ programming/structures/ case structure
- Lấy một cấu trúc event: Function/ programming/structures/event structure
- Lấy một cấu trúc while loop: Function/ programming/structures/while loop

- Lấy one button dialog: function/programming/dialog user interface/one button dialog. Từ cấu trúc event ta click phải chuột vào phần selecte label phía trên của cấu trúc chọn add event case để đặt sự kiện. ta chọn 2 sự kiện tác động của radio button và nút stop, và chọn sự kiện là value change. Ở sự kiện radio button ta chọn cấu trúc case để thực hiện, ở sự kiện stop ta nối nút nhấn stop vào điều kiện của vòng lặp while để dừng chương trình.

Từ cấu trúc case ta cũng click phải chuột như click ở event structure và chọn add case after sau đó nhập nội dung là tên của system radio.

Có 3 system radio nên ta phải chọn 3 điều kiện để case làm việc. và ở một điều kiện tương ứng phải đặt một string vào trong case với nội dung cần hiển thị và nối dây ra từ case vào one button dialog. Và dùng công cụ nối dây để nối dây của các hàm lại. Sau khi hoàn tất block diagram như hình sau:



**Hình 2. 80 Block sau khi xây dựng xong**



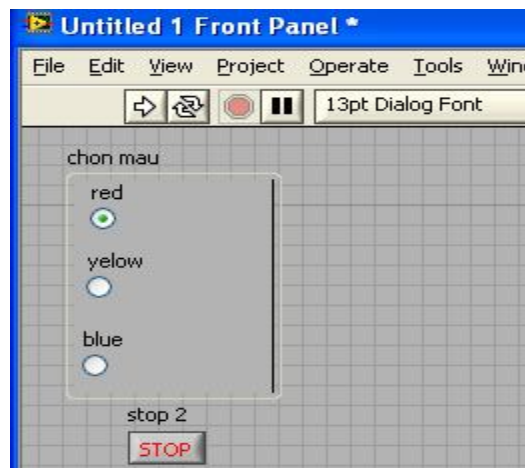
**Hình 2. 81 Font panel khi chạy chương trình**

Khi đã hoàn tất, chạy chương trình và dùng chuột chọn vào bất kì ô nào thì hộp thoại sẽ xuất hiện với nội dung đã định sẵn như hình trên.

### Ví dụ: thay đổi màu của font panel:

#### 1. Xây dựng Font panel:

- Dùng một classic radio buttons: Control/classic/classic Boolean/classic radio buttons
- System radio button: Control/classic/classic Boolean/classic radio button. Đặt system radio button vào trong classic radio buttons và đặt tên cho mỗi system radio button. Sau khi xây dựng xong font panel như sau:



**Hình 2. 82 Font panel sau khi xây dựng xong**

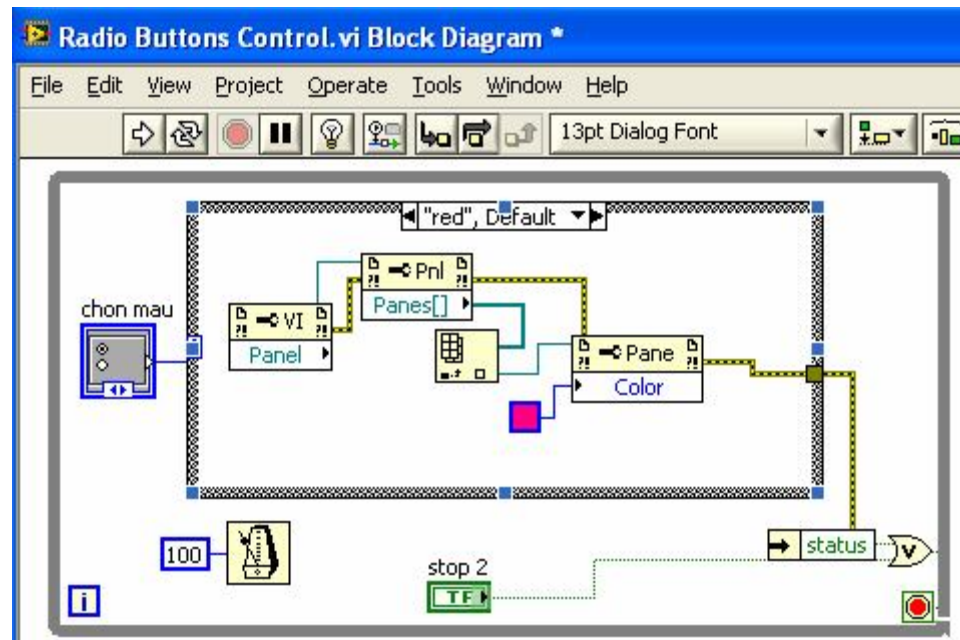
#### 2. Xây dựng block diagram:

- Case structure: Functions/programming/structure/case structure
- While loop: Functions/programming/structure/while loop.
- Index array: Functions/programming/array/ index array.
- Color box constant: Functions/programming/dialog & user interface/color box constant
- Unbundle by name: Functions/programming/cluster, class &varian/ unbendle by name.
- Or: : Functions/ Express/Boolean/ or.
- Property node: : Functions/programming/application/property node. Sử dụng 3 hàm property node. Mỗi property chúng ta chọn mỗi class và property khác nhau.

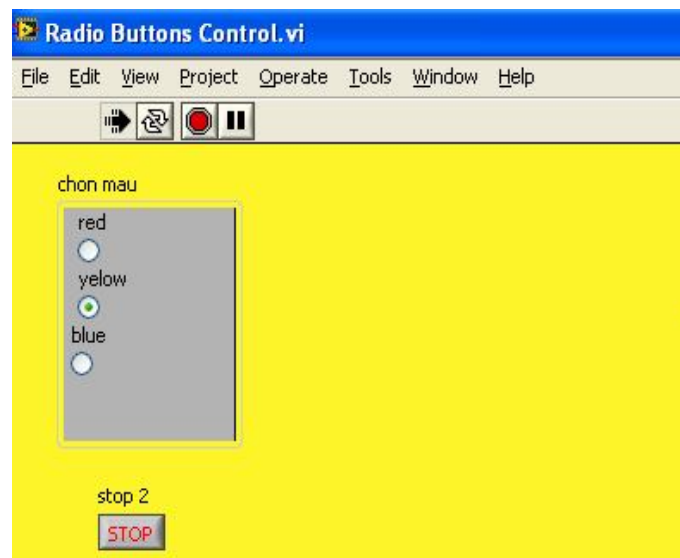
Property node thứ nhất chọn lass là VI và property là font panel. Property node thứ 2 chọn lass là panel và property là panes[]. Property node thứ 3 chọn lass là pane và property là color.

Trong case structure ta click phải vào selecte label và chọn add case after và nhập màu cần hiện thị ở mỗi điều kiện có 3 hàm property như đã nói ở trên và một color box với

một màu riêng. Dùng công cụ nối dây để nối các hàm. sau khi xây dựng xong block như hình sau:



Hình 2. 83 Block sau khi đã xây dựng xong



Hình 2. 84 Font panel sau khi chạy chương trình

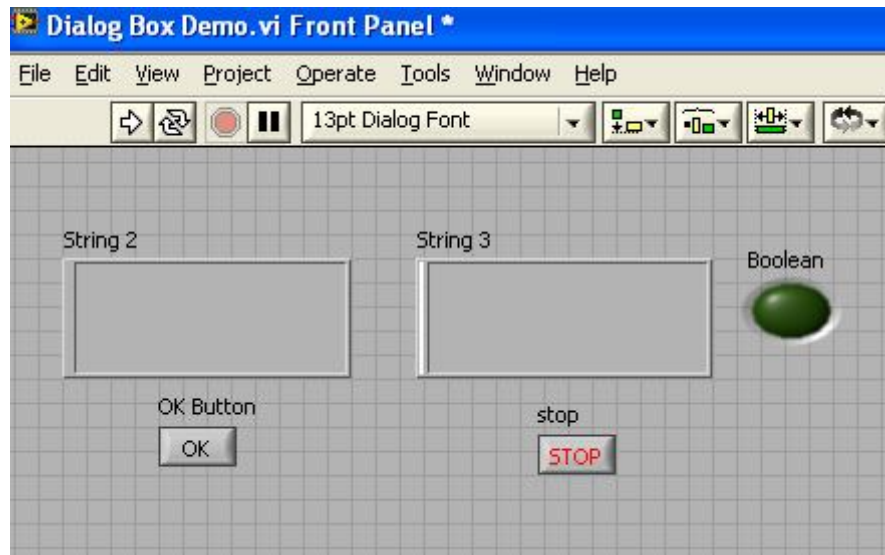
Sau khi hoàn thành ta nhấn nút chạy trên thanh công cụ và chọn yellow ta sẽ thấy font panel thay đổi tương ứng với màu đã chọn.

**Ví dụ xuất hộp thoại và hiển thị nội dung nhận được**

**Xây dựng Font panel:**



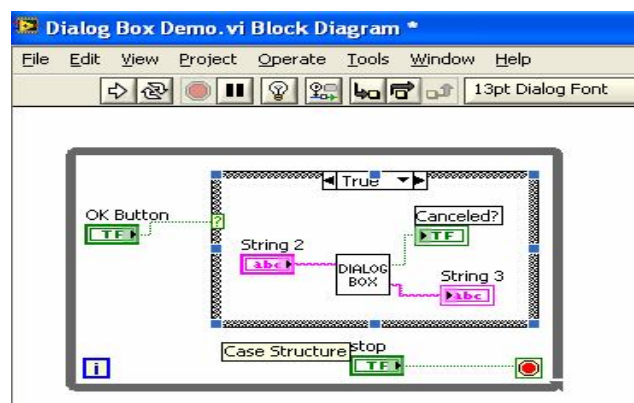
Lấy string control và string indicator: Controls/ classic/ classic string & path, đặt lên Font panel.  
Lấy một nút ok. Một led và nút stop: Controls/express/Boolean & swiches sau khi xây dựng xong Font panel như hình sau:



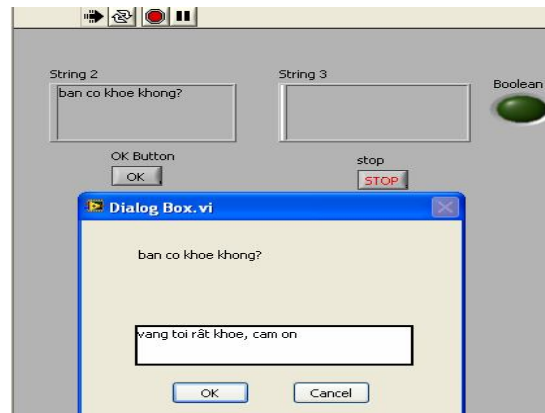
Hình 2. 85 Font panel sau khi xây dựng

### Xây dựng Block diagram:

Ta sử dụng một hàm case và hàm while loop. Lấy 2 hàm này xin xem lại ví dụ trước. trong chương trình này ta sử dụng một chương trình con (sub VI). Ví dụ này sử dụng sub VI có sẵn trong labview. Function/ selecte a VI/ ổ đĩa c/ program/instruments/labview 8. 6/ examples/general/viopts/dialogbox.vi. Dùng công cụ nối dây nối các hàm lại với nhau. Sau khi hoàn thành như hình sau:



Hình 2. 86 Block diagram sau khi xây dựng xong



**Hình 2. 87 khi chạy chương trình**

Khi nhấn nút chạy ta nhập nội dung trong string2 và bấm nút ok và nhập nội dung đáp ứng vào hộp thoại và nhấn nút ok của hộp thoại để ghi nội dung vào trong string3.

Trong labview có rất nhiều ví dụ sử dụng các hàm. chúng ta có thể mở và tham khảo và làm theo. Khi muốn mở những ví dụ này ta vào help/ Find examples. Sau đó vào search và gõ từ khóa muốn tìm để tìm và mở ví dụ. chúng ta muốn biết chức năng của hàm nào đó chúng ta vào help / show context help và di chuyển con trỏ chuột trên hàm đó để đọc thông tin. đặc biệt labview có một số tài liệu hướng dẫn sử dụng labview và các chức năng khác. Tài liệu này nằm trong labview manuals. Để vào labview manuals ta vào star/ all program/ national instruments/ labview 8. 6/ labview manuals