

實驗七：函數波形產生器編程控制與應用

前言

本實驗的主要目的是透過電腦編程來控制函數波形產生器，讓參與者學習如何在電腦上自行建立人機介面 (Human-Machine Interface, HMI)，並與函數波形產生器進行有效的溝通與互動。在本實驗中，我們選用了 C#(讀作”C-sharp”) 作為開發語言，因為其強大的功能、靈活的介面設計能力以及與各種硬體設備的良好兼容性，使其成為構建此類應用程序的理想選擇。

在實驗過程中，參與者將學習如何使用 C#編寫程式碼來控制函數波形產生器的各種操作，包括波形類型的選擇、頻率的設定、振幅的調整等，透過這項實驗，不僅可以加深參與者對 C#編程的理解，還能夠掌握如何將軟體技術應用於實際的硬體控制中，達到編程與實踐相結合的效果。

(詳細函數波形產生器之介紹詳見實驗一：儀器介紹與基本操作 P.12-15)

(本實驗包含一練習作業、實驗內容一至三)

背景

透過 NI (National Instruments) VISA (Virtual Instrument Software Architecture)的動態連結檔 (DLL)，並在 C#中自行建立人機介面，主要原因在於 VISA 提供了統一的界面和標準，使得不同廠商的儀器(如此次函數波形產生器為 Rigol 所屬)能夠透過相同的 API 進行控制和通訊，以下是其中的幾個關鍵因素

VISA 是一個標準化的軟體架構，專為控制各類儀器設計。這些儀器可能通過 USB、GPIB 等方式連接到電腦，NI VISA 提供了一個統一的介面，使不同廠商的儀器能夠使用相同的函數庫進行控制，這解決了不同硬體設備之間的兼容性問題。

NI VISA 提供的動態連結檔 (如此次會用到的 NationalInstruments.VisaNS.dll) 包含了所有必要的函數，讓使用者能夠在 C#中調用這些函數來與儀器進行通訊。這種 DLL 的設計使得程式開發者不必了解儀器底層的通訊細節，而是可以直接透過高階函數來操作儀器。

C#是一種功能強大且靈活的編程語言，特別適合用來開發具有圖形化用戶介面(GUI, Graphical User Interface)的應用程式。C#提供了豐富的工具，讓開發者可以輕鬆地設計和實現人機介面，並透過 NI VISA 與儀器進行交互。

《下載開發工具 Visual Studio》

下載開發工具 Visual Studio(至官網下載 community 社群版本，如下圖 1 所示)，其是微軟提供的強大整合開發環境，專為 C# 和 .NET 開發量身訂製，它提供了除錯、編譯、測試等一系列的功能，極大地提升了開發效率，而 C#非常適合開發 Windows 桌面應用程式，也就是我們前述所說的人機介面。



圖 1、Visual Studio download

下載完成後點開 Setup.exe 請選擇.NET 桌面開發再選擇安裝



圖 2、Visual Studio download

下載完成後即可點開 Visual Studio 2022 (以此版本為例) 並啟動，選擇建立新的專案，接著下拉選單選擇使用 C#進行開發，平台於 Windows，應用於桌面，找到 Windows Forms App (.NET framework) 並選擇下一步，訂定專案名稱、位置，以及所需之架構(.NET Framework 4.8)，最後選擇建立。

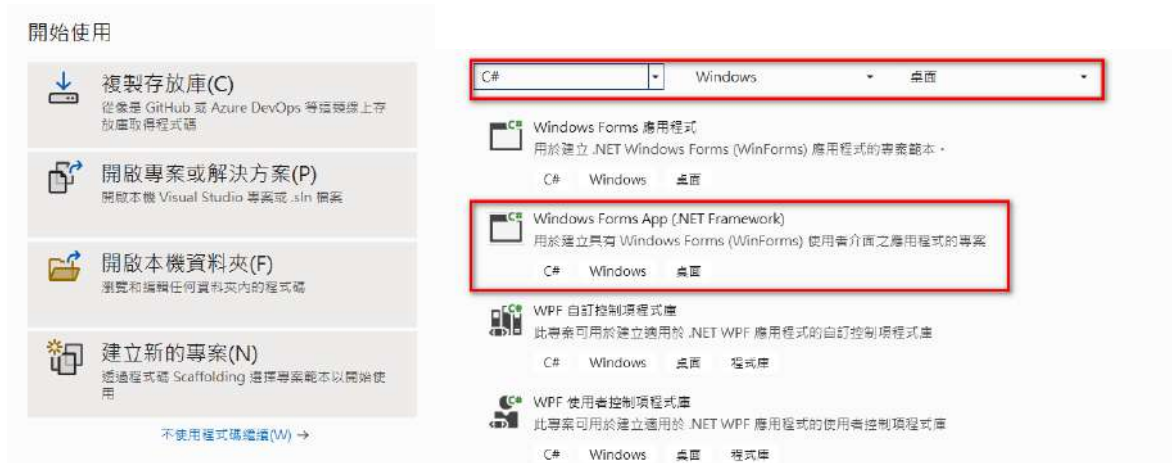


圖 3、Visual Studio 建立新專案

打開專案，出現以下視窗：

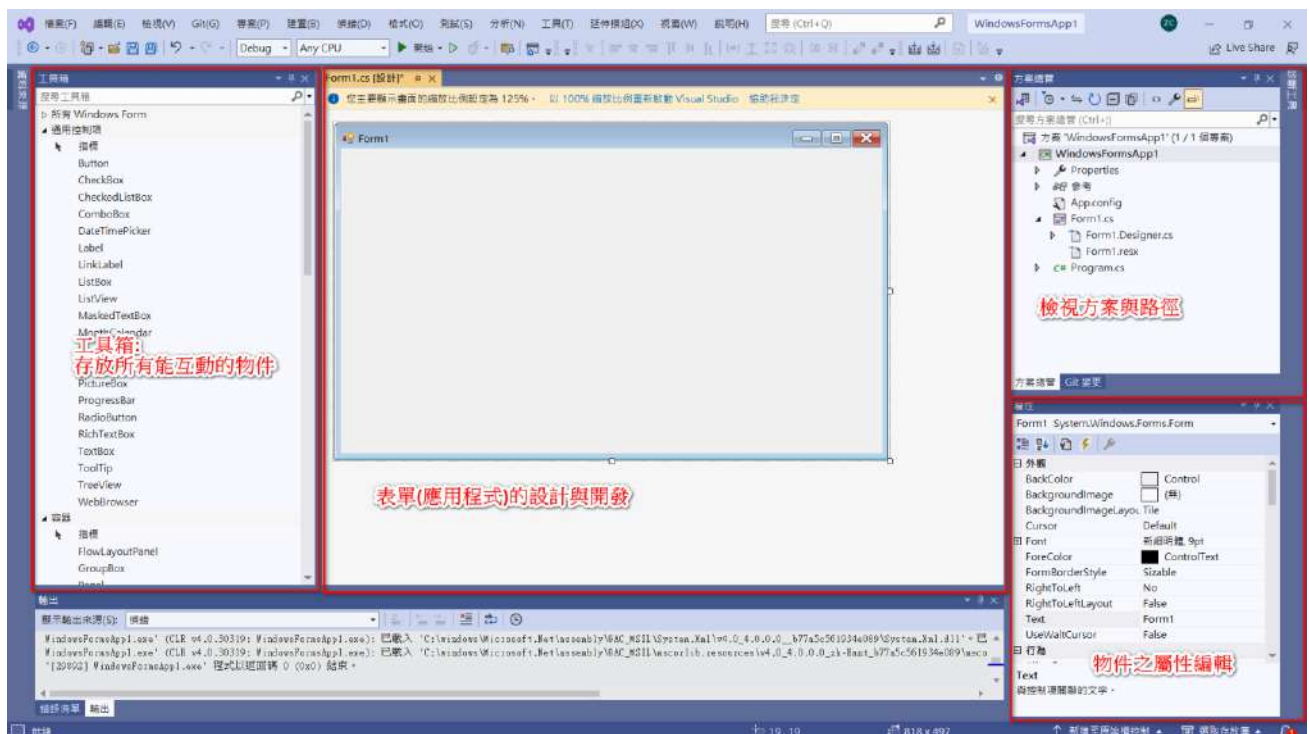


圖 4、Visual Studio 開發人機介面

在工具箱中，有常見的物件如按鈕、標籤、字串盒，可拖拉進 Form1 的表單視窗中，且可隨意更改其物件大小，至右下方屬性窗格更改其物件的屬性（此物件之名稱、顏色、字體等）。

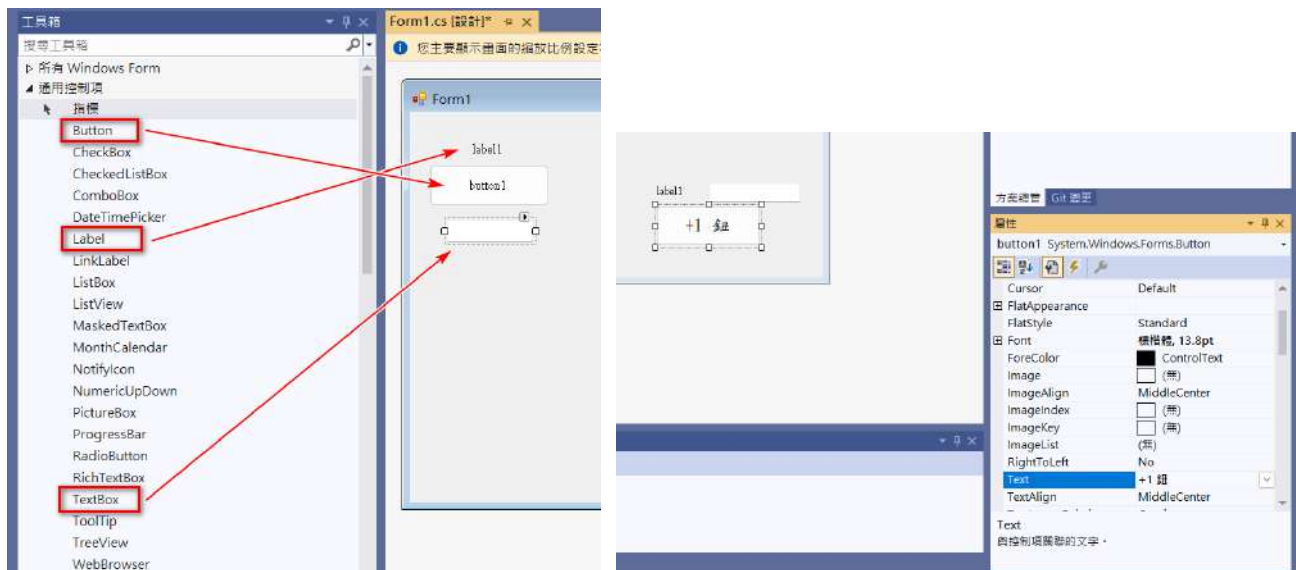


圖 5、C# 物件應用

可先點選開始鈕，表單 Form1 會跳出視窗成為應用程式，如下圖 7 所示。至目前為止此 Form1 可稱為最簡單功能的人機介面應用程式（未賦予任何功能），將應用程式關閉即可跳轉回開發環境中。



圖 6、C# 應用程式試啟動

《製作簡易應用程式教學》

我們寫的應用程式形成一個 Form，要顯示文字或數字就用 Label 物件，按鈕則是 Button 物件，如上圖 6 所示，在 C# 當中編寫程式基本上就是在安排這些物件的行為，包括位置、外觀、功能等，因此編寫視窗應用程式的設計流程大致上可分為：

1. 確認程式功能
2. 設計視窗(表單)畫面
3. 拖拉建立物件(Label、Button...)
4. 編寫程式(敲鍵程式碼)
5. 執行

接著做簡易的應用程式練習-加減法。首先拖拉 1 個 label 與 3 個 button 至表單 form1 中，(拖拉的順序即會決定該物件的順序) 並在表單中調整適當的大小，以及在右側屬性選單中在 text 處可更改名稱；在 Font 處可調整其字體、字體大小，完成後如下圖 7 所示。(請注意點選表單中的物件時滑鼠點一下再至右側屬性選單編輯即可，若雙擊物件會跳轉至該物件的程式編輯介面，若不慎點進程式編輯區，點回上面選單”Form1.cs[設計]”即可)

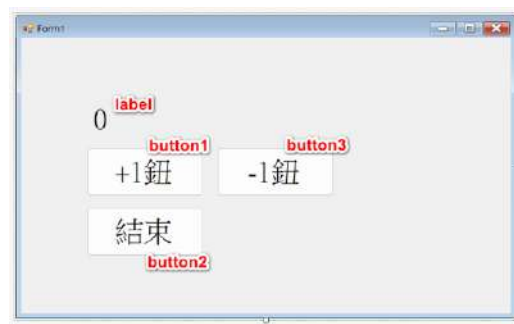


圖 7、表單建立加減法標籤與按鈕



建立好物件介面，接著開始賦予此表單應用程式的功能，因此會在物件內編寫程式，對建立好的+1 鈕點兩下進入此按鈕的編程介面，畫面會挑轉至該按鈕的程式編輯區當中。

private 表示這個宣告方法的存取範圍是私有的，也就是說這個宣告只能在它所在的類別被呼叫，而不能在類別外部被存取；void 代表這個方法沒有返回值，執行完畢後不會返回任何資料；button1_Click 通常表示這個方法是用來處理 button1 的點擊事件，而點擊這個按鈕後需要發生什麼事件，則要我們將相對應的行為輸入進去。

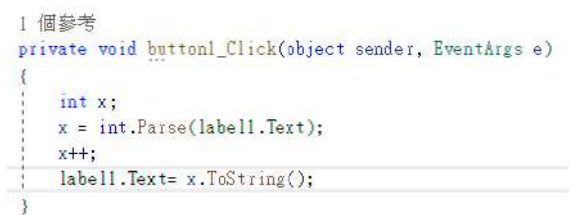
輸入後按開始，執行程式碼確認+1 鈕按下去之後，標籤是否有做加法的動作。



```
34
35
36
37
38
39
40
41
```

1 個參考
private void button1_Click(object sender, EventArgs e)
{
}
}

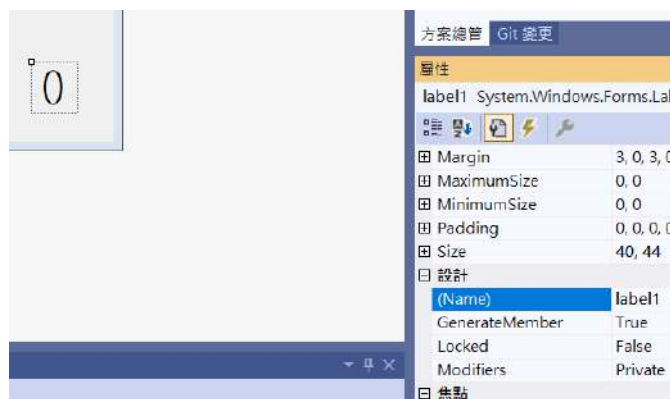
圖 9、進入+1 鈕(button1)程式編輯初始介面



```
1 個參考  
private void button1_Click(object sender, EventArgs e)  
{  
    int x;  
    x = int.Parse(label1.Text);  
    x++;  
    label1.Text= x.ToString();  
}
```

先宣告變數，將字串值轉為變數值
執行累加
再將變數值轉為字串值

圖 10、進入+1 鈕(button1)程式編輯後介面



在屬性窗格當中，下拉找到設計中的(name)，可以清楚的知道目前該物件在編程介面所代表的名稱。例如在開發環境中我將這個label的text更改為0，但是在編程介面它識別依然還是label1

圖 11、程式編輯介面須知

接著+1 鈕做完之後，可以接著做-1 鈕以及結束鈕，分別再開發環境中雙擊-1 鈕與結束鈕建立各自的點擊事件，在裡面新增其行為的程式碼，如下圖 12 所示，完成後點選上面選單開始測試表單應用程式功能。

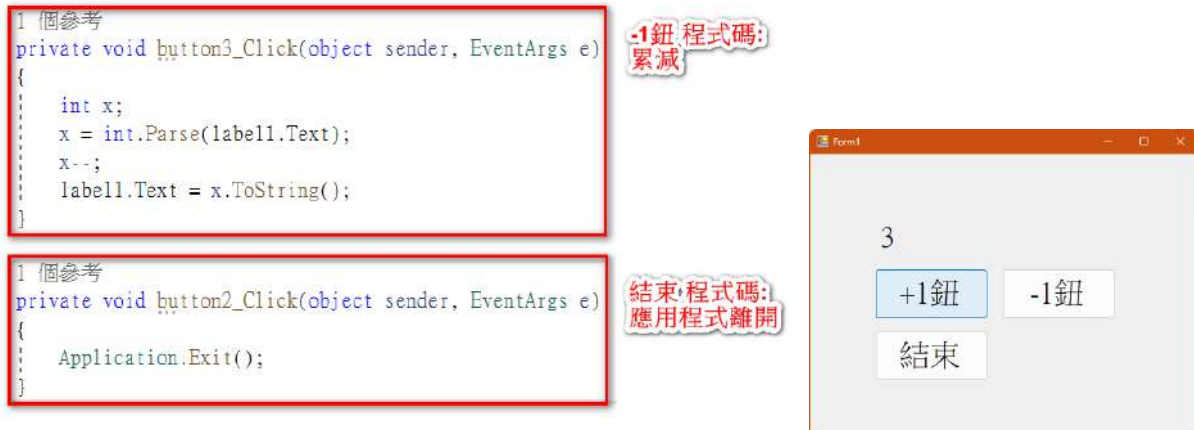


圖 12、加減法應用程式完成

《練習作業》

至目前為止同學們應已可以了解 C#開發人機介面基本操作，本實驗即為使用 C#編程控制機器：使用 USB 連接線連接電腦與函數波形產生器，並用人機介面搭配產波器編程手冊控制函數波形產生器。以下練習作業，希望同學能夠更熟悉 C#設計與開發，並將你開發的過程一同寫入預報當中：

1. 前述已教學如何建立簡易的加減法應用程式，因此練習題請做出能夠輸入一整數 x 與一整數 y ，並可以對 x 、 y 互相進行加減乘除，且答案能夠出現小數點，點擊結束鈕可關閉應用程式。

(已知會用到的物件為 text box、label、button)

(參考資料可參考：

<https://www.youtube.com/watch?v=qKEFE5hfu-Q&list=PLGLZVd2c1WxdpDpjxqyeSjjYSwfeEbTkU&index=5>

https://www.youtube.com/watch?v=zzJIWC_HFU4&list=PLGLZVd2c1WxdpDpjxqyeSjjYSwfeEbTkU&index=6

)

《實驗內容前置(於電子電路實驗室)》

建立編程環境

首先到 NI 官網下載 Measurement Studio (如下圖 13 所示，注意支援的作業系統與版本相容問題)，其是一個專為 Microsoft Visual Studio 設計的附加工具，旨在幫助開發工程應用軟體，尤其是在 C# 中進行資料擷取、測試與儀器控制等應用，當我們安裝 NI Studio 時會自動包含一些常用的 DLL，其中就有我們要的 National Instruments.Common.dll，下載完成這些 DLL 文建會自動配置在正確的路徑下。

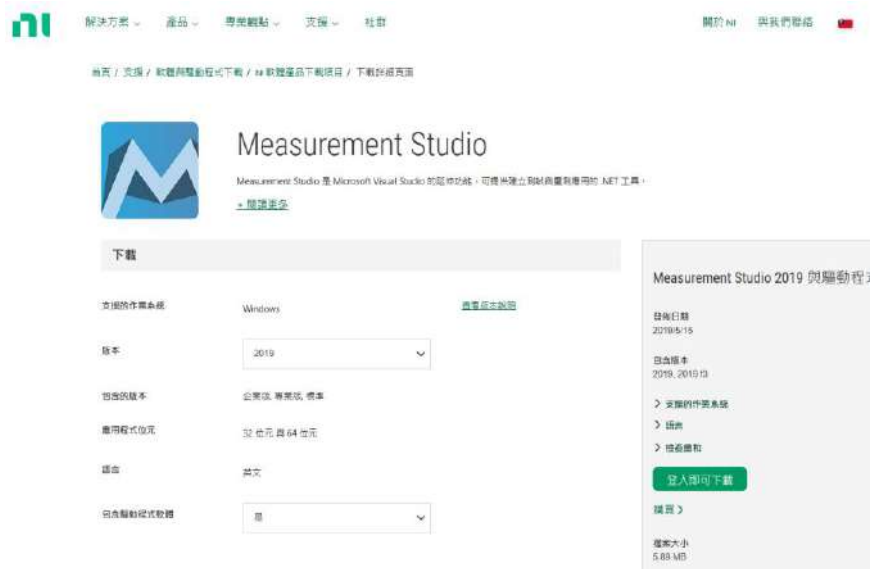


圖 13、NI Studio download

接著此次實驗會附上 NationalInstruments.VisaNS.dll 的動態連結檔，會一同附在資料夾當中。下載好 NI Studio 後，打開 Visual Studio (未下載的電腦請先下載，或者用自己的電腦)，建立新專案，如下圖 14 所示。



圖 14、Visual Studio 建立新專案

接著請在方案總管這裡在參考處添加 NationalInstruments.VisaNS.dll 的動態連結檔，流程如下圖 15 所示。

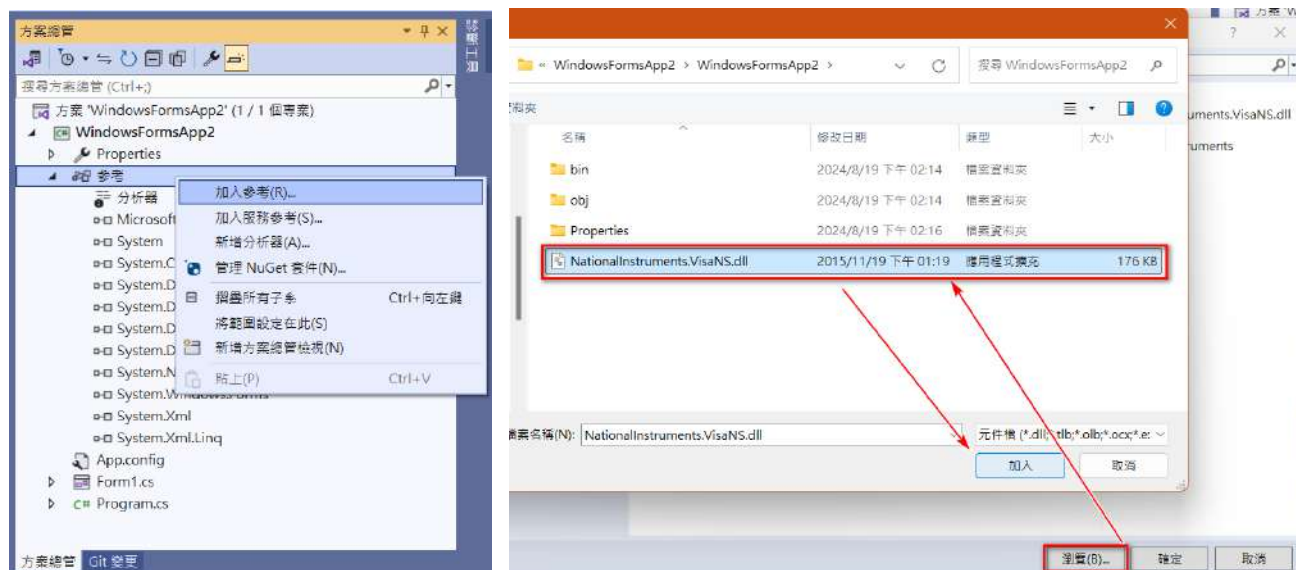


圖 15、參考路徑添加動態連結檔

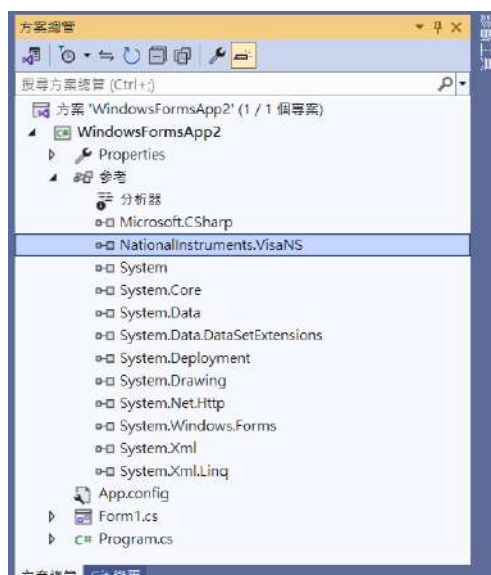


圖 16、路徑添加完成

接著請參考資料夾中 C_NET General Communication.pdf 的簡易溝通應用程式，並自行建立出來。

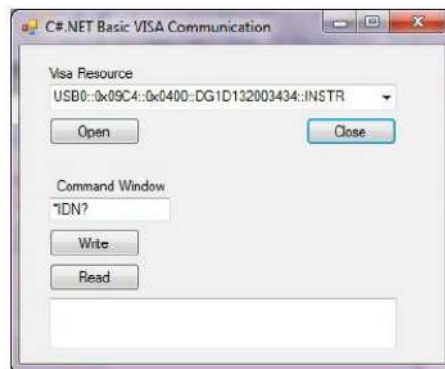


圖 17、Communication.pdf 內的範例程式

首先在專案內的表單拖拉進相關物件，並在屬性 text 更改其名稱（須注意屬性下方設計處 (name) 程式碼識別名稱）如下圖 18 所示，這樣已經完成初步的簡易溝通人機介面，接著我們希望各個按鈕以及文字盒輸入後都能有其實際功能，因此在建立按鈕當前需確認我們需要有什麼樣的功能，以這次的產波器簡易溝通應用程式為例：

1. Textbox1：輸入函數波形產生器之型號(UID)。
2. Button1(Open)：按下之後，偵測並接收 textbox1 之 ID 文字至波形產生器。
3. Textbox2：輸入與產波器溝通之指令。
4. Button3：按下之後，偵測並接收 Textbox2 之指令文字至波形產生器。
5. Button4：按下之後，波形產生器接收 Textbox2 之指令文字並回傳反饋值給電腦，且顯示其回饋值資訊至 textbox3 當中。
6. Close：按下之後，結束應用程式。

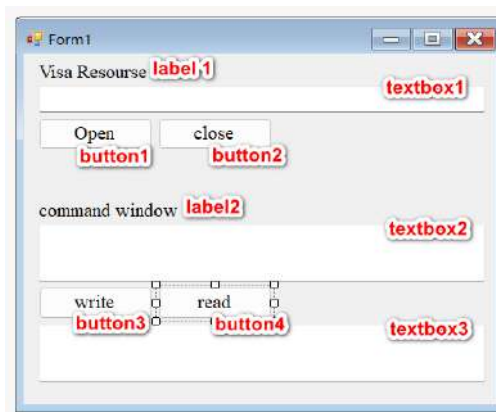


圖 18、在自己開發環境中建立相關物件

首先雙擊表單 form1 進入表單的程式編輯介面，並在 form1 底下輸入此行程式碼，如下圖 19 所示。MessageBasedSession 是 National Instruments VISA 函式庫中的一個類別，用於與儀器進行基於消息的通信。這個類別提供了與儀器之間傳送和接收命令、數據的功能，通常用於控制像是示波器、信號產生器、萬用表等儀器；mbSession 這是聲明的變數名稱，它將用來引用 MessageBasedSession 類型的物件。

```
8   using System.Linq;
9   using System.Threading.Tasks;
10  using System.Windows.Forms;
11  using static System.Windows.Forms.VisualStyles.VisualStyleElement;
12
13  namespace WindowsFormsApp2
14  {
15      3 個參考
16      public partial class Form1 : Form
17      {
18          private MessageBasedSession mbSession; //create Message based session
19
20          1 個參考
21          public Form1()
22          {
23              InitializeComponent();
24          }
25      }
26  }
```

圖 19、先建立 mbSession 變數

再來各別去編輯各個物件的程式功能，我們首要目的是要將 textbox1 的字串值(UUID)接收並執行產波器啟動的動作，雙擊 open 鈕進入其程式編輯介面，將以下程式碼（紅框處）輸入進去（注意大小寫，需輸入正確）。

```
1 個參考
private void button1_Click(object sender, EventArgs e)
{
    string strVISARsrc = textBox1.Text; //Get VISA resource ID from combobox
    try
    {
        mbSession = (MessageBasedSession)ResourceManager.GetLocalManager().Open(strVISARsrc);
        //Instantiate and open a Message Based VISA Session
    }
    catch (InvalidCastException)
    {
        MessageBox.Show("Resource selected must be a message-based session");
    }
    catch (Exception exp)
    {
        MessageBox.Show(exp.Message);
    }
}
```

圖 20、Open 鈕 (button1)程式碼

雙擊 close 鈕進入其程式編輯介面，將以下程式碼（紅框處）輸入進去（注意大小寫，需輸入正確）。

```
1 個參考
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        mbSession.Dispose(); // 作用是釋放mbSession佔用的資源，Dispose方法通常用於清理未管理資源（如打開的文件、網路連接、儀器會話等）。
    }
    catch (Exception exp)
    {
        MessageBox.Show(exp.Message);
    }
    Application.Exit(); // 離開應用程式
}
```

圖 21、close 鈕 (button2)程式碼

我們期望將 textbox2 的字串值(指令)接收並寫入產波器當中，接著雙擊 write 鈕進入其程式編輯介面，將以下程式碼（紅框處）輸入進去（輸入大小寫，需輸入正確），基本上輸入指令之後按 write 按鈕就可以寫入產波器，而產波器也會啟用相對應的指令功能，這部分稍後可做測試。

```
1 個參考
private void button3_Click(object sender, EventArgs e)
{
    string strWrite = textBox2.Text; //Get command from text box
    try
    {
        mbSession.Write(strWrite); //Send Write command to VISA resource
    }
    catch (Exception exp)
    {
        MessageBox.Show(exp.Message); //檢視有無例外異常
    }
}
```

圖 22、write 鈕 (button3)程式碼

有些指令值在產波器接收之後，是會有反饋值的，這時候需按下 read 鈕，將反饋值顯示在 textbox3 當中，這時我們才好做下一步的判斷。接著雙擊 read 鈕進入其程式編輯介面，將以下程式碼（紅框處）輸入進去（輸入大小寫，需輸入正確）。

```
1 個參考
private void button4_Click(object sender, EventArgs e)
{
    string strRead = null; //宣告了一個名為strRead的字串變數並初始化為null
    try
    {
        strRead = mbSession.ReadString(); //Read string from VISA resource
        textBox3.Text = strRead; //指派給textbox3
    }
    catch (Exception exp) //檢視有無例外異常
    {
        MessageBox.Show(exp.Message);
    }
}
```

圖 23、read 鈕 (button4)程式碼

至此我們已經將產波器的簡易溝通應用程式建立完畢，接著可以使用 USB 連接線（如下圖 24 所示）將產波器與電腦連接，這意味著接下來操作上，會以電腦這邊的應用程式來控制函數波形產生器。



圖 24、USB 連接線

連接好後回到 C#的 Form1.cs[設計]介面，在上面選單點開始執行程式，接著 Visa Resource 這裡會要求你將這台產波器的 UID 輸入進去，函數產生器之型號（ID name），請回到產波器按 Utility 鈕 → 接口(or I/O)，找到其 ID 後輸入至人機介面的 Visa Resource 窗格。



圖 25、函數波形產生器之 ID name

爾後，點應用程式的 open 鈕啟動產波器，接下來我們想知道電腦是否有抓到這台儀器，因此會開始在 command window 中下達指令（在本次實驗資料夾當中，會一同附上本機台的 DG1000 編程手冊，裡面集合了所有能對函數波形產生器下達的指令(指令都很短、很簡單))，因此翻閱編程手冊 P2-2，將這行指令打在 command window 中，並按下 write 以及 read 鈕，如下圖 26 所示。

1. *IDN?	
命令格式	*IDN?
功能描述	查詢儀器 ID 字符串
返回值	查詢返回 4 個由逗號 “,” 分隔的字段：製造商,型號,序列號,由點號分隔并由數字組成的版本号。如： RIGOL TECHNOLOGIES,DG1022,DG1D100,00.02.00.06.00.02.06

圖 26、識別 ID 指令

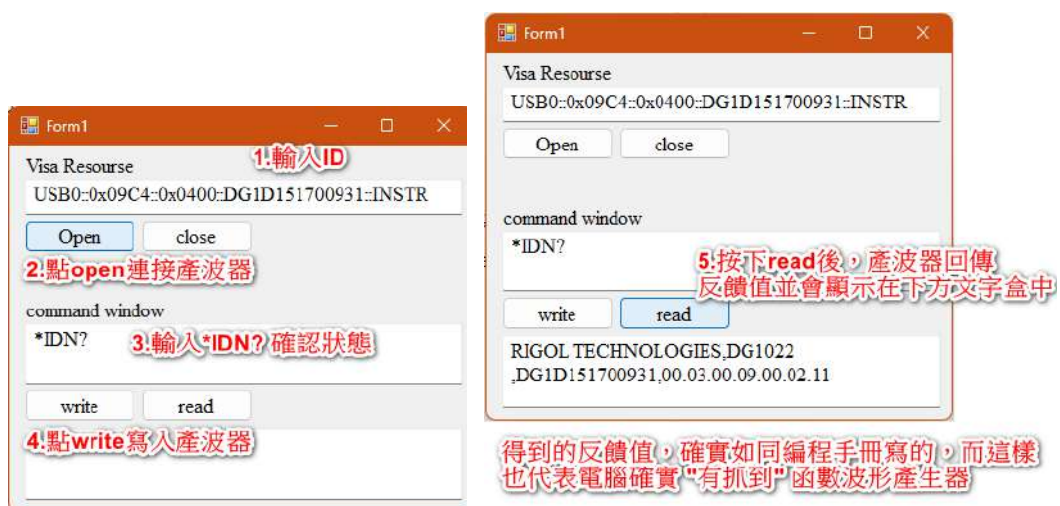


圖 27、識別 ID

《實驗內容一》

至目前為止我們已經會運用電腦上自製的簡易溝通應用程式與波形產生器做溝通，甚至是下達簡單的指令給產波器，接著以下為實驗內容一：

1. 請將產波器連上 BNC 纜線連接至示波器，接著請自行翻閱 DG1000 編程手冊，透過電腦 command window 處下達指令，由產波器給予示波器 Sine 波、頻率 1 kHz、5Vpp、0 Offset，請將需要下達的指令及示波器畫面寫入結報中。
2. 新增一條 BNC 纜線至 CH2，透過電腦 command window 處下達指令，由產波器給予示波器 CH2 頻道 Square 波、頻率 1500 kHz、4Vpp、0 Offset，請將需要下達的指令及示波器畫面寫入結報中。

《實驗內容二》

目前為止我們做的只是簡易的溝通器，對波形產生器下達指令需在 command window 一行一行地寫入才能告訴產波器我們的需求，十分耗費時間，因此實驗內容二：

1. 請翻閱 DG1000 編程手冊，要求同學改寫此應用程式，期望操作者能直接輸入頻率、Amp、Offset 的數值並”按下”write 鈕即能給予函數波形產生器指令，不需一行一行寫入，並且可以任意選擇期望的波形。撰寫完畢且測試成功後，請將你們這組的自製 APP 樣貌如下圖 28 一樣截圖，以及擷取示波器當前畫面寫入結報中。

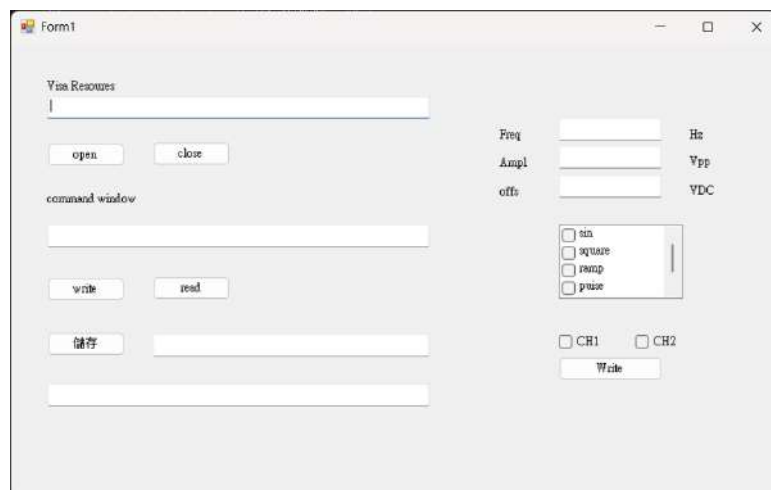


圖 28、實驗內容二目標

(hint：在 Form1 將範圍拉大足夠畫面，拖曳所需 label、textbox、button、checkboxlistbox、checkbox 數量並自行排版，write 鈕必須能集合頻率、振幅、Offset 的文字盒中的數值並進行寫入，Frequency 指令格式請參考 P2-13；Voltage 指令格式請參考 P2-16)

<code hint>

```
private CheckedListBox GetCheckedListBox1()
{
    return checkedListBox1;
}
// 取得使用者選擇的通道
private string GetSelectedChannel()
{
    if (checkBox1.Checked && !checkBox2.Checked) return "SOUR1";
    if (checkBox2.Checked && !checkBox1.Checked) return "SOUR2";
    MessageBox.Show("請選擇一個通道 (CH1 或 CH2)", "錯誤",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
    return null;
}

private string GetSelectedWaveform()
{
    if (checkedListBox1.CheckedItems.Count == 0)
    {
        MessageBox.Show("請選擇一個波形", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return null;
    }

    Dictionary<string, string> waveformMap = new Dictionary<string, string>()
    {
        {"sin", "SIN"},
        {"square", "SQU"},
        {"ramp", "RAMP"},
        {"pulse", "PULS"},
        {"noise", "NOIS"}
    };
};
```

```

string selected = checkedListBox1.CheckedItems[0].ToString().ToLower();
if (waveformMap.ContainsKey(selected))
    return waveformMap[selected];

MessageBox.Show("選擇的波形不支援", "錯誤",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
return null;
}

private (string freq, string ampl, string offs) GetParameters()
{
    string freq = textBox5.Text.Trim();
    string ampl = textBox6.Text.Trim();
    string offs = textBox7.Text.Trim();

    if (string.IsNullOrEmpty(freq) || string.IsNullOrEmpty(ampl) || string.IsNullOrEmpty(offs))
    {
        MessageBox.Show("請輸入頻率、振幅與偏壓", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return (null, null, null);
    }

    // 驗證數值格式
    if (!double.TryParse(freq, out double freqValue) || freqValue <= 0)
    {
        MessageBox.Show("頻率必須是有效的正數", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return (null, null, null);
    }

    if (!double.TryParse(ampl, out double amplValue) || amplValue <= 0)
    {
        MessageBox.Show("振幅必須是有效的正數", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return (null, null, null);
    }

    if (!double.TryParse(offs, out _))

```

```

    {
        MessageBox.Show("偏壓必須是有效的數字", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return (null, null, null);
    }

    return (freq, ampl, offs);
}

private void SendSCPI(string channel, string waveform, string freq, string ampl, string offs)
{
    if (mbSession == null)
    {
        MessageBox.Show("請先連接到儀器", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    try
    {
        mbSession.Write($"{channel}:FUNC {waveform}");
        mbSession.Write($"{channel}:FREQ {freq}");
        mbSession.Write($"{channel}:VOLT {ampl}");
        mbSession.Write($"{channel}:VOLT:OFFS {offs}");
        mbSession.Write($"{channel}:OUTP{channel.Substring(4)} ON");
        MessageBox.Show(
            $"已設定 {channel} 輸出 {waveform} 波形\n" +
            $"頻率={freq} Hz, 振幅={ampl} Vpp, 偏壓={offs} V",
            "設定完成", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"錯誤: {ex.Message}", "錯誤",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void button6_Click(object sender, EventArgs e)

```

```
{  
    string channel = GetSelectedChannel();  
    if (channel == null) return;  
  
    string waveform = GetSelectedWaveform();  
    if (waveform == null) return;  
  
    var parameters = GetParameters();  
    if (parameters.freq == null) return;  
  
    SendSCPI(channel, waveform, parameters.freq, parameters.ampl, parameters.off);  
}
```

《實驗內容三》

相比範例的簡易 APP，我們在實驗內容二已經完成與產波器溝通的自定義 APP，提升了不少功能，接下來實驗內容三會進階至任意波形的編輯（手動操作產波器之任意波形功能請詳閱此講義後方附錄一），請參閱編程手冊 P3-4：

1. 期望能有個按鈕，點下去之後能切換至 Arbitrary 任意波形功能，接著輸入任意波的 Frequency、高/低電位水平，寫入之後即可開始輸入任意波形的每個節點電壓值，完成之後寫入並輸出。
2. 請自訂頻率、高/低電位水平，並在兩電位水平間自行決定要給多少點、每個點的電壓值，並用逗號隔開，完成之後請將應用程式 APP、產波器畫面、示波器畫面一同截取並寫至結報中。

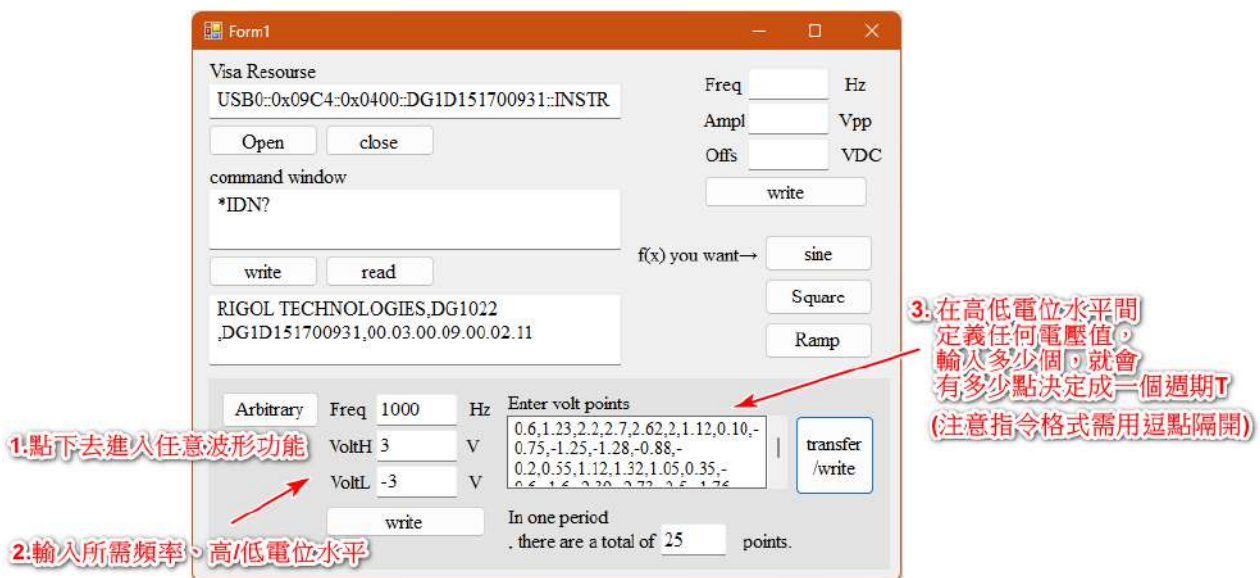


圖 29、實驗內容三目標

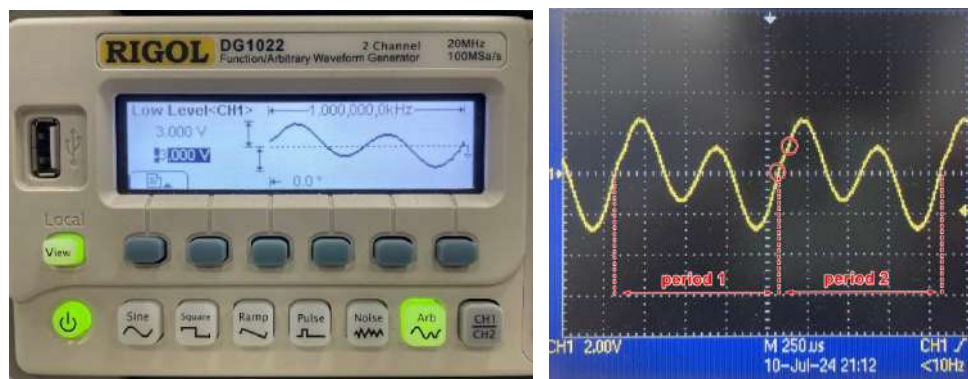


圖 30、設定完成後產波器會自動設定成自定義波形，右圖為示波器畫面

(hint1：在 Form1 將範圍拉大足夠畫面，拖曳所需 label、textbox、button 數量並自行排版，write 鈕必須能集合頻率、高/低電位水平的文字盒中的數值並進行寫入，指令格式請參考編程手冊，需注意兩條寫入指令之間要有時間間隔。

hint2：在函數波形產生器內的垂直分辨率為 14 bits，因此 0~16383 兩端分別對應低電位水平與高電位水平，這也就是為什麼你之後在決定每個節點的電壓值時不能設超過兩者的範圍，否則產波器會識別不出來。

注意

寫入之後可看到自定義之任意波形已寫入至函數波形產生器，在函數波形產生器之 View 上切換成圖形顯示模式來檢視，可發現函數波形產生器會自動切入 Arb，並顯示成使用者自訂參數之波形，若身旁有示波器，可用雙頭 BNC 纜線連接函數波形產生器與示波器，點開 Signal Output，即可將自定義之波形輸出至示波器檢視。

（注意給予數據點數至訊號波形產生器後，其會自動將此波形依給定頻率視為一個周期，因此會在示波器屏幕上，數據之最後一個點會與下一周期之第一數據點做線性連接。））

<code hint1>

```
string strWrite2_1 = textBox7.Text; //Get command from text box
string strWrite2_2 = textBox8.Text;
string strWrite2_3 = textBox9.Text;

try
{
    mbSession.Write("freq " + strWrite2_1); //Send Write command to VISA resource
    Thread.Sleep(200); //wait 0.2 s
    mbSession.Write("volt:high " + strWrite2_2);
    Thread.Sleep(200); //wait 0.2 s
    mbSession.Write("volt:low " + strWrite2_3);
}
catch (Exception exp)
{
    MessageBox.Show(exp.Message);
}
```

<code hint2>

```
// 假設有一個包含電壓值的字串，每個值之間使用逗號分隔
string voltageInput = textBox10.Text;

// 將字串分割為數字陣列
double[] voltages = voltageInput.Split(',').Select(double.Parse).ToArray();

// 設定最小和最大電壓值
double minVoltage = Convert.ToDouble(textBox9.Text);
double maxVoltage = Convert.ToDouble(textBox8.Text);

// 計算振幅值的範圍
int amplitudeRange = 16383;

// 創建振幅值陣列
int[] amplitudes = new int[voltages.Length];
```

```

// 計算並填充振幅值陣列
for (int i = 0; i < voltages.Length; i++)
{
    double voltage = voltages[i];

    // 反向映射
    int amplitude = (int)Math.Round((voltage - minVoltage) / (maxVoltage - minVoltage) *
amplitudeRange);

    // 將振幅值存儲在陣列中
    amplitudes[i] = amplitude;
}

// 將振幅值陣列輸出至 TextBox
string strWrite3_1 = string.Join(",", amplitudes);
string strWrite3_2 = "func:user volatile";
int aa = voltages.Length;
textBox11.Text = aa.ToString();

try
{
    mbSession.Write("data:dac volatile," + strWrite3_1); //Send Write command to VISA resource
    Thread.Sleep(200); //wait 0.2 s
    mbSession.Write(strWrite3_2);
}
catch (Exception exp)
{
    MessageBox.Show(exp.Message);
}

```

(在 PC 環境下，此人機介面需要使用 Win 10 或更新的作業系統，並需要安裝 NI Measurement Studio 來建立適合的資料環境，以便控制 Rigol 的函數波形產生器。)

《附錄一》

《函數波形產生器(Function generator)之任意波形編輯/創建》

在函數波形產生器上，選擇 **Arb** 鍵，常規顯示模式下，在螢幕下方會顯示任意波的操作選單，通過使用任意波形的選單，我們可以對任意波的”波形”進行參數上的設置。

任意波包括系統內建可選擇的波形和使用者自定義波形兩種類型的任意波形。設置任意波的參數主要包括：頻率/周期，振幅/高電壓水平，偏移/低電壓水平，相位。通過改變這些參數，可得到不同的任意波，如下圖 31 所示。

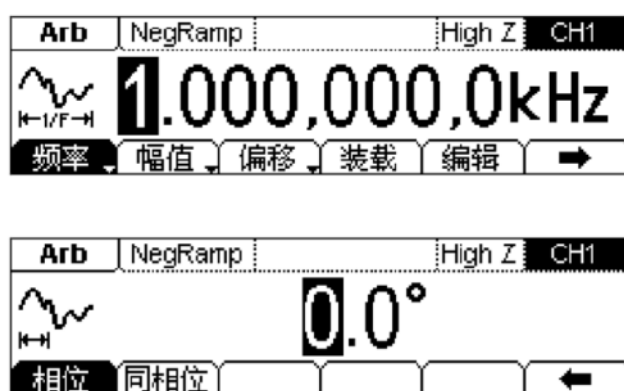


圖 31、任意波形參數設置顯示介面

表 1、Arb 波形的主選單說明

功能選單	說明
頻率/周期	設置波形頻率/周期
振幅/高電壓水平	設置波形振幅/高電壓水平
偏移/低電壓水平	設置波形偏移量/低電壓水平
裝載	選擇內建任意波形作為輸出
編輯	創建和編輯任意波形
相位	設置任意波形初始相位

選擇內建波形

訊號產生器內部存有 48 個內建任意波形，並提供 10 個非易失性存儲位置以存儲使用者自定義的任意波形。若欲選擇其中的內建波形，按 **Arb** → **裝載** → **內建**，進入下圖 32 所示之介面。



圖 32、內建波形顯示介面

表 2、Arb 波形的主選單說明

功能選單	設定	說明
常用	NegRamp/AttALT/AmpALT/StairDown/StairUp/StairUD/Cpulse/PPulse/NPulse/Trapezia/RoundHalf/AbsSine/AbsSineHalf/SineTra/SineVer	選擇常用波形
數學	ExpRise/ExpFall/Tan/Cot/Sqrt/X^2/Sinc/Gauss/HaverSine/Lorentz/Dirichlet/GaussOulse/Airy	選擇常用數學函數
工程	Cardic/Quake/Gamma/Voice/TV/Combin/BandLimited/StepResp/Butterworh/Chebyshev1/ Chebyshev2	選擇常用工程波形
窗型函數	Boxcar/Barlett/Triang/Blackman/Hamming/Hanning/Kaiser	選擇常用窗型函數波形
其他	RounsPM/DC	選擇其他波形
選擇		選擇已選中的波形

如下圖 33 所示，按數學，使用旋鈕選中”ExpRise”函數，然後按選擇，訊號產生器將輸出所選波形，此時按 View 鍵切換為圖形顯示模式即可查看方才所選之波形。

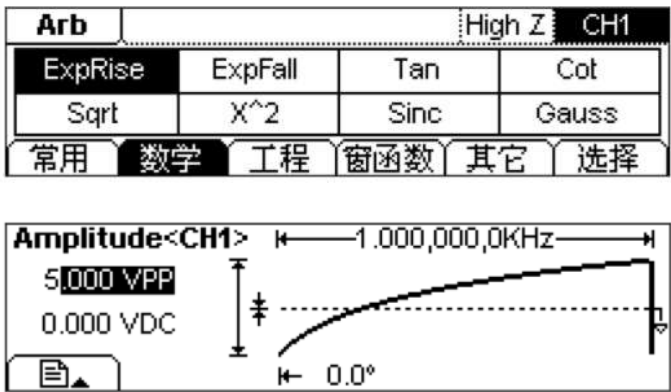


圖 33、圖形顯示模式下的波形演示

使用者自定義任意波形

訊號產生器具有編輯任意波形的功能，使用者可以通過初始化點的操作來創建任意的新波形，具體操作如下。

按 Arb → 編輯，進入下圖 34 所示之界面。

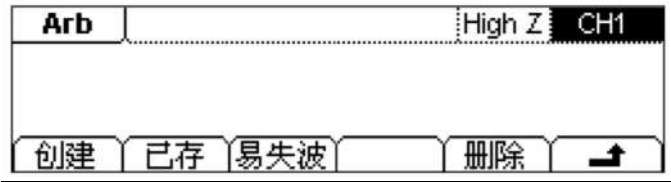


圖 34、操作功能選單

表 3、編輯波形的選單說明

功能選單	說明
創建	創建新的任意波形，並覆蓋易失性存儲器中的波形。
已存	編輯存儲在非易失性存儲器中的任意波形
易失波	編輯存儲在易失性存儲器中的任意波形
刪除	刪除存儲在 10 個非易失性存儲器中的一個任意波形

創建新波形，按 Arb → 編輯 → 創建進入下圖所示界面，對你要建立的波形進行前置設置。



圖 35、創建新波形參數值設置顯示界面

表 4、設置新創鍵波形參數

功能選單	說明
周期	設置任意波形的周期
高電壓水平	設置任意波形的最高電壓水平
低電壓水平	設置任意波形的最低電壓水平
插值開/插值關	啟用在波形的定義點之間的線性內插 禁用在波形的定義點之間的線性內插
點數	設置任意波形初始化點數
編輯點	啟動波形編輯器

1. 設置點數

選擇**點數**，設置任意波形的初始化點數，如下圖所示：

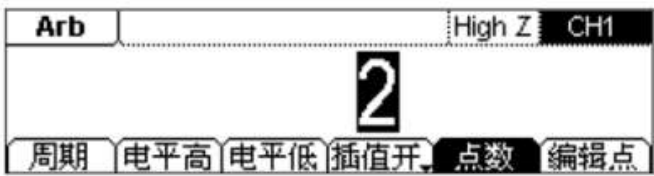


圖 36、設置此波形點的總數

當創建新波形時，波形編輯最初建立一個具有兩個點的波形，波形編輯器自動地將波形的最後一個點連接到點 1 的電壓水平，以創建一個連續波形。可創建最多 4096 的點的任意波形。

在默認情況下，點 1 設置為高電壓水平，固定在 0 秒，點 2 設置為低電壓水平，設置為指定循環周期的一半。

2. 設置插值

選擇**插值開**，啟用在波形點之間進行線性內插。

選擇**插值關**，在波形點之間維持不變的電壓水平，並創建一個類似的數字波形。

3. 編輯波形點

通過為每個波形點指定”時間”和”電壓值”來定義波形，選擇**編輯點**，進入如下圖 37 所示之界面，首先編輯第一個點，按**電壓**，進入圖 38 所示指定第 1 個點的電壓值。



圖 37、進入編輯點界面



圖 38、編輯第 1 個點的電壓值

編輯完第一個點的電壓再按**點**回界面後，將旋鈕往右，進入圖 39 所示，編輯第 2 個點，從第 2 個開始需要指定時間。



圖 39、編輯第 2 個點

表 5、編輯波形點參數選單說明

功能選單	說明
點	選擇不同的波形點，對各點的時間和電壓參數進行設置。
時間	設置當前點的時間值
電壓	設置當前點的電壓水平
插入	在當前點和下一個定義點的中間插入一個新的波形點，使用時間和電壓定義新點。
刪除	刪除當前的波形點
保存	將已創建的波形存至非易失性存儲

4. 存儲波形至非易失存儲器

波形創建完畢後，按保存進入 Store/Recall 的功能界面，如下圖所示，將波形保存到非易失性存儲器或外部存儲器中（注意在非易失性存儲器中，每個波形存儲的位置，只能存儲一個波形，如果有新波形存儲進來，舊的波形將被覆蓋）。



圖 40、保存使用者自定義任意波

存儲完畢後，按 View 鍵切換為圖形顯示模式，查看所創建的波形，如下圖 41 所示。

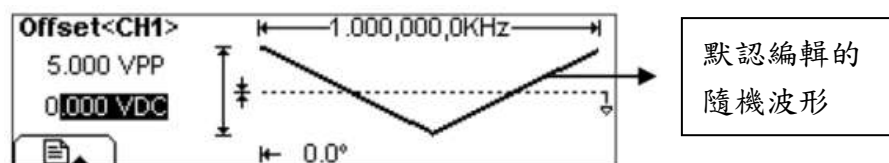


圖 41、圖形顯示模式下的波形