

雷射掃瞄儀

一、 實驗題目舉例：

1. 撰寫一程式能連續讀取資料
2. 撰寫一程式能將資料顯示在按鈕(Button)上
3. 撰寫一程式能顯示英制與公制的資料(請參考儀器面板上的設定)
4. 撰寫一程式能連續讀取 10 次的值
5. 撰寫一程式能只顯示小數部分
6. 撰寫一程式能只顯示前 5 位的部分
7. 撰寫一程式能讀取 10 次的值並能算出其整數的最大值
8. 設定雷射掃描儀當整數部分數值大於設定值時就警告(使用 label 元件, 顯示警告文字)

進階實驗題目舉例

1. 測量半透明物體，如透鏡及玻璃棒，撰寫一程式能做校正，使量測值合理化。
2. 指紋掃瞄門禁管理。

二、 實驗目的：

本實驗介紹雷射掃瞄儀之光學架構與量測原理，以及真圓度的定義與量測方式。根據真圓度的定義，我們在個人電腦上撰寫量測真圓度之程式，利用雷射掃瞄儀測量圓柱體工件的直徑值，再藉由 RS232 傳輸線的連接，將直徑值傳輸給電腦的程式做處理，我們即得此圓柱體之真圓度。

三、 實驗儀器：

雷射掃瞄儀

一台

RS232 傳輸線	一條
個人電腦 (含 Borland C++ Builder 軟體)	一台
圓柱體待測工件	一件

3、實驗原理：

3.1 雷射掃瞄儀的介紹

雷射掃瞄儀，是將雷射光束對被測物件做高速掃瞄，然後計算投影而得測量尺寸。因為是採用非接觸式的連續性測量方式，對於各種材質的工件均能做高精度的尺寸量測。雷射掃瞄儀除可對單一工件做量測外，亦可放在工作台上對生產線上之工件做連續性量測，在測量的功能上有標準值及上下限設定的裝置與不合格之警告信號，可以達到自動判別合格的功能。

雷射掃描器除了測量顯示外，也可以作統計運算，再測量完整批工件或在測量中，都可以按鍵檢查測量件數，最大、最小值、平均值、散佈、標準方差。此外，雷射掃瞄儀亦可外接電腦、印表機等，可以將測量所得的數據，記錄下來或作成檔案，以供分析或高階統計之用，而達成全自動測量及數據處理。圖1為雷射掃瞄儀之實體圖。



圖 1 雷射掃瞄儀之實體圖

雷射掃瞄儀內部有一個雷射紅光二極體，當雷射掃瞄儀啟動時，紅光雷射會入射馬達上轉動的反射面鏡，由反射面鏡反射出來的雷射光經過掃瞄透鏡後，會

成為一束掃瞄待測工件之平行光，如圖 2 所示，掃瞄光束若經過待測工件則被遮擋，無法到達接收透鏡，沒有經過待測工件的掃瞄光束到達接收透鏡後，會聚集在光檢測器上，我們根據光檢測器上工件的陰影來判斷工件的直徑大小。

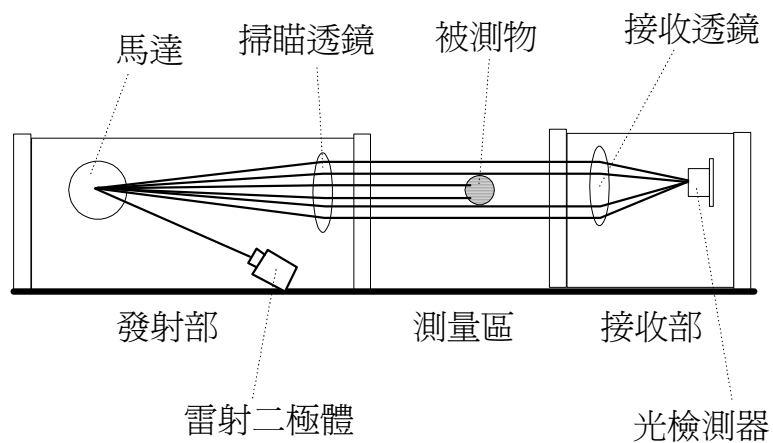


圖2 雷射掃瞄儀之測量原理

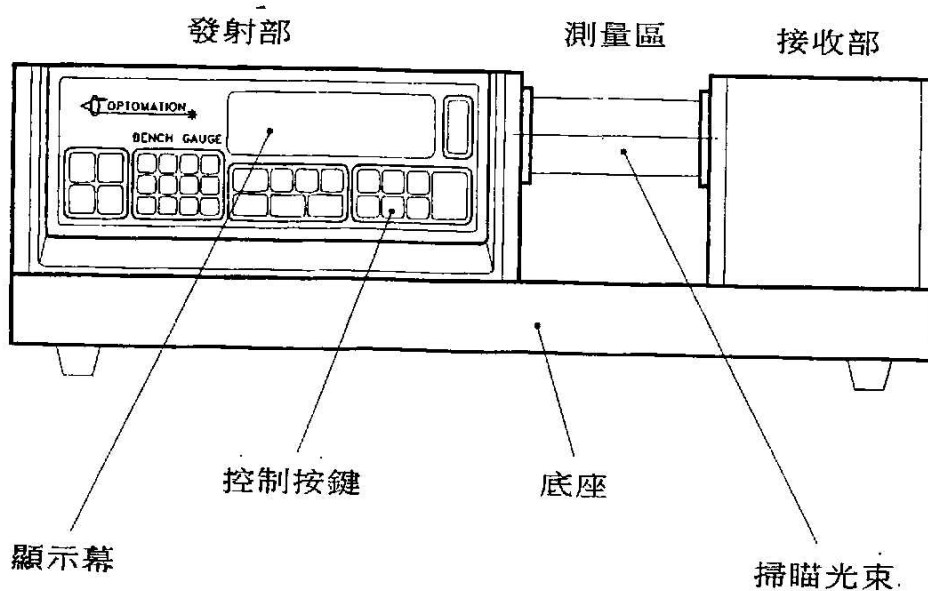


圖3 外觀說明圖

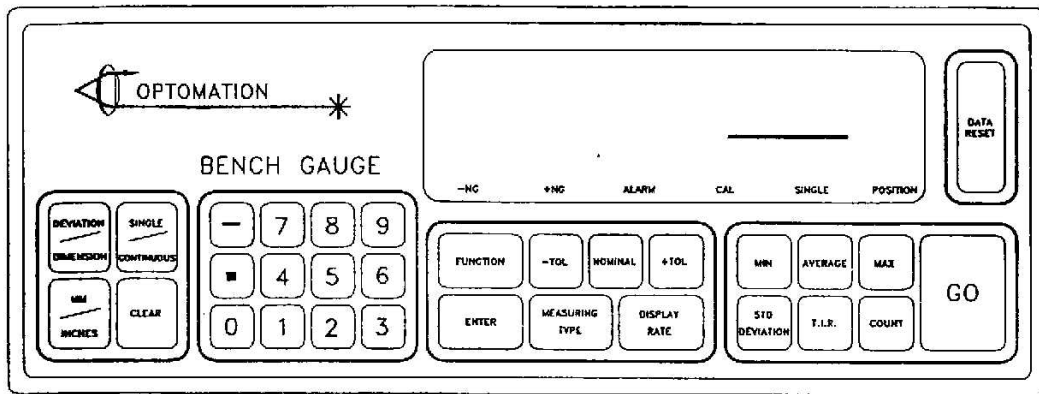


圖4 功能說明

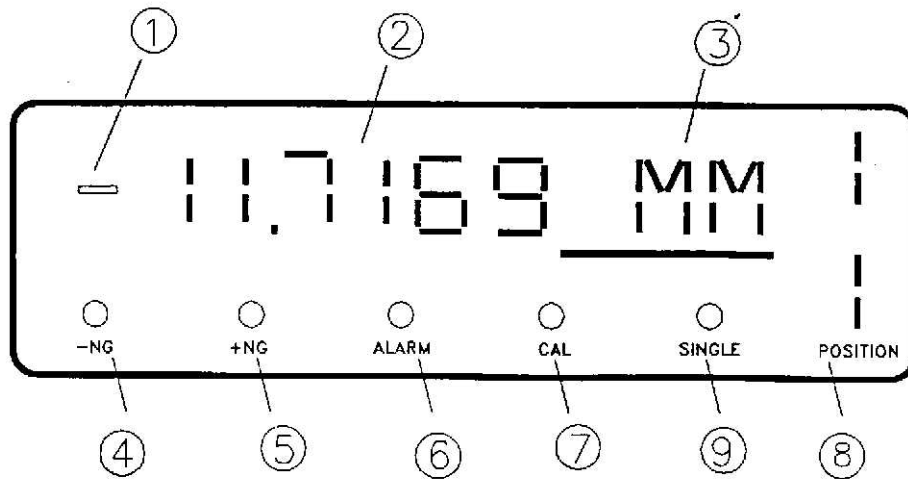













圖5 顯示幕說明








- 1 正負號顯示。
- 2 測量讀值顯示 (6位3LED)
- 3 文字訊號顯示 (3位4LED)
- 4 下限不合格警示燈。
- 5 上限不合格警示燈。
- 6 蜂鳴器警告開關指示燈。
- 7 校準狀態指示燈。
- 8 測量物高低位置指示燈。
- 9 測量狀態指示燈。

狀態鍵

- | | | |
|-------|---|--------------|
| (1) |  | 差值/絕對值顯示切換鍵 |
| (2) |  | 單點測量/連續測量切換鍵 |
| (3) |  | 公制/英制顯示切換鍵 |
| (4) |  | 清除誤置工件測量值的按鍵 |

設定鍵

- | | | |
|-------|---|-----------|
| (1) |  | 標準值設定鍵 |
| (2) |  | 負向容許值設定鍵 |
| (3) |  | 正向容許值設定鍵 |
| (4) |  | 測量功能設定鍵 |
| (5) |  | 功能設定鍵 |
| (6) |  | 讀值顯示頻率設定鍵 |
| (7) |  | 顯示幕清除鍵 |

- | | | |
|-------|---|---|
| (1) |  | 測量值中的最小值 |
| (2) |  | 測量值中的平均值 |
| (3) |  | 測量值中的最大值 |
| (4) |  | 測量值的差值，就是最大值與最小值的差。 |
| (5) |  | 標準方差。 |
| (6) |  | 清除測量記憶·讀值及統計結果 |
| (7) |  | 當選擇工件測壘(SINGLE)時，每放入一個工件進測量區後·需押 GO 一次·雷射掃瞄儀才對工件作一次測量·並保持住讀值·待下一次押 GO 才變更讀值 |

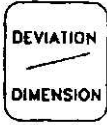
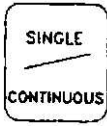
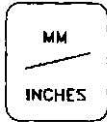
開機設定

本掃瞄儀以測量精確、功能完備、使用簡單為設計前提，所以針對大部份的使用場合，一開機後，不須設定·已俱有預先設定的功能如下：

- | | |
|-------------------|--------|
| (1)DIMENSION | 絕對值顯示 |
| (2)MM | 公制單位 |
| (3)MEASURING TYPE | 測量外徑功能 |
| (4)-TOL | 99,000 |
| (5)+TOL | 99.000 |
| (6)DISPLAY RATE | 每秒顯示一次 |

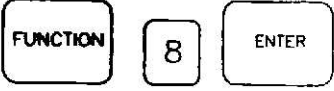

3.2 操作說明

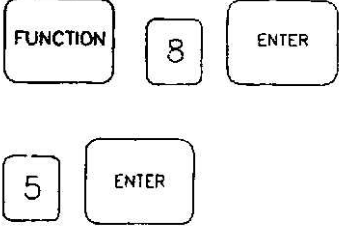
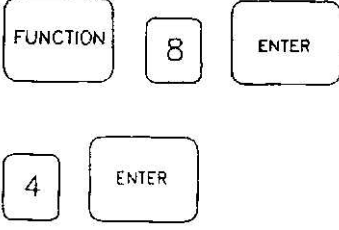
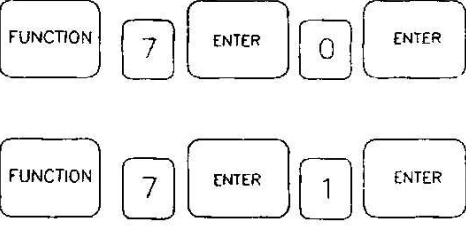
顯示幕顯示

測量模式切換	按鍵步驟
<p>1-1 差值/絕對值顯示切換</p> <p>1 · 2357MM</p> <p>1 · 2357DEV</p> <p>I-2357MM</p>	
<p>1-2 零件測量/連續測量切換</p> <p>OSINGLE燈亮起。讀值鎖定</p> <p>SINOLB燈熄滅。讀值隨測量物變化</p>	
<p>1-3 公制/英制互換</p> <p>1 · 00000IN</p> <p>25.4000MM</p> <p>1.00000IN</p>	

顯示位數設定

顯示幕顯示

	按鍵步驟
<p>(1) 顯示六位數(0.1um)</p> <p>12.315MM</p> <p>12.3154MM</p>	 

<p>(2)顯示五位數(1um)</p> <p>12.3154MM</p> <p>12-315MM</p>	
<p>(3)顯示四位數(10um)</p> <p>12.315MM</p> <p>12.31MM</p>	
<p>2-5蜂鳴器開關設定</p> <p>0ALARM燈熄滅·測量值不合格時無警告聲</p> <p>0ALARM燈亮起，測量值不合格時，發出警告聲</p>	

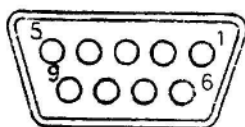
3.3 信號連接:

說明

雷射掃瞄儀的測量讀值，可經由RS-232C的連出，傳送至至電腦，以供作進一步的分析與記錄。當雷射掃瞄儀設定在零件測量狀態時，每按一次GO鍵，則顯示一組讀值，同時經由RS-232C傳送一組信號出來，等到下一次按GO鍵才會再傳送信號。如果設定在連續測量狀態，則依顯示頻率，連續的經由信號接頭傳送信號，例如設定每秒顯示一次，則每秒傳送一組讀值的信號出來。

接點位置信號圖

RS-232C信號接頭在雷射掃瞄儀的左側板上



接腳號碼	信號名	說明
1	CO	資料載波檢測出
2	TXD	發送資料
3	RXD	接收資料
4	DSR	資料組備妥
5	GND	信號用接地
6	DTR	資料終端備妥
7	CTS	發送許可
8	RTS	發送要求

表 1

RS-232C資料傳送形式RS-232C的信號無論為絕對值或是偏差值輸出，皆是以ASCII, Code傳送資料形式同面板輸出。

校準

雷射掃瞄儀在出廠時已作出廠校準，均合於規格內的精度（標準溫度為20C）。但是，使用者所處的環境溫度及使用狀況可能與儀器出廠標準不同，如果要求較佳的精度，就必須要依下述步驟校準。

將標準工件（例中為12.000mm直徑的圓棒規）置於測量區中

顯示幕顯示

輸入鍵位

12.0735	MM	
	FUN	FUNCTION
1	FUN	1 ENTER
12.0000	FUN	1 2 . 0 0 0 0
12.0000	MM	ENTER

五、實驗步驟

雷射掃描儀的作動與操作是需 RS-232C 通訊的族群中，最簡單的一個實驗，其啟動步驟如下

1. 連接 RS-232C 至電腦
2. 設定顯示單位（公制/英制）
3. 設定標準值
4. 設定上下限
5. 啟動程式

範例程式

程式中的示範如何驅動 RS-232C 和抓取資料

```
//-----
#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
```

```

#include "Math.h"
//-----
#pragma package(smart_init)
#pragma link "SPComm"
#pragma resource "*.dfm"
TForm1 *Form1;
//-----
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
    LastSize = 0;
}
//-----
void __fastcall TForm1::GoClick(TObject *Sender)
{
    Comm1->StartComm();      //啓動 RS-232C
    Mem1->Lines->Add(" Open Success ");
}
//-----

void __fastcall TForm1::StopClick(TObject *Sender)
{
    Comm1->StopComm();      //停止 RS-232C
    Mem1->Lines->Add(" Port Is Closed ");
}
//-----

void __fastcall TForm1::Comm1ReceiveData(TObject *Sender, Pointer Buffer,
    WORD BufferLength)
{
    //這是範例程式最重要的地方，示範了如何抓取資料，並轉為數字。這裡很重要的一點，我們
    //抓到的資料是文字並不是數字。
    AnsiString InMessage = (char *) Buffer;
    int iDot = InMessage.LastDelimiter(".");    //找尋逗點在文字中的位子
    if( iDot > 1)
    {
        AnsiString asNumber = InMessage.SubString( 0, iDot-1 );    //抓出整數的部分
        int Size = StrToInt(asNumber);    //轉為數字
    }
}

```

```

        asNumber = InMessage.SubString( iDot+1,4 );      //抓出小數的部分
        Size = Size*10000 + StrToInt( asNumber );      //轉為數字
//將資料加至顯示區
        if( abs(LastSize - Size) >= (10000/pow(10,iAccurate)))
            Memol->Lines->Add( InMessage );
        LastSize = Size;
    }
    else Memol->Lines->Add( InMessage );
}
//-----
void __fastcall TForm1::editClick(TObject *Sender)
{
    StopClick(this);
    Form2->ShowModal();
}
//-----
//以下是 from2 的部分，都只是一些細瑣的的參數設定的問題
//應該都能輕易地看懂
//-----
#include <vcl.h>
#pragma hdrstop

#include "Unit2.h"
#include "Unit1.h"
//-----
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//-----
__fastcall TForm2::TForm2(TComponent* Owner)
    : TForm(Owner)
{
}
//-----

void __fastcall TForm2::ParityCheckClick(TObject *Sender)
{

```

```

    Parity->Enabled = true ;
    Form1->Comm1->ParityCheck = true ;
}
//-----

void __fastcall TForm2::ParityChange(TObject *Sender)
{
    Form1->Comm1->Parity = Parity->ItemIndex;
}
//-----

void __fastcall TForm2::ParityCheckDb1Click(TObject *Sender)
{
    Parity->Enabled = false ;
    Form1->Comm1->ParityCheck = false ;
    ParityCheck->Checked = false;
}
//-----

void __fastcall TForm2::BytesChange(TObject *Sender)
{
    Form1->Comm1->ByteSize = Bytes->ItemIndex;
}
//-----

void __fastcall TForm2::PortsChange(TObject *Sender)
{
    Form1->Comm1->CommName = Ports->Items->Strings[Ports->ItemIndex];
}
//-----

void __fastcall TForm2::BaudRateChange(TObject *Sender)
{
    Form1->Comm1->BaudRate = StrToInt(BaudRate->Items->Strings[BaudRate->ItemIndex]);
}
//-----

void __fastcall TForm2::StopBitsChange(TObject *Sender)
{
    Form1->Comm1->StopBits = StopBits->ItemIndex;
}
//-----

void __fastcall TForm2::CloseForm2Click(TObject *Sender)

```

```
{
    CloseForm2->Checked = false;
    Form2->Close();
}
//-----
void __fastcall TForm2::accChange(TObject *Sender)
{
    Form1->iAccurate = acc->ItemIndex;
}
//-----
```