==========
使用者編程功能
==========
功能: 使用者 可以自行加入 C 的程式碼, 當成模組使用.

==========
使用方法
==========
程式模組:
在 "元件模組" 的資料夾下 點選 "編程", 輸入名稱 (識別用)
點擊 "編程" 按鈕, 則可以開始輸入程式碼.
注意: 並非所有 Arduino 的 函式 都可以使用, 請參閱 支援函式說明

自定義程式全域變數:
在 "變數設定" 的資料夾下 點選 "編程變數",
可以開始寫入 變數的宣告 以及 setup() 是程式啟動時的初始化的程式碼
若有 "自訂的函式", 也可以寫在這邊.

==========
Console Input 使用說明
==========
1. 使用系統變數讀取 Console Input 輸入的資料
系統變數名稱為 S_CONSOLE_INPUT 類型為 字串, 在程式方塊中可以比對此變數內容.
注意: 輸入的字串會儲存在 S_CONSOLE_INPUT 變數, 但程式方塊無法修改 S_CONSOLE_INPUT 的內容,
因此若只單純的比對字串 則會發生每次都成立, 除非輸入新的字串.
所以 必須建立另一個字串變數 MyInputString, 用下面的邏輯來避開此問題

```
if (MyInputString != S_CONSOLE_INPUT)
{
  MyInputString = S_CONSOLE_INPUT;
  //Do something to check S_CONSOLE_INPUT ...
}
```

請參閱 ConsoleInput.fco

2. 使用 使用者編程讀取 Console Input 輸入的資料

相關函式說明: 請參考 "支援函式說明"

範例:

```
if ( IsGotConsoleInput() == 1 )
{
  if ( strcmp(gConsoleInputData, "f1") == 0 ) {
    ConsoleWriteString("1: Hello\n");
  } else if (strcmp(gConsoleInputData, "f2") == 0 ) {
    ConsoleWriteString("2: World\n");
```

```
  }
  ResetConsoleInput();
}
```

```
============================
支援函式說明
============================
******************
* Digital I/O
******************
* pinMode(pin, mode)
  指定 pin 是輸入 或是 輸出.

* digitalWrite(pin, value)
  GPIO 控制 OUTPUT pin 設為 High or Low

* digitalRead(pin)
  讀取 GPIO INPUT pin 的狀態

******************
* Analog I/O
******************
* analogReference(type)
  設定輸入的參考電壓, type: DEFAULT, INTERNAL1V1, INTERNAL2V56, EXTERNAL

* analogRead(pin)
  從指定的 pin 腳讀取類比電壓訊號,

* analogWrite(pin, value)
  對 pin 腳輸出 PWM 波型

******************
* Time
******************
* delay(ms)
  讓程式延遲 millisecond

* delayMicroseconds(us)
  讓程式延遲 microsecond

******************
* Advanced I/O
******************
```

* tone(pin, frequency, duration)
  在指定的 pin 腳上產生一個指定頻率的方波，可以指定持續時間或者持續直到呼叫 noTone()，在這個指定的 pin 腳上可以接蜂鳴器或者喇叭來發出一段音頻

* noTone(pin)
  停止由 tone() 產生的方波，在沒有任何音頻產生的時候呼叫此函式是沒有作用的

****
* Math
****
* min(x, y)
  比較出兩數中最小者

* max(x, y)
  比較出兩數中最大者

* abs(x)
  計算出一個數值的絕對值

* constrain(x, a, b)
  限制一數值於某範圍內

* map(value, fromLow, fromHigh, toLow, toHigh)
  將某個範圍內的一個數值重新映射到另一個範圍，亦即數值 fromLow 被映射到數值 toLow 去，
  而數值 fromHigh 則被映射到數值 toHigh， value 則維持與 fromLow、fromHigh 的比例關係映射至新的範圍

* pow(base, exponent)
  計算一個數字的乘方結果

* sqrt(x)
  計算一個數字的平方根

****
* Trigonometry
****
* sin(rad)
  計算角度（弧度）的正弦值。結果將會在 -1.0 到 1.0 之間

* cos(rad)
  計算角度（弧度）的餘弦。結果將會在 -1.0 到 1.0 之間

* tan(rad)
  計算角度（弧度）的正切。結果將會在負無窮大到無窮大之間

****
* Characters
****
* isAlphaNumeric(c)
  分析一個字元是否為英文字母或數字（即 a ~ z 或 A ~ Z 或 0 ~ 9 其中之一）。

* isAlpha(c)
  分析一個字元是否為英文字母（即 a ~ z 或 A ~ Z 其中之一）

* isAscii(c)
  分析一個字元是否為 ASCII 字符

* isWhitespace(c)
  分析一個字元是否為空格（' '）或 Tab（'\t'）字符

* isControl(c)
  分析一個字元是否為控制字符

* isDigit(c)
  分析一個字元是否為數字 0 ~ 9 其中之一

* isGraph(c)
  分析一個字元是否為空格以外的可列印字符

* isLowerCase(c)
  分析一個字元是否為小寫的英文字母

* isPrintable(c)
  分析一個字元是否為可列印的字符（包含空格字符）

* isPunct(c)
  分析一個字元是否為空格、數字、英文字母以外的可列印字符

* isSpace(c)
  分析一個字元是否為空格符號（' '）、換頁符號（'\f'）、新列符號（'\n'）、回捲符號（'\r'）、水平 Tab（'\t'）、垂直 Tab（'\v'）

* isUpperCase(c)
  分析一個字元是否為大寫的英文字母

* isHexadecimalDigit(c)
  分析一個字元是否為十六進制的數字（即 0 ~ 9、A ~ F 其中之一）

****
* Random Numbers

****
* long random_max(long max)
  取得亂數 回傳值 小於 max

* long random_min_max(long min, long max)
  取得亂數 回傳值 大於等於 min, 且小於 max

* void randomSeed(unsigned int seed)
  設定 亂數種子

****
* Bits and Bytes
****
* lowByte(x)
  擷取一個變數中的低位元組 (例如一個 word 中最右邊的位元組)。

* highByte(x)
  擷取一個 word 中的高位元組 (最左邊的位元組，或者一個更大的資料型別中的第二低的位元組)。

* bitRead(x, n)
  讀取一個數字中的某個位元。

* bitWrite(x, n, b)
  對一個數字中的某個位元寫值。

* bitSet(x, n)
  將變數數值的某一個 bit 設定成 1。

* bitClear(x, n)
  將變數數值的某一個 bit 設定成 0。

* bit(n)
  計算指定位元為 1 時所代表的數值 (例如：位元 0 的值是 1，位元 1 的值是 2，位元 2 的值是 4 …等)

****
* Console
****
* uint8_t IsGotConsoleInput()
  判斷是否有收到 Console Input 輸入的資料, 傳回 1 表示有, 其它值 表示無

* void ResetConsoleInput()
  重置 收到的資料與判斷旗標, 若 IsGotConsoleInput() 為 1, 執行此函式之後,
  IsGotConsoleInput() 旗標會被設為 0 直到下次再收到 Console Input 才會傳回 1

* char *gConsoleInputData
  這是字串變數會記錄 收到的 Console Input 字串.

* ConsoleWriteString(char *pStr)
  輸出 字串 到 Console Output