# USER'S MANUAL

**Protech**

**API Package**

# 軟體操作手冊

版本: **M2** 日期: **2012/04/06**

# Protech API Package
# 軟體操作手冊

## 關於本手冊

本手冊將說明如何操作與設定 Protech API Package 軟體。本手冊及相關應用軟體之著作權爲南京資訊股份有限公司所有，受著作權法、國際著作權條約以及其他智慧財產之法律及條約之保護，未經本公司書面同意不得複製本軟體和隨附之使用手冊及其他各項資料之全部或一部，並不得以任何形式重製或使用，也不可未經製造商允許以電子或機械方式重製(影印、記錄、儲存)或使用，除原產品套件已具備之程式原始碼及專案特殊用途外，不得利用任何方法進行取得、使用本軟體受保護之程式原始碼、文字資料、圖形或音效等檔案。

在相關法律所允許之最大範圍內，南京資訊或其經銷商對於因使用或不能使用軟體產品而遭受之衍生性損害(包括但不限於營業利益之損失、營業中斷、營業資訊之損失或其他金錢損失)不負任何損害賠償責任，並不因事先告知南京資訊或其經銷商，該損害發生之可能性而有所不同。

南京資訊股份有限公司保留對本手冊所提供之產品規格以及描述進行變更或修改之權利，所陳述之資訊係僅供參考，恕不另行通知。本手冊之所有部份，已於撰寫中善盡注意其說明正確性之職責，惟本公司並不保證毫無訛誤，特此聲明。在任何情況下，對資料遺失、收益損失或因此所造成任何特別、意外、重要、直接或非直接的損害，恕不負責。

閱讀完本手冊之後若仍無法解決您的問題，我們提供了有效的諮詢管道。針對所有一般性的技術支援、系統硬體規格建議、系統除錯報告、版本更新、資料庫與系統修改的選項等，若您需要更多產品資訊及支援，請與我們的經銷商聯繫。



南京資訊股份有限公司
114 台北市內湖區陽光街365巷24號
電話：(02) 8751-1111
傳真：(02) 8751-1199

http://www.protech.com.tw

# 軟體簡介

感謝您使用Protech API Package軟體。

在軟體開發的過程中常會遇到許多平台與硬體上的差異問題,在實際運用上除了需要了解硬體特性、控制方法、通訊協定等，處理的過程中往往耗費許多時間。南京資訊提供使用者一套API 解決方案並讓使用者不需了解硬體特性便能輕易控制硬體的方法，即Protech API Package軟體。

## 軟體特色

南京資訊API 解決方案，將讓我們的客戶獲得如下的收益：

- 加快產品上市時間：API 軟體幫助開發者在不了解晶片組規格及驅動架構的情況下編寫程序。
- 減少項目開發工作量：客戶不需要從頭開始編寫硬體的驅動程序，而是直接調用API軟體，便可通過底層直接控制硬體。

## 作業環境

- Windows 32 bit OS + .NET Framework 2.0 或以上版本

## 適用產品

- Industrial CPU Board
- Applied Computer
- Panel PC
- POS PC

## 支援功能

- Programmable GPIO
- Digital IO
- Watch Dog
- Cash Drawer
- Hardware Monitor
- i-Button function
- UPS function

# 目錄

# 第1章 開始使用

1

第1章將概略說明API Package的功能、檔案內容以及如何執行。

本章節次內容如下：

# 第1節 API Package內容

您可以在南京資訊產品隨附的光碟中找到API軟體相關檔案，內容說明如下。

| **Operation System** | Windows 32 bit + .NET Framework 2.0 或以上版本 | | |
|---|---|---|---|
| **Directory** | **Contents / File Name** | | **Description** |
| **Document** | Protech API Package User Guide A01-0000-000-02-xxxxxx_en.pdf | | 英文版操作手冊 |
| | Protech API Package User Guide A01-0000-000-02-xxxxxx_ch.pdf | | 中文版操作手冊 |
| | IO Description.pdf | | --- |
| | UPS Standard SBS Commands.pdf | | --- |
| **Function DLL** | | | |
| **Directory** | **Function** | **File Name** | **Description** |
| **ProxAPI standard** | *Cash Drawer* | Cash Drawer.dll | 控制Cash Drawer所需Dll |
| | *Digital* | Digital.dll | 控制Digital所需Dll |
| | *GPIO* | GPIO.dll<br>WinIo.dll<br>WinIo.lib<br>WinIo.sys<br>WINIO.VXD | 控制GPIO所需Dll |
| | *SMBUS* | WinIo.dll<br>WinIo.lib<br>WinIo.sys<br>WINIO.VXD<br>SMBUS.dll | 使用SMBUS所需要的Dll |
| | *WDT* | Watchdog.dll | 控制WDT所需Dll |
| | *i-Button* | IButtonAPI.dll<br>IBFS32.dll | 讀取i-Button所需Dll |
| | *Hardware Monitor* | Hardware Monitor.dll | 讀取硬體所需Dll |
| | *Battery* | SBS_Battery.dll<br>phymem.sys<br>pmdll.dll | 讀取控制電池資訊所需Dll |
| | **multilangXML.dll** | | 讀取XML必要檔案 |
| | **Initial.xml** | | 存放目前機型檔案 |
| | **ProxAP.exe** | | 程式執行檔 |
| | XML Files\Model Name*\Initial.xml | | 對應各個不同機型所對應到的檔案 |
| | Version.ini | | 紀錄版本資訊檔案 |
| **Sample Program** | | | |
| **Directory** | **Contents / File Name** | | **Description** |
| **DEMO PROJECT** | DEMO PROJECT\GPIO Sample Code | | C# VB6 VB.net Source Code |
| | DEMO PROJECT\Digital Sample Code | | C# VB6 VB.net Source Code |
| | DEMO PROJECT\Watchdog Sample Code | | C# VB6 VB.net MFC Source Code |

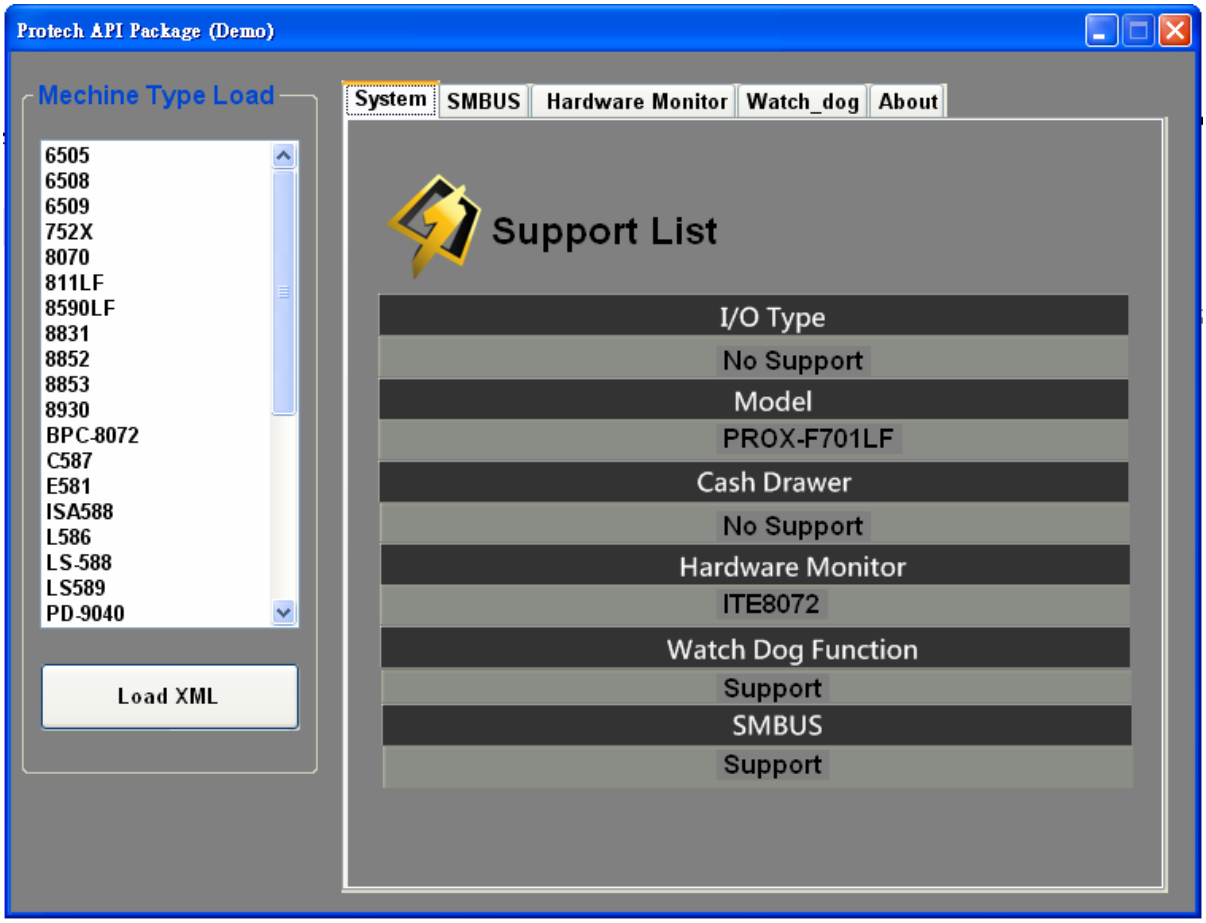
📖 Model Name is dependent on your machine type.

## 第2節 開啟Protech API Package

在API Package內，需要一個XML檔案，才能確定執行程式能夠被開啓使用。
以爲PS6509LF爲例, API Package所需檔案內容如下:
- ProxAPI standard\Cash Drawer.dll
- ProxAPI standard\multilangXML.dll
- ProxAPI standard\Watch dog.dll
- ProxAPI standard\Hardware Monitor.dll
- ProxAPI standard\XML Files\6509**Initial.xml**
- ProxAPI standard\ProxAP.exe

執行ProxAP程式(即ProxAP.exe檔案)，程式會依照您所點選的機型不同產生對應的功能頁籤。您可以在程式的System頁籤中左列Machine Type Load清單內, 點選產品的型號, 再點按**[Load XML]**即可取得此型號可支援的功能。

如下圖所示可知PROX-F701LF系列產品可支援的API功能有Hardware Monitor、Watch Dog、SMBUS等。



# 第2章 API使用範例

2

第2章將說明操作此API的程序，並列舉各類程式語言的sample code。

本章節次內容如下：

## 第1節 API操作程序

以**VB2005 .NET**爲例, 首先您必須宣告function, 您可以在您的專案中創造一個模組, 並填入下列function內容。

```
Declare Function GetCashDrawerStatus Lib CashDrawer.dll (ByVal num_drawer as short)
As Boolean

Declare Function CashDrawerOpen Lib CashDrawer.dll (ByVal num_drawer as short) As
Boolean
```

下一步, 創造按鈕來呼叫API Function (本例爲Cash drawer function)

1.呼叫Cash drawer open事件:

```
Private Sub cash_btn1_Click (ByVal Sender As System.Object, ByVal e As
System.EventArgs) Handles cash_btn1.Click
CashDrawerOpen(1), “1” specifies the cash drawer 1 port
CashDrawerOpen(2), “2” specifies the cash drawer 2 port
Timer1.start
```

2.偵測Cash drawer狀態

可創建timer將要做的動作寫到timer事件中

```
Private Sub Timer1_Tick (ByVal Sender As System.Object,ByVal e As System.EventArgs)
Handles Timer1.Tick
    Dim Receive_Status1 as Boolean
    Dim Receive_Status2 as Boolean
    Receive_Status1 = CashDrawerOpen(&H1)
    If Receive_Status1 = true then
       Text1.text = “cash drawer1 open”   ‘enter text into textbox.
Else
       Text1.text = “cash drawer1 close”    ‘enter text into textbox.
End if
‘=======================================
    Receive_Status2 = CashDrawerOpen(&H2)
    If Receive_Status2 = true then
       Text2.text = “cash drawer2 open”   ‘enter text into textbox.
Else
       Text2.text = “cash drawer2 close”    ‘enter text into textbox.
End if
‘=======================================
End sub
```

# 第2節 Sample Code

**(1) VB Declaration**

```
Declare Function GetCashDrawerStatus Lib CashDrawer.dll (ByVal num_drawer as short) As Boolean

Declare Function CashDrawerOpen Lib CashDrawer.dll (ByVal num_drawer as short) As Boolean
```

**(2) Call Function**

**Open cash drawer:**

```
CashDrawerOpen(1)
```

**Open cash drawer1**

```
CashDrawerOpen(2)
```

**Open cash drawer2**

**Check cash drawer status:**

```
Dim receive_status as Boolean
```

**Check cash drawer1 status**

```
Receive_Status = CashDrawerOpen(&H1)
```

**Check cash drawer2 status**

```
Receive_Status = CashDrawerOpen(&H2)
```

---

**(1) C# Declaration Method**

```
Public class PortAccess
{
[DllImport("CashDrawer.dll",EntryPoint = "Initial_CashDrawer")]
Public static extern void Initial_CashDrawer();
[DllImport("CashDrawer.dll",EntryPoint= "GetCashDrawerStatus")]
Public static extern bool GetCashDrawerStatus()
[DllImport("CashDrawer.dll",EntryPoint = "CashDrawerOpen")]
Public static extern bool CashDrawerOpen(short num_drawer);
}
```

**(2) Call Function**

**Open cash drawer1**

```
PortAccess. CashDrawerOpen(0x01);          //check cash drawer1 status
```

**Open cash drawer2**

```
PortAccess. CashDrawerOpen(0x02);          //check cash drawer2 status

Bool bstatus;
bstatus = PortAccess. GetCashDrawerStatus(0x01);
bstatus = PortAccess. GetCashDrawerStatus(0x02); //Before get cash drawer status, need to initial cash drawer first
```

**VB.NET extern function:**

```
Declare Function SetMinSec Lib “WatchDog.dll” (ByVal kind As Short,ByVal
delay_time As Short) As Boolean
Declare Function Stopwatchdog Lib “WatchDog.dll” ( ) As Short
Declare Function Setwatchdog Lib “WatchDog.dll” (ByVal value As Short) As Boolean
‘============================================================
Declare Function Digital_Initial Lib “Digital.dll” ( ) As Long
Declare Function Digtial_Set Lib “Digital.dll”(ByVal hex_value As Short) As Long
Declare Function Digtial_Get Lib ”Digital.dll” ( ) As Short
‘============================================================
Declare Function GPIO_Initial Lib “GPIO.dll” ( ) As Long
Declare Function GPIO_SetPort Lib “GPIO.dll”(ByVal direct As long)
Declare Function GPIO_Set Lib “GPIO.dll”(ByVal dout_value As long) As Boolean
Declare Function GPIO_Get Lib “GPIO.dll”( ) As Short
‘============================================================
Declare Function GetCashDrawerStatus Lib CashDrawer.dll (ByVal num_drawer as short)
As Boolean
Declare Function CashDrawerOpen Lib CashDrawer.dll (ByVal num_drawer as short) As
Boolean
```

---

**VB 6 extern function:**

```
Declare Function CashDrawerOpen Lib "CashDrawer.dll" (ByVal num_drawer As
Integer) As Boolean
Declare Function GetCashDrawerStatus Lib "CashDrawer.dll" (ByVal num_drawer As
Integer) As Boolean
```

🕮 VB.net short = integer VB6

# 第3章 API Package功能

3

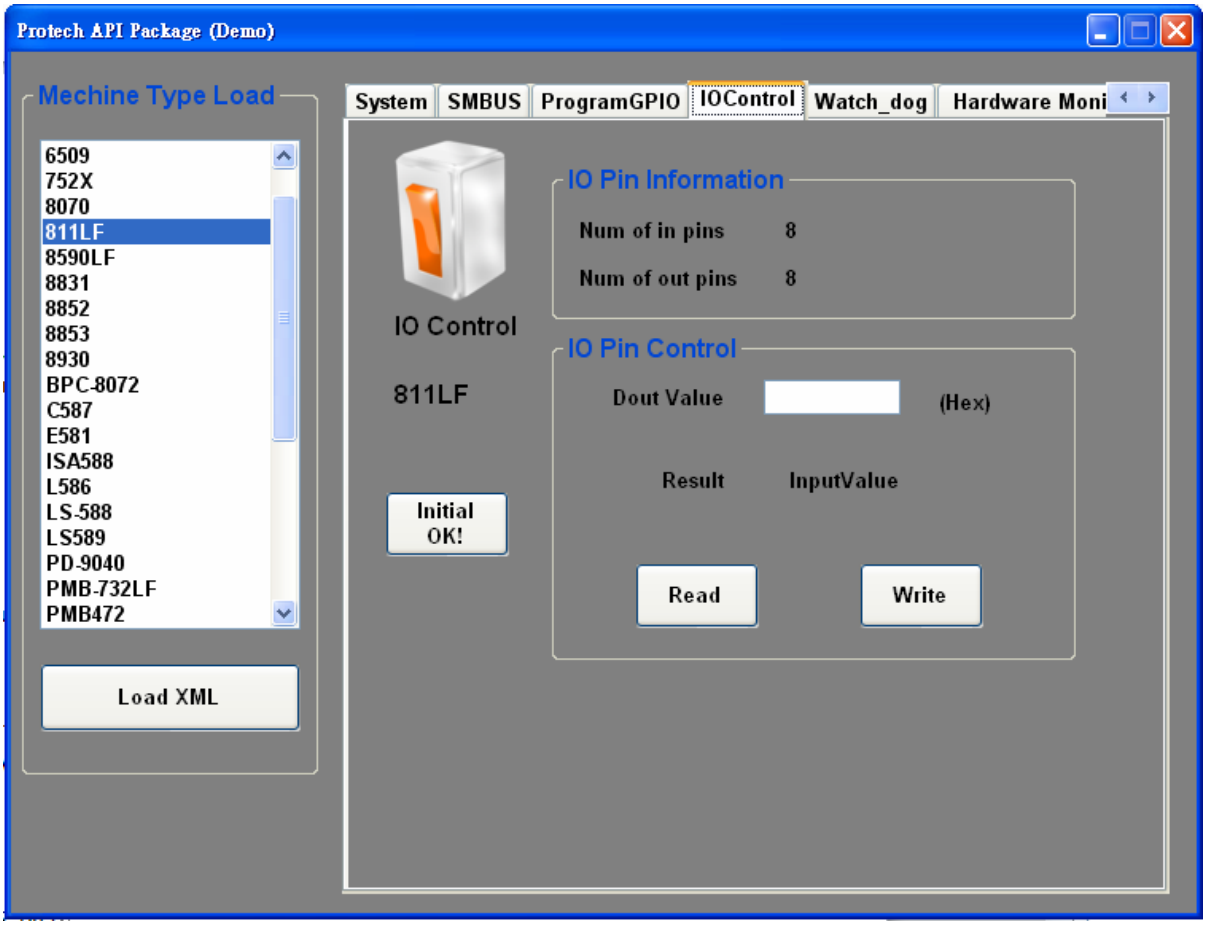
第3章將說明API Package各頁籤的功能屬性。

本章節次內容如下：

# 第1節 IO Control

ProxAPI程式示範了如何將ProxAPI Library包入使用者的應用程式

🕮 本程式是由VB.NET開發完成，在使用前須先注意是否安裝了.NET Framework2.0或以上版本。




**[Initial]**

初始化IO Function，初始化OK圖示如右:


🕮 若無顯示此圖示之後的操作，都可能無效。

**IO Pin Information**

顯示此台機型的輸入/輸出Pin數量

## IO Pin Control

- **Dout Value** 輸入所要傳送到IO Port的值

本例811LF預設有8Pin做爲輸出，接下來介紹如何控制。
當要使此8Pin 輸出皆爲High，則在DoutValue輸入0x00FF
00FF中的FF表示8bit的Binary值(11111111)分別對應到下表

| Bit7(IO7) | Bit6(IO6) | Bit5(IO5) | Bit4(IO4) | Bit3(IO3) | Bit2(IO2) | Bit1(IO1) | Bit0(IO0) |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

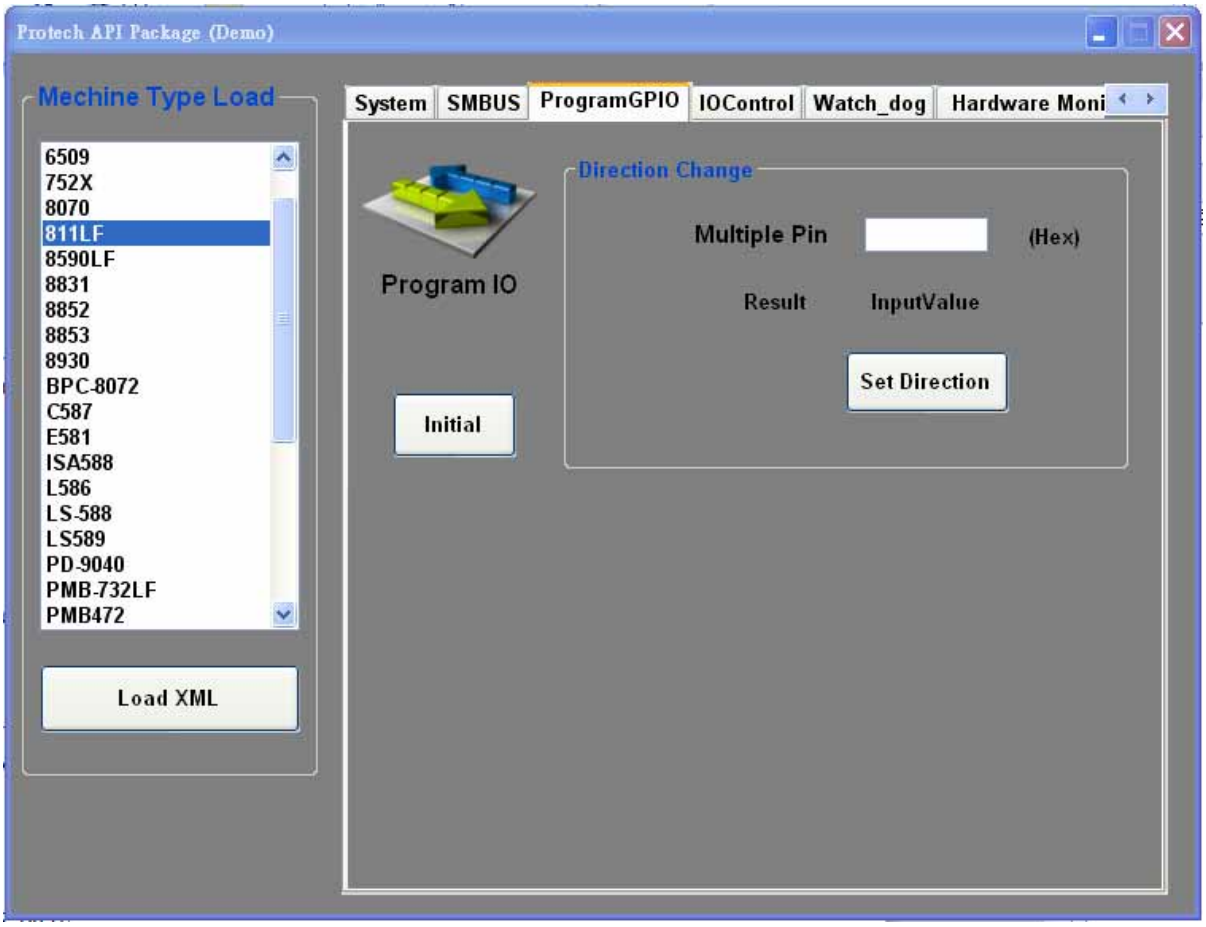
同理此8Pin要輸出皆爲Low則填入0x0000。

當爲4in/ 4out型式，則填入0F (即後4BIT表示要控制的IO Pin位置)

| N/A | N/A | N/A | N/A | Bit3(IO3) | Bit2(IO2) | Bit1(IO1) | Bit0(IO0) |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

- **[Write]** 將DoutValue輸出，將所填入的數值反應至硬體上。
- **[Read]** 讀取Input輸入的數值並將結果顯示到Result上。
- **Result** 將輸入的結果顯示出來，顯示的進制爲16進制值。

## 第2節 Program GPIO




**[Initial]**

初始化IO Function，初始化OK圖示如右:


🕮 若無顯示此圖示之後的操作，都可能無效。

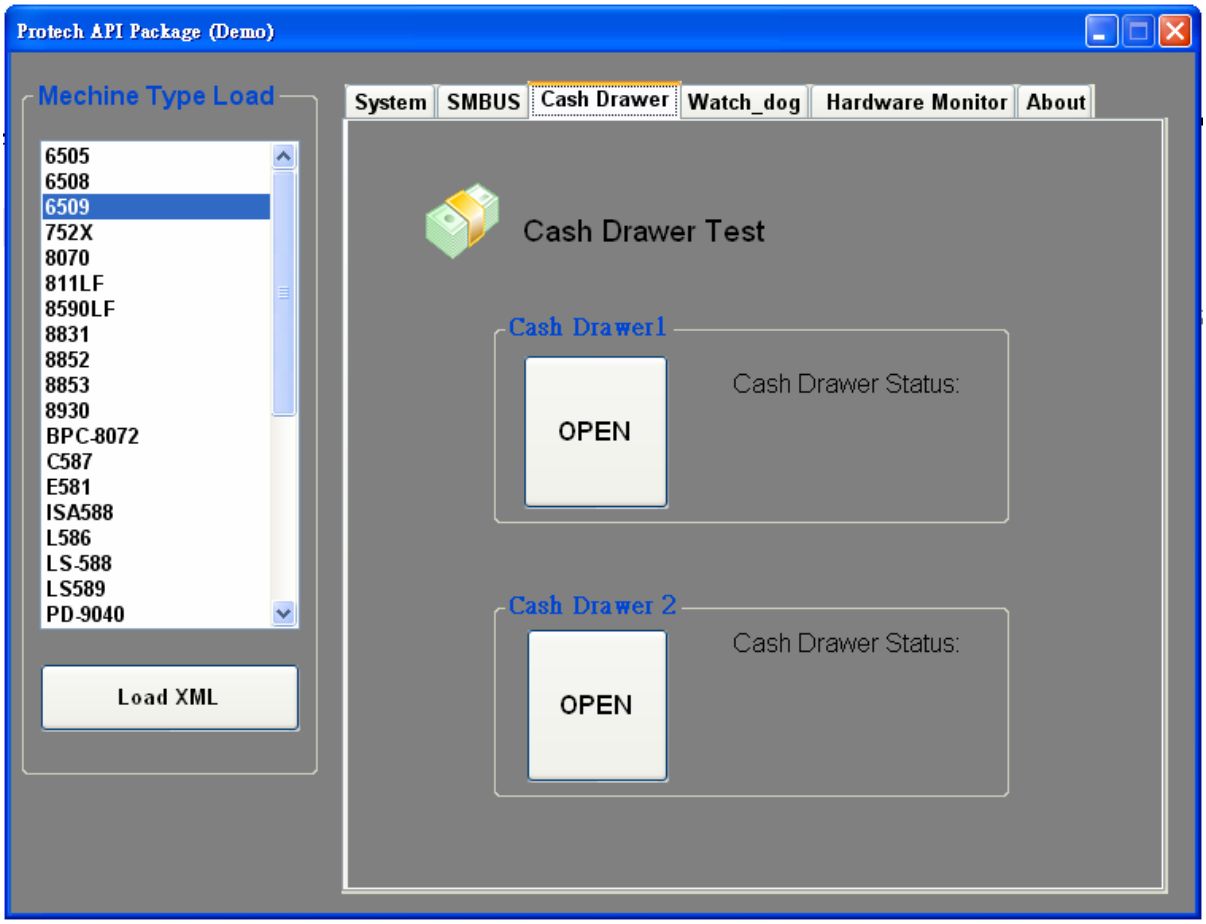
**Direction Change**

- **Multiple Pin** 控制Pin爲輸出或輸入。
  Binary “1”爲設定輸出; Binary “0”爲設定輸入。
  本例811LF預設爲8in/ 8out型態，每根Pin皆可設定輸出或輸入。
  當在Multiple Pin Text Box內輸入“FFFF”即表示全部設定爲“輸出型態”表示現在811LF爲16 Pin皆可輸出型態
  (MSB) F F F F (LSB) 分別對應bit16 ~ bit1
  🕮 當重開機之後會回到初始狀態(8in/ 8out型態)

- **[Set Direction]** 將Multiple Pin text box內所設定的值,設定並反應至系統IO上,改變輸入/ 輸出狀態

- **Result** 回傳值，回傳true(設定成功) or false(設定失敗)

# 第3節 Cash Drawer




**[OPEN]**
開啓Cash drawer

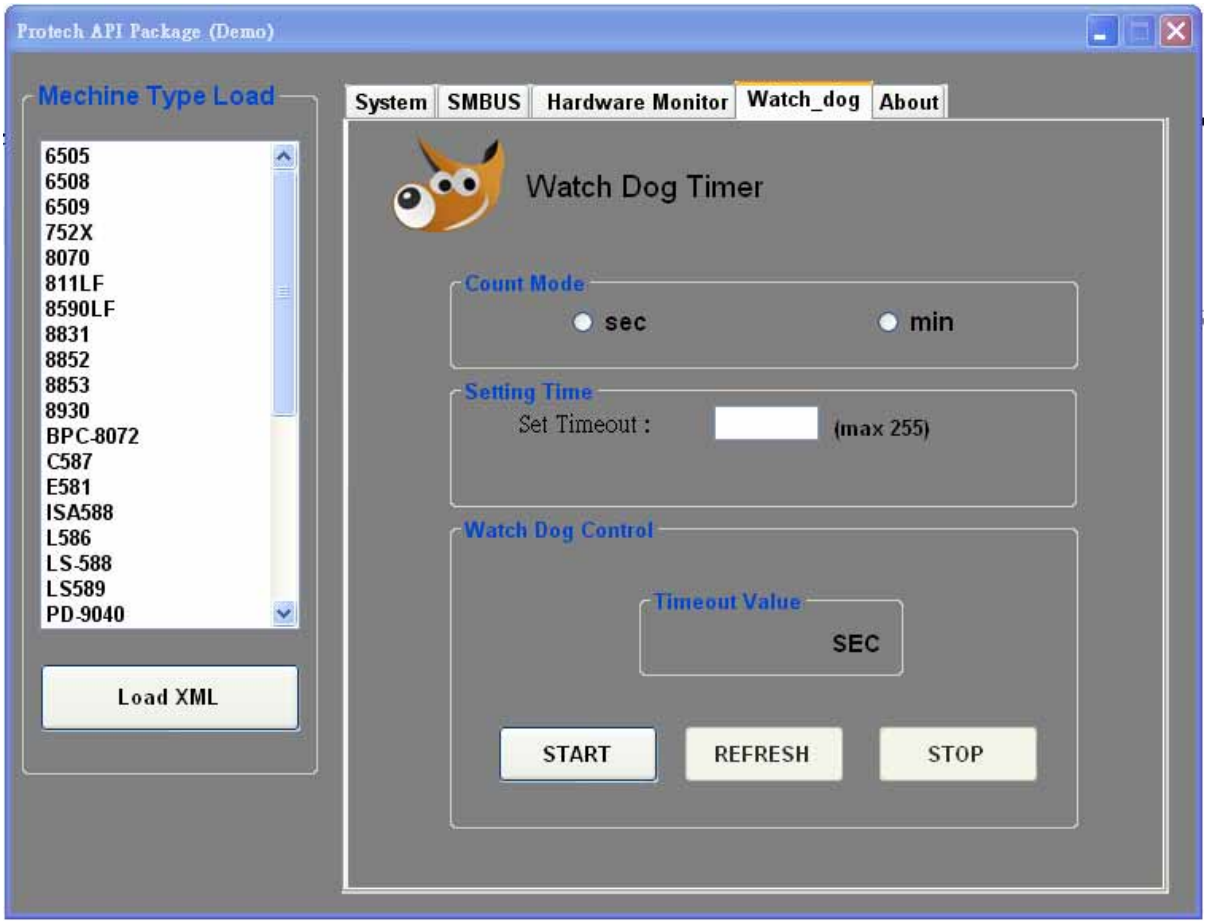
**Cash Drawer Status**
偵測目前Cash drawer開關狀態

- Cash drawer爲關閉狀態圖示如右:


- Cash drawer爲開啓狀態圖示如右:


🕮 本例PS-6509有兩個Cash drawer, ProxAP.exe創建兩個Cash drawer按鍵。
如果該機型爲單一cash drawer port，將創建一個按鍵；當機器只支援一個Cash Drawer時則只顯示一個Cash Drawer。

# 第4節 Watch Dog




## Count Mode

Watch dog設定計數時間單位爲sec或min

## Setting Time

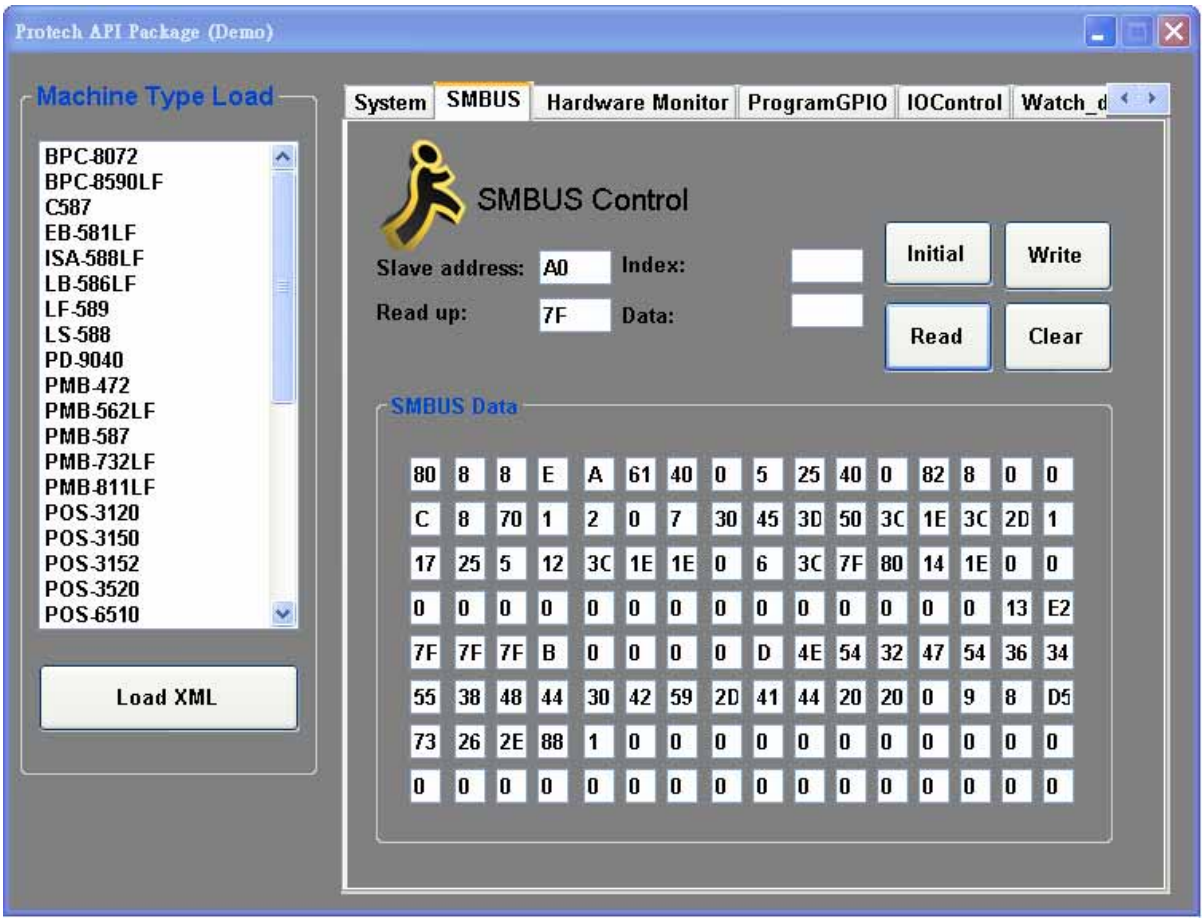
- **Set Timeout** 設定watch time-out的時間

## Watch Dog Control

- **Timeout Value** 顯示目前Watch Dog的時間，此爲軟體模擬的Timer，並非Watch dog的硬體時鐘，所以並不是非常準確
- **[START]** 當**[START]**被按下後, **[REFRESH]**與**[STOP]**被Enable, WDT Timer被啓動
- **[STOP]** 停止watch dog
- **[REFRESH]** 當按下**[REFRESH]**會去做RecountWatchdog動作

# 第5節 SMBUS

提供使用者測試SMBUS Bus上所掛載的裝置狀態




**[Initial]**
初始化SMBUS AP

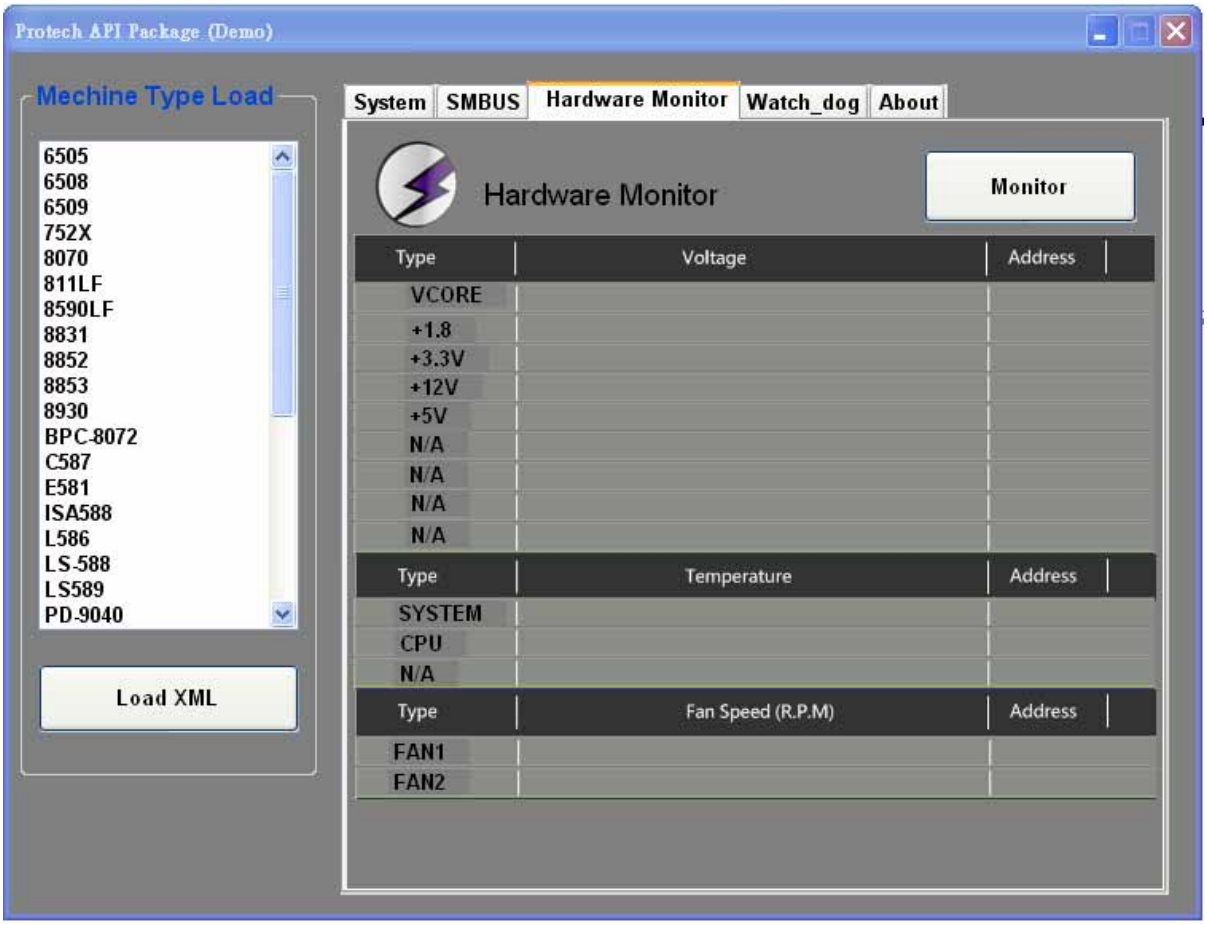
**Slave Address**
設定欲讀取的SMBUS裝置位置

| ▸ 讀取資料 | ▸ 寫入資料 |
|---|---|
| **Read up**<br>設定讀取數量<br><br>**[Read]**<br>讀取資料到下方的文字方塊 | **Index**<br>設定寫入資料的位置<br>**Data**<br>設定要寫入的資料<br>**[Write]**<br>寫入資料到下方的文字方塊 |

**SMBUS Data**
當**[Read]**與**[Write]**被按下後，讀取或寫入的資料將顯示在下方的文字方塊

**[Clear]**
清除下方的文字方塊

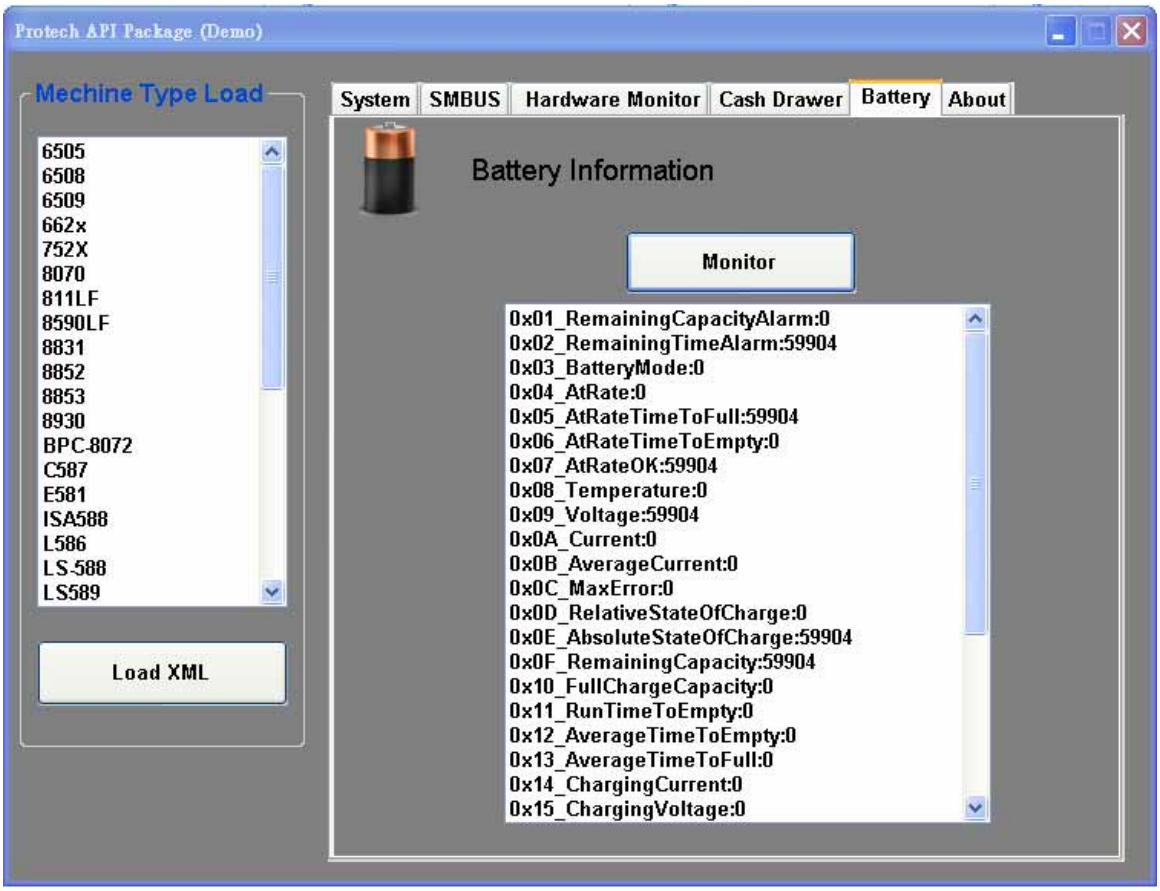
## 第6節 Hardware Monitor




**[Monitor]**
讀取Hardware Monitor參數值，爲電壓、溫度、風扇轉速

🕮 讀取參數會因機型變化。

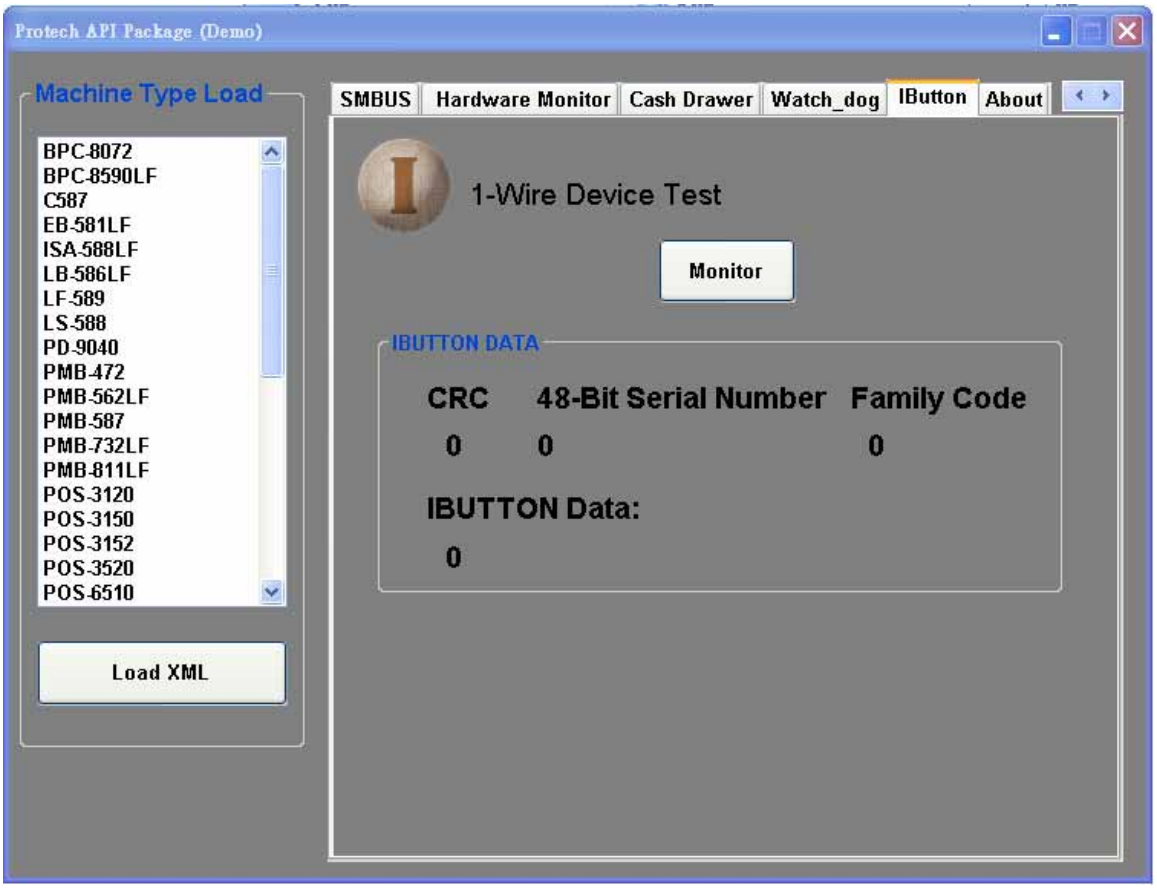
# 第7節 Battery




**[Monitor]**
讀取UPS參數值

📖 讀取參數會因機型變化。

## 第8節 I-Button




**[Monitor]**
開始讀取I-Button資料

# 第4章 程式開發

4

第4章將說明程式開發時所使用到的相關函式。

本章節次內容如下：

# 第1節 API Function

我們提供API程式以及相關使用範例，讓使用者容易的使用API Package，範例程式使用VB.NET與C#爲開發環境，主要API Function如下表所示:

| API Function | | DLL | |
|---|---|---|---|
| **Digital IO** | Digital_Initial<br>Digital_Set<br>Digital_Get | multilangXML.dll | Digital.dll |
| **GPIO (IO)** | GPIO_Initial<br>GPIO_SetPort<br>GPIO_Set<br>GPIO_Get | | GPIO.dll |
| **Cash Drawer** | CashDrawerOpen<br>GetCashDrawerStatus | | CashDrawer.dll |
| **Watchdog (WD)** | Watchdog_Set<br>Watchdog_Stop<br>Watchdog_SetMinSec<br>Watchdog_Recount | | WatchDog.dll |
| **Hardware Monitor** | HMWVoltage_Get<br>HWMTemperature_Get<br>HWMFanSpeed_Get | | Hardware Montior.dll |
| **SMBUS** | SMBUS_Initialization<br>SMBUS_Write<br>SMBUS_Read | | SM_Control.dll |

# 第2節 Digital IO Function

## Digital_Initial

**Bool Digital_Initial ( ) ;**

目 的 初始化Digital API Package
參 數 None
回傳值 True (1) 表示成功, False (0)表示失敗

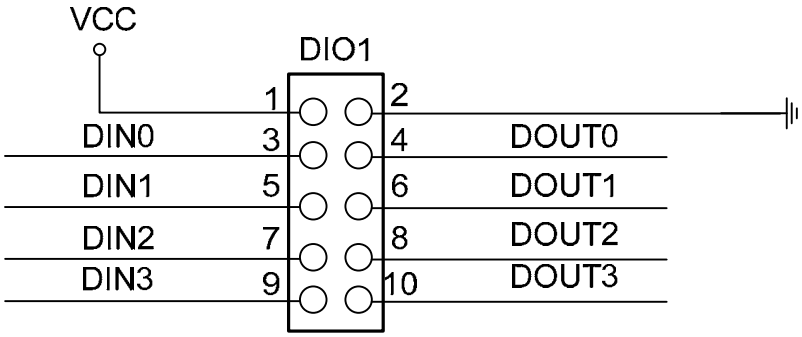
🕮 使用API Package之前必須先呼叫Digital_Inital, 讓XML內變數傳遞到DLL內

## Digital_Set

**Bool Digital_Set (short hex_value);**

參 數 hex_value bit0 ~ bit3分別對應到Digital output上
回傳值 True (1) 表示成功, False (0)表示失敗

如下圖所示 (4in/ 4out型式):




Hex_value "1" (high) "0" (low)

Example Digtial_Set(0x01); // Set DOUT0 as High

Digtial_Set(0x09); //1001 DOUT3 and DOUT0 are High

## Digital_Get

**Short Digital_Get (void);**

目　的　取得Digital輸入狀態
參　數　None
回傳值　回傳Digital輸入pin邏輯狀態

Example
```
Short data;
data = Digital_Get();          //data as DIN High Low change status
```

# 第3節 GPIO Function

## GPIO_Initial

**Bool GPIO_Initial (void);**

目 的 初始化GPIO API Package
參 數 None
回傳值 True (1) 表示成功, False (0)表示失敗

🕮 在使用API之前必須先呼叫此函式

## GPIO_Set

**Bool GPIO_Set (long dout_value)**

目 的 設定GPIO邏輯狀態
參 數 dout_value (16進制) bit1 ~ bit8分別代表每個GPIO Pin
回傳值 True (1) 表示成功, False (0)表示失敗

## GPIO_Get

**Bool GPIO_Get ( )**

目 的 取得目前GPIO的輸入狀態
參 數 None
回傳值 回傳GPIO輸入pin邏輯狀態

🕮 當要讀取輸入pin時, 請注意是否將該pin設定爲輸入狀態

## GPIO_Setport

**Bool GPIO_SetPort (long DirectValue)**

目 的 設定GPIO目前爲輸出或輸入狀態
參 數 DirectValue (16進制)設定爲輸入或輸出狀態
回傳值 True (1) 表示成功, False (0)表示失敗

Example GPIO_Set(0x11); 0x11 => 00010001
GPIO0 -> set to output
GPIO4 -> set to output

# 第4節 Cash Drawer Function

## CashDrawerOpen

**Bool fnPCashDrawerOpen (short num_drawer);**

目 的 設定Cash drawer API開啓
參 數 num_drawer 1 = Open cash drawer1 port
2 = Open cash drawer2 port
回傳值 True (1) 表示成功, False (0)表示失敗

Example CashDrawerOpen(0x01); Cash Drawer 1 open

## GetCashDrawerStatus

**Bool GetCashDrawerStatus (short num_drawer);**

目 的 取得目前Cash Drawer開關狀態
參 數 num_drawer = 1 取得目前cash drawer1狀態
2 取得目前cash drawer2狀態
回傳值 True (1) 表示成功, False (0)表示失敗

Example
```
Short data;
data= GetCashDrawerStatus(0x01); Cash Drawer 1
if (data)
MsgBox(“open1”)//cash drawer1 “open”
Else
MsgBox(“open2”)//cash drawer1 “close”
Endif
```

# 第5節 Watch Dog Function

## Watchdog_Set

**Bool Watchdog_Set (int value)**

參 數 設定watch dog timer的值
回傳值 True (1) 表示成功, False (0)表示失敗

## Watchdog_SetMinSec

**Bool Watchdog_SetMinSec (int kind)**

參 數 Kind = 1 (Sec)
2 (Min)
回傳值 True (1) 表示成功, False (0)表示失敗

## Watchdog_Stop

**Bool Watchdog_Stop (void)**

參 數 None
回傳值 True (1) 表示成功, False (0)表示失敗

## Watchdog_Recount

**Bool Watchdog_Recount (void)**

參 數 None
回傳值 True (1) 表示成功, False (0)表示失敗

# 第6節 Hardware Monitor Function

## HMWVoltage_Get

**Float HMWVoltage_Get (short VoltType)**

參 數

| VoltType | W83627HF | W83627EHF | SMSC3114 | W83627UHG |
|---|---|---|---|---|
| 0x01 | VCoreA | CPU VCore | N/A | VCore |
| 0x02 | VCoreB | VIN0 | +1.5V | VIN0 |
| 0x03 | +3.3VIN | AVCC | N/A | AVCC |
| 0x04 | +5VIN | +3VCC | +5VIN | 5VCC |
| 0x05 | +12VIN | VIN1 | +12V | VIN1 |
| 0x06 | -12VIN | VIN2 | N/A | VIN2 |
| 0x07 | -5VIN | VIN3 | N/A | N/A |

回傳值 回傳float型別資料顯示相關電壓參數

## HMWTemperature_Get

**Float HMWTemperature_Get (short TempType)**

參 數

| TempType | W83627HF | W83627EHF | SMSC3114 | W83627UHG |
|---|---|---|---|---|
| 0x01 | CPU temperature | System temperature | CPU temperature | CPU temperature |
| 0x02 | N/A | CPU2 temperature | N/A | N/A |
| 0x03 | N/A | N/A | N/A | N/A |

回傳值 回傳float型別資料顯示相關溫度參數

## HMWFanSpeed_Get

**Float HMWFanSpeed_Get (short FanType)**

參 數

| FanType | W83627HF | W83627EHF | SMSC3114 | W83627UHG |
|---|---|---|---|---|
| 0x01 | Fan1 | SysFanIN | FAN1 | FAN1 |
| 0x02 | Fan2 | CPUFANIN | FAN2 | FAN2 |
| 0x03 | N/A | AUXFANIN | N/A | N/A |

回傳值 回傳float型別資料顯示相關風扇轉速參數 (單位: r.p.m)

# 第7節 SMBUS Function

## SMBUS_Initialization

**bool SMBUS_Initialization (int Device)**

目 的 初始化SMBUS API Package並設定SMBUS初始位置
參 數 None
回傳值 True (1) 表示成功, False (0)表示失敗

## SMBUS_Read

**int SMBUS_Read (int Index)**

目 的 讀取SMBUS資料
參 數 Index: 讀取位置
回傳值 Byte Array陣列, 爲回傳資料

## SMBUS_Write

**bool SMBUS_Write(int Index, int data)**

目 的 寫入SMBUS資料
參 數 Index: 寫入位置
Data: 寫入的值
回傳值 True (1) 表示成功, False (0)表示失敗

# 第8節 UPS Function

## Initialization

**bool SMBUS_Initialization (int Device)**

參　數　Device = 0x16 (The bq20z90/bq20z95 SBS Device Address)
回傳值　True(1) success, (0) failed

## RemainingCapacityAlarm

**uint RemainingCapacityAlarm()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## RemainingTimeAlarm

**uint RemainingTimeAlarm()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## BatteryMode

**byte BatteryMode()**

參　數　None
回傳值　Hex value with a range of 0 to 0xe383

## AtRate

**int AtRate()**

參　數　None
回傳值　signed int value with a range of -32768 to 32767

## AtRateTimeToFull

**uint AtRateTimeToFull()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## AtRateTimeToEmpty

**uint AtRateTimeToEmpty()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## AtRateOK

**uint AtRateOK()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## Temperature

**uint Temperature()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## Voltage

**uint Voltage()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## Current

**int Current()**

參　數　None
回傳值　signed int value with a range of -32768 to 32767

## AverageCurrent

**int AverageCurrent()**

參　數　None
回傳值　signed int value with a range of -32768 to 32767

## MaxError

**uint MaxError()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 100

## RelativeStateOfCharge

**uint RelativeStateOfCharge()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 100

## AbsoluteStateOfCharge

**uint AbsoluteStateOfCharge()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 100

## RemainingCapacity

**uint RemainingCapacity()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## FullChargeCapacity

**uint FullChargeCapacity()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## RunTimeToEmpty

**uint RunTimeToEmpty()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## AverageTimeToEmpty

**uint AverageTimeToEmpty()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## AverageTimeToFull

**uint AverageTimeToFull()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## ChargingCurrent

**uint ChargingCurrent()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## ChargingVoltage

**uint ChargingVoltage()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65534

## BatteryStatus

**uint BatteryStatus()**

參　數　None
回傳值　Unsigned int value with a range of 0x0000 to 0xdbff

## CycleCount

**uint CycleCount()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## DesignCapacity

**uint DesignCapacity()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## DesignVoltage

**uint DesignVoltage()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## SpecificationInfo

**byte SpecificationInfo()**

參　數　None
回傳值　Hex value with a range of 0 to 0xFFFF

## CellBoltage01

**uint CellBoltage01()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## CellBoltage02

**uint CellBoltage02()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## CellBoltage03

**uint CellBoltage03()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

## CellBoltage04

**uint CellBoltage04()**

參　數　None
回傳值　Unsigned int value with a range of 0 to 65535

### SBS Command Values

| Name | Format | Size in Bytes | Min Value | Max Value | Default Value | Unit |
|---|---|---|---|---|---|---|
| RemainingCapacityAlarm | unsigned int | 2 | 0 | 65535 | 300 | mAh or 10mWh |
| RemainingTimeAlarm | unsigned int | 2 | 0 | 65535 | 10 | min |
| BatteryMode | hex | 2 | 0x0000 | 0xe383 | — | |
| AtRate | signed int | 2 | -32768 | 32767 | — | mA or 10mW |
| AtRateTimeToFull | unsigned int | 2 | 0 | 65534 | — | min |
| AtRateTimeToEmpty | unsigned int | 2 | 0 | 65534 | — | min |
| AtRateOK | unsigned int | 2 | 0 | 65535 | — | |
| Temperature | unsigned int | 2 | 0 | 65535 | — | 0.1 K |
| Voltage | unsigned int | 2 | 0 | 65535 | — | mV |
| Current | signed int | 2 | -32768 | 32767 | — | mA |
| AverageCurrent | signed int | 2 | -32768 | 32767 | — | mA |
| MaxError | unsigned int | 1 | 0 | 100 | — | % |
| RelativeStateOfCharge | unsigned int | 1 | 0 | 100 | — | % |
| AbsoluteStateOfCharge | unsigned int | 1 | 0 | 100+ | — | % |
| RemainingCapacity | unsigned int | 2 | 0 | 65535 | — | mAh or 10mWh |
| FullChargeCapacity | unsigned int | 2 | 0 | 65535 | — | mAh or 10mWh |

(continued)

| Name | Format | Size in Bytes | Min Value | Max Value | Default Value | Unit |
|---|---|---|---|---|---|---|
| RunTimeToEmpty | unsigned int | 2 | 0 | 65534 | — | min |
| AverageTimeToEmpty | unsigned int | 2 | 0 | 65534 | — | min |
| AverageTimeToFull | unsigned int | 2 | 0 | 65534 | — | min |
| ChargingCurrent | unsigned int | 2 | 0 | 65534 | — | mA |
| ChargingVoltage | unsigned int | 2 | 0 | 65534 | — | mV |
| BatteryStatus | unsigned int | 2 | 0x0000 | 0xdbff | — | |
| CycleCount | unsigned int | 2 | 0 | 65535 | — | |
| DesignCapacity | unsigned int | 2 | 0 | 65535 | 4400 | mAh or 10mWh |
| DesignVoltage | unsigned int | 2 | 0 | 65535 | 14400 | mV |
| SpecificationInfo | hex | 2 | 0x0000 | 0xffff | 0x0031 | |
| CellVoltage4 | unsigned int | 2 | 0 | 65535 | — | mV |
| CellVoltage3 | unsigned int | 2 | 0 | 65535 | — | mV |
| CellVoltage2 | unsigned int | 2 | 0 | 65535 | — | mV |
| CellVoltage1 | unsigned int | 2 | 0 | 65535 | — | mV |

# 第9節 I-Button Function

## Decode_Ibutton_Process

**bool Decode_Ibutton_Process(short[] buffer)**

參　數　buffer = ibutton read will sent to this buffer
回傳值　True(1) success, (0) failed

# 附錄A FAQ

A

第5章將解釋使用API Package可能會產生的疑問。

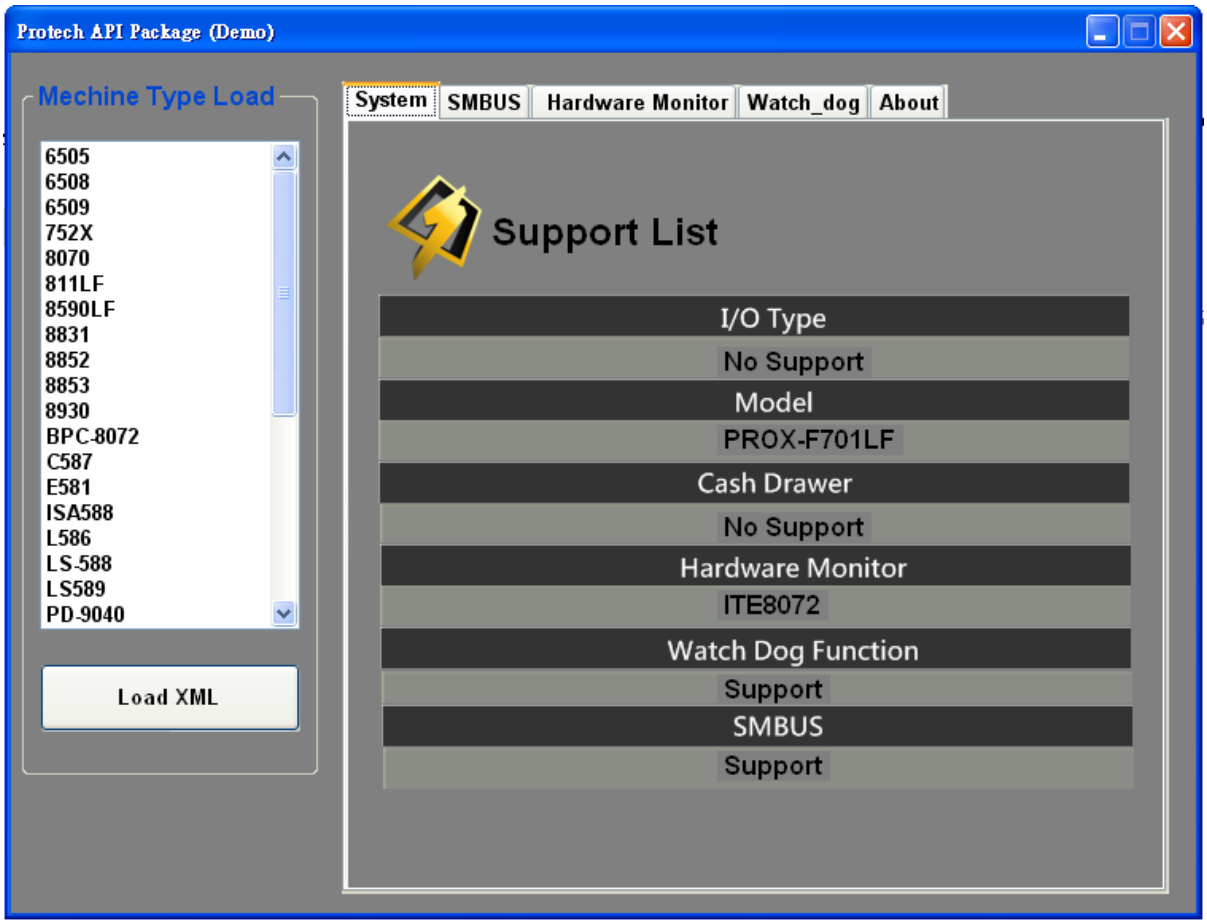
本章節次內容如下：

## 第1節 Demo AP.exe無法開啟?

**Answer:** 可能原因有兩種:
(1).使用者未曾安裝過.net framework 2.0或以上版本, 需先安裝
(2).API Package內無XML File導致一開始AP無法啓動。

## 第2節 如何確定XML檔案是否正確?

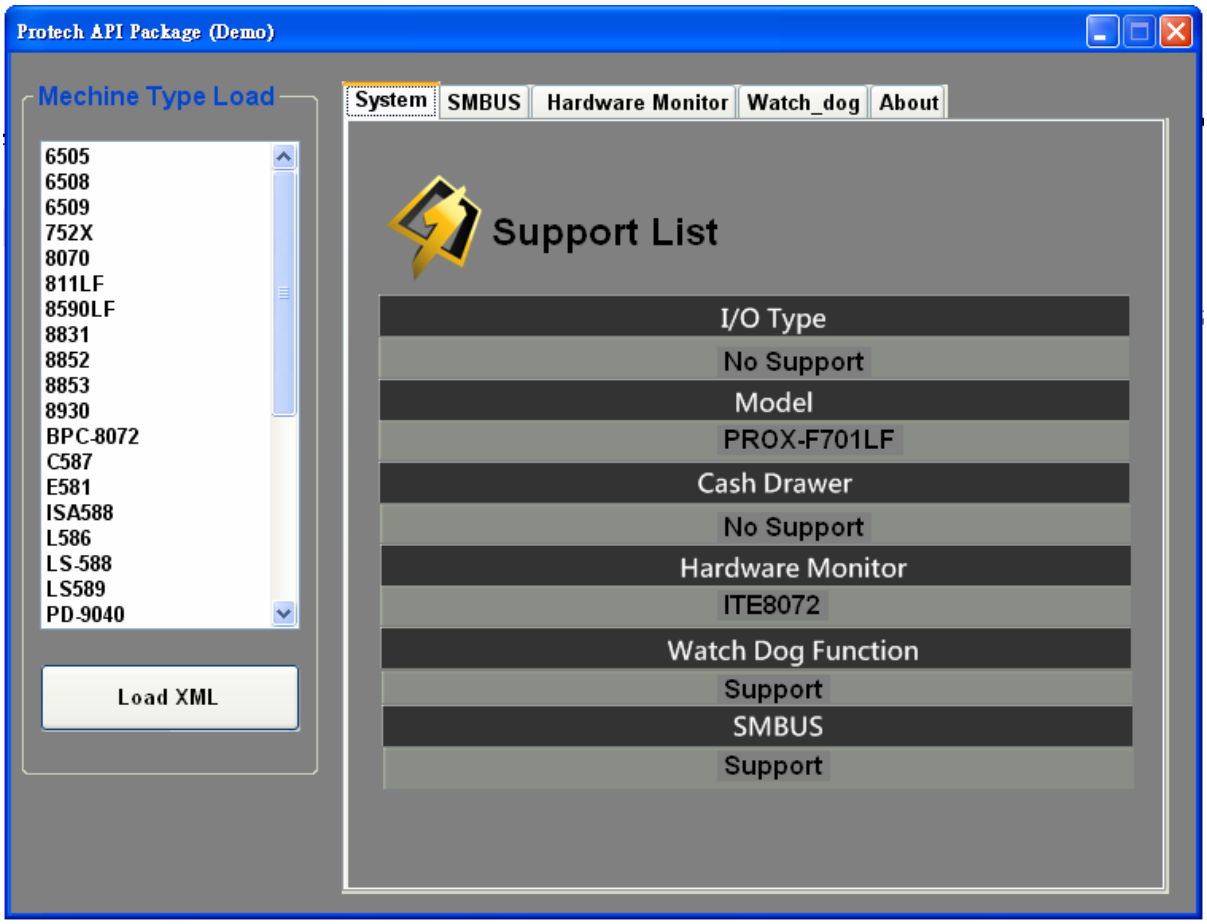
**Answer:** 在Demo AP一開始即可確認所使用的XML檔案是否正確



🕮 ProxAPI standard資料夾內的Initial.xml針對不同機型需要做抽換。

以下爲抽換XML方式
如果想使用PS3100，需先到API Package File內XML File ->PS3100 -> Initial.xml複製貼到ProxAPI standard，將原本Initial.xml覆蓋掉即可。

當覆蓋掉之後，開啓程式即可在system頁籤中驗證所使用之AP正確!
如下圖所示，程式顯示出3100 Cash Drawer功能，而此功能並未包含在811LF內。



## 第3節 套用XML有些功能在Support List上沒有顯示?

**Answer:** 因機台不同有些功能不支援關係, 例如PS3100無Digital, 所以在system頁面上I/O Type則顯示不支援, 相對IO頁面亦不會顯示。

## 第4節 使用者自行開發AP時無法執行?

**Answer:** 使用者必須先將把整個API Package檔案放置所開發的路徑檔案下, 設定好連結方式, 同時並確定XML Initial檔案放置是否正確, 才能確保Function可以正常工作。

## 第5節 Demo Project無法使用時?

**Answer:** Demo Project的release內是有一份API 包, 但是需注意API包內的Initial.xml爲所開發機型的xml檔案, 才能確保API連結正確。

## 第6節 Digital IO與GPIO差異?

**Answer:** Digital IO與GPIO差異在於Digital IO不可更改其IO方向性, 而GPIO可更改其方向性。例如GPIO可設定所有IO皆爲輸出或輸入特性

當使用者開啓AP時, 若爲GPIO, 除了在System頁面上會顯示爲GPIO Function, 同時會帶起Program IO頁面, 則表示該平台支援修改。

預設時我們提供使用者一個規範爲4In/ 4Out型態, 並定義在User Manual中會有說明IO在主板上的位置, 當修改IO方向性時, 重開機之後會自動歸回到4In/ 4Out型態。